

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

EIGENSPACE METHODS AND SUBSET SELECTION IN KERNEL BASED LEARNING

Promotoren:
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. J.A.K. Suykens

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Luc HOEGAERTS

Juni 2005



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

EIGENSPACE METHODS AND SUBSET SELECTION IN KERNEL BASED LEARNING

Jury:
Prof. P. Van Houtte, voorzitter
Prof. J. Vandewalle, promotor
Prof. J. Suykens, promotor
Prof. B. De Moor
Prof. D. Bollé
Prof. S. Vandewalle
Prof. B. Hammer (Osnabrück)
Prof. V. Wertz (UCL)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Luc HOEGAERTS

U.D.C. 681.3*I2 Juni 2005

©Katholieke Universiteit Leuven – Faculteit Toegepaste Wetenschappen
Arenbergkasteel, B-3001 Heverlee (Leuven), Belgium.

Alle rechten voorbehouden. Niets uit deze uitgave mag vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

Wettelijk depot D/2005/7515/46

ISBN 90-5682-612-3

UDC 681.3*I2

Voorwoord

Ruim vier jaar geleden heeft Prof. Joos Vandewalle me de kans gegeven om aan een doctoraat te beginnen. Ik werkte toen als software engineer en wilde me toespitsen op eerder de analyse, dan de allocatie van data. Ik heb dan het geluk gehad me te kunnen aansluiten op de onderzoeksgroep in Support Vector Machines van Prof. Johan Suykens. En dan, over de tijd, heeft een ingewikkelde combinatie van discussies, voorstellen, ideeën en mogelijkheden zich gevormd tot een stel van publicaties, waarvan de synthese in dit boekje werd samengevat. Ik ben mijn promotoren zeer erkentelijk voor hun steun, de bijsturing, de kritische opinie, de feedback, de aanmoedigingen en de kans om deze proef te hebben mogen realizeren. Voor een aanzienlijk deel van de thesis ben ik ook veel dank verschuldigd aan dr. Lieven de Lathauwer, die me begeleidde in de (multi)lineaire algebra.

De assessoren van het leescomité wil ik bedanken voor hun begeleiding gedurende de vier onderzoeksjaren en hun constructieve kritiek in verband met het verbeteren van de tekst. Prof. Bart De Moor heeft me geleerd om vooral ook horizontaal en praktisch te denken, over de grenzen van het eigen terrein. Hij heeft me de kans gegeven om mee te werken in het BASF-project i.v.m. softsensor ontwikkeling voor chemische processen en het ELIA-project i.v.m. modellering van vermogen in het Belgische hoogspanningsnetwerk. Prof. Désiré Bollé ben ik erkentelijk voor het zorgvuldig nalezen van de tekst en de goede raad betreffende correctheid en formulering. Ik wil ook Prof. Stefaan Vandewalle, Prof. Barbara Hammer en Prof. Vincent Wertz bedanken voor hun bereidheid om deel uit te maken van de jury en Prof. Paul Van Houtte voor het waarnemen van het voorzitterschap van de examencommissie.

Mijn collega's wens ik te danken voor de fijne samenwerking en toffe werksfeer (niet te vergeten de momenten zoals op de sistaweekends, voetbalpartijtjes, conferenties en kletspartijen), o.m. Ivan Goethals, Kristiaan Pelckmans, Bart Hamers, Jos De Brabanter, Oscar Barrero, Marcelo Espinoza, Evelyne Dewitte, Tom Van Herpe, Bert Pluymers, Raf Van de Plas, Dries Van Dromme, Nathalie Pochet, Sven Maerivoet, Mieke Schuermans, Maarten Van den Nest, Patrick Glennisson, Ida Tassens, Tony Van Gestel, Jeroen Boets, Jairo Espinosa, en nog vele anderen. Tevens moet ik mijn dank betuigen aan het Instituut voor de aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen voor de financiële steun.

En ten slotte draag ik dit werk op aan mijn moeder, die zeker heel trots geweest zou zijn.

Abstract

The last decades have been characterized by a considerable increase in the amount and complexity of information, far exceeding the ability to process and interpret the data. Inference tools for data mining and knowledge extraction draw upon techniques from artificial intelligence, machine learning and statistics. In this thesis we generalize or extend some of those methods that are based on the kernel paradigm. This is the latest generation of models that have the capability to learn (non)linear relations from a given collection of pairwise examples, like for example in classification or regression problems. We investigated upon five main topics: (i) we derive a primal-dual interpretation for Kernel Partial Least Squares and provide a unifying framework for a set of existing least squares kernel based regression methods in a Reproducing Kernel Hilbert space, (ii) we endow these methods with a sparse formulation via the Nyström approximation, which permits to perform kernel regression on very large data sets with limited loss of accuracy, (iii) we explore possible up/downdating strategies for Least Squares Support Vector Machines, with induced data point selection criteria, (iv) we devise a novel tracking algorithm to efficiently approximate the dominant eigenspace of the kernel matrix and (v) we derive a generalized Grassmann-Rayleigh Quotient Iteration for the computation of the best rank-(R1,R2,R3) approximation of higher-order tensors.

Korte inhoud

De laatste decennia werden gekenmerkt door een aanzienlijke toename van de hoeveelheid en complexiteit van informatie, die de capaciteit om de data te verwerken en te interpreteren ver overtreft. Inferentiemiddelen voor gegevensontginning en kennisextractie vinden hun basis in artificiële intelligentie, lerende machines en statistiek. In deze thesis veralgemenen of breiden we enkele van de methodes uit die gebaseerd zijn op het kernel paradigma. Dit is de laatste generatie van modellen die (niet-)lineaire relaties tussen gegeven paren van voorbeelden kunnen beschrijven, zoals bvb. in classificatie- en regressieproblemen. We verrichtten onderzoek rond vijf thema's: (i) we leiden een primair-duale interpretatie af voor Kernel Partial Least Squares en reiken een unificerend interpretatiekader aan voor een verzameling van reeds bestaande kleinste kwadraten gebaseerde kernel regressie methodes in een Reproducerende Kernel Hilbert ruimte, (ii) we voorzien deze methodes van een spaarse formulering via de Nyström benadering, wat toelaat om kernel regressie op zeer grote data sets toe te passen met beperkt verlies van nauwkeurigheid, (iii) we verkennen mogelijke up/downdating strategieën voor Least Squares Support Vector Machines, met geïnduceerde data punt selectiecriteria, (iv) we ontwikkelen een nieuw tracking algoritme om de dominante eigenruimte van de kernel matrix efficient te benaderen (v) we leiden een veralgemeende Grassmann-Rayleigh Quotient Iteratie af voor de berekening van de beste rang-(R1,R2,R3) benadering van hogere-orde tensoren.

Nederlandse samenvatting

Eigenruimte Methodes en Subset Selectie in Kernel gebaseerde Lerende Systemen

1. Inleiding

Het behoeft geen betoog om te stellen dat enerzijds de hedendaagse mens in overvloedige mate wordt omringd door informatie, maar dat aan de andere kant de nood aan kennis nog sneller groeit. In deze kloof schuilt een enorm potentieel aan vooruitgang en welvaart, dat dringend aangeboord dient te worden. De wetenschap die zich bezighoudt met het afleiden van relaties uit ruwe gegevens is het gebied van de *Gegevensontginning en Kennisextractie*.

Het doel is vooral om dit proces op een automatische, snelle en transparante manier uit te voeren. Daartoe wendt men specifieke methodes en algoritmes aan die bvb. relaties afleiden tussen variabelen of functies afleiden die toelaten om toekomstige waarden te voorspellen of te classificeren. Veelal zijn de technieken afkomstig vanuit *Statistiek, Artificiële Intelligentie en Lerende Systemen*, die aangepast en verbeterd worden om aan de toenemende vereisten en complexiteit te kunnen voldoen. Het is in dit kader, meerbepaald de 'engineering' van methodes in Lerende Systemen, dat dit onderzoekswerk zich situeert.

Dit werk handelt over technieken voor automatisch leren met behulp van een gegeven verzameling van een eindig aantal paarsgewijze voorbeelden (ingang-uitgang). Voorbeelden hiervan zijn classificatie en het schatten van functieverbanden. In deze thesis sluiten we aan bij de laatst ontwikkelde benaderingen die het landschap van Artificiële intelligentie en Lerende Systemen

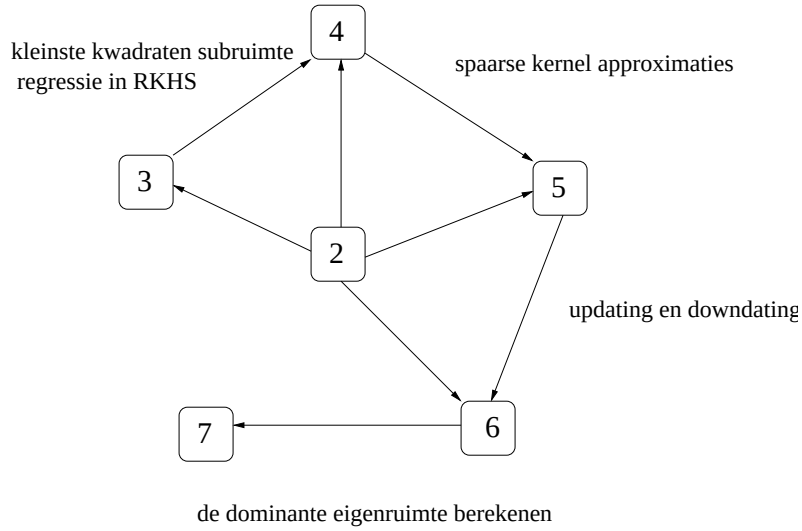
overheersen, namelijk de modelklasse van *kernel methodes*. In dit werk richten we ons voornamelijk op het subdomein van kernel gebaseerde methodes.

Kernel methodes zijn een nieuwe generatie van technieken die het leren van relaties op een stevige theoretische basis grondvest. De eerste klasse van methodes die de theorie implementeerden waren Support Vector Machines (SVM). Het zijn algemene approximatoren (afhankelijk van de gekozen geassocieerde kernel functie) voor de (niet-lineaire) relatie, gekarakteriseerd door de gegeven voorbeeldparen. Hun krachtige modelleringsvermogen en elegantie hebben de attentie van vele onderzoekers getrokken. Onder de andere paradigma's (zoals neurale netwerken, Gaussiaanse Processen) zijn ze een gedegen concurrent geworden met sterke capaciteit. Ze zijn bvb. conceptueel simpel en transparant, ze leveren een eenduidige en globale oplossing en ze zijn minder gevoelig aan de hoge ingangsdimensionaliteit. Maar voor alles hebben zij een nieuwe ruimte gecreëerd met plaats voor uitbreidingen en verbeteringen. Met dit werk beogen we een bijdrage hierin.

Het intuïtieve idee achter kernel methodes is dat de data op een niet-lineare wijze afgebeeld wordt in een hoger-dimensionale 'kenmerken' ruimte, gerelateerd aan een bepaalde kenmerkende niet-lineaire kernel functie. In het algemeen heeft zulke procedure het voordeel dat lineair niet-scheidbare data dan linear scheidbaar wordt. Maar het nadeel is dat berekeningen met de hoog-dimensionele datapunten (te) rekenintensief worden - behalve voor inwendige producten. Bijgevolg kunnen enkel inwendig produkt gebaseerde algoritmes aangewend worden in deze kenmerken ruimte, doch dit is meestal voldoende en voor een grote klasse van algoritmes kunnen we de oplossing in de inputruimte uitdrukken in functie van de oplossing in de kenmerken ruimte. In Hoofdstuk 2 gaan we in op de essenties van het paradigma en schetsen we de nodige achtergrond voor de thesis. In de andere hoofdstukken beschrijven we gevonden oplossingen voor een selectie van enkele problemen en vraagstukken in het domein.

Deze thesis is onderverdeeld in zes hoofdstukken. Alle hoofdstukken zijn zodanig opgesteld dat ze zo veel mogelijk afzonderlijke entiteiten vormen. Hoofdstuk 1 situeert het onderzoek. Hoofdstukken 3-4-5-6 delen een gemeenschappelijke achtergrond die uitgelegd wordt in Hoofdstuk 2, terwijl Hoofdstuk 7 meer op zichzelf staat. In het algemeen kan elk hoofdstuk gezien worden als een logische uitbreiding van het vorige hoofdstuk, zodanig dat een causale ketting wordt gevormd. Deze structuur is voorgesteld in Figuur 0.1.

De inhoud van de hoofdstukken in deze thesis is georganiseerd als volgt. Hoofdstuk 1 geeft een inleiding en beschrijft de achtergrond van dit manuscript in brede algemene termen. In Hoofdstuk 2 gaan we dieper in op de fundamentele concepten van lerende systemen, waarbij we een basis creëeren voor de volgende hoofdstukken met een overzicht van het kernel paradigma. We volgen hoofdzakelijk het standpunt van Statistisch Leren, de hoeksteen van de Support Vector Machines voorgesteld door Vapnik. In Hoofdstuk 3 sluiten



Figuur 0.1: Structureel overzicht van de verbanden tussen de hoofstukken in deze thesis. De pijlen duiden de afhankelijkheid (specifiëring) aan tussen de hoofdstukken. De erbij horende titels duiden het gemeenschappelijke thema aan dat de hoofdstukken verbindt, wat aldus een logische aaneenschakeling vormt.

we aan bij de belangrijke klasse van Least Squares Support Vector Machines (LS-SVM), die zijn gebaseerd op een kleinste kwadraten kost functie, met regularizatie. Naast LS-SVM regressie met een extra afstraffing op de norm van de parameter vector, plaatsen wij een klasse van kernel regressie methodes met restrictie tot een subruimte. In hoofdstuk 4 breiden we dit kleinste kwadraten subruimte regressie raamwerk uit door de introductie van benadering van de subruimte, gebaseerd op een kleiner deel van de training dataset, met als voordeel dat grote datasets kunnen worden verwerkt. In Hoofdstuk 5 verkennen we de mogelijkheden om, net als in vorig hoofdstuk, spaarsheid in het LS-SVM model te introduceren, door enerzijds een studie van snoeimethodes en anderzijds een analyse van foutmaten en hun geïnduceerde selectie criteria om de LS-SVM te up- en downdaten. In Hoofdstuk 6 gaan we dieper in op een update en downdate procedure voor de efficiente berekening van de dominante principale componenten van de kernel matrix, die van belang zijn in vele kernel methodes. In Hoofdstuk 7 connecteren we met het voorgaande hoofdstuk door de berekening van de dominante eigenruimte voor matrices uit te breiden tot die van hogere-orde tensoren. In Hoofdstuk 8 vatten we de resultaten van de thesis en publicaties samen en geven een uitzicht op mogelijke verdere onderzoekspaden.

2. Het kernel paradigma in Lerende Systemen

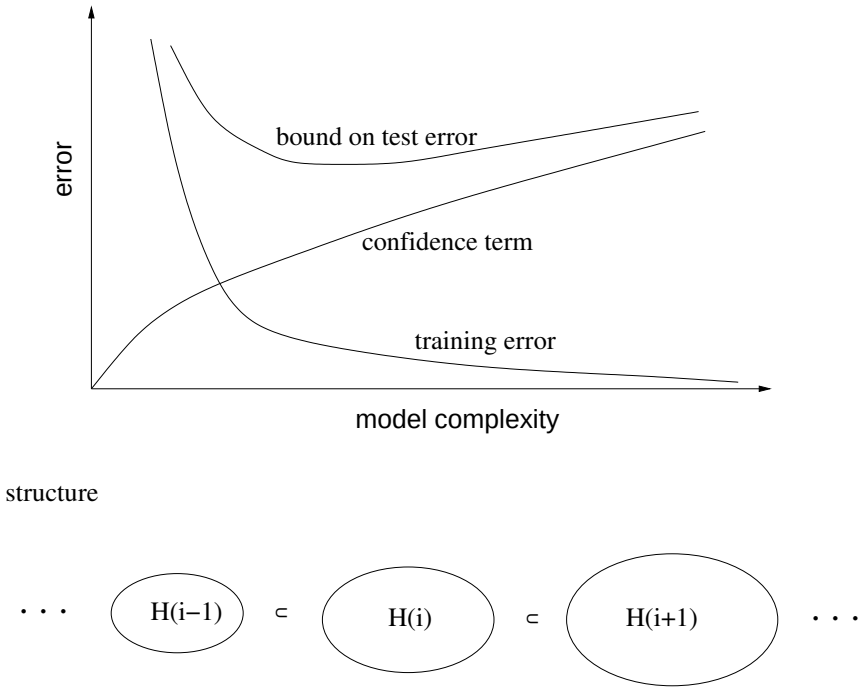
Een lerend systeem wordt verondersteld zichzelf te verbeteren door ervaring (of voorbeelden) met betrekking tot een bepaalde klasse van taken en een performantiemaat [123]. In Lerende Systemen beschouwt men het probleem van de constructie van een model of algoritme om een onderliggende relatie te leren van een gegeven collectie van voorbeelden. We zullen onszelf beperken tot het geval waarbij de voorbeelden bestaan uit paren (input-output) van vectoren (gesuperviseerd leren). Met model bedoelen we een geparametriseerd functievoorschrift, inclusief eventuele veronderstellingen.

Het schatten van de relatie noemt men inferentie en dit kan op verschillende manieren gedaan worden. Traditioneel heeft men bvb. klassieke statistiek, met hypothesetoetsen en maximal likelihood schattingen, en ook Bayesiaanse statistiek, met Occam's razor, explicitering van priors en posterior maximalizatie. Sinds 1950 echter zijn vele uiteenlopende/overlappende gespecialiseerde modellen ontstaan, gaande van meer praktisch tot theoretisch (bvb. decision trees, instance-based learning, rules learning, reinforcement learning, neural networks, computational learning, function approximation theory, etc.) [123, 14]. Het laatste decennium wordt min of meer gedomineerd door *statistical learning theory*. Deze theorie heeft een grote invloed gehad de laatste jaren sinds de publicatie van het werk van Vapnik [183].

De performantie van een model moet aangeven hoe accuraat de geschatte output is van een input. Hiervoor stelt men een risicofunctionaal op die de fout aangeeft per model. De minimalizatie van deze maat over een gegeven verzameling van modellen levert de beste oplossing en is in feite het onderwerp van de variatierekening. Van zodra echter de voorbeelden eindig, discreet en stochastisch van aard zijn, wordt het leerprobleem gecompliceerder. Men neemt dan meestal de verwachtingswaarde van een kostfunctie als risicofunctionaal en een werkhypothese dat de integraal door een som kan worden benaderd (principe van het empirisch risico). Dichtheidsschatting, classificatie en regressie zijn typische voorbeelden van deze benadering. De keuze telkens van de concrete kostfunctie in een bepaalde metriek specifiek aangepast aan de gegeven data is theoretisch meestal te bepalen, maar omwille van praktische redenen (rekentijd, oplosbaarheid) zal men het moeten stellen met een redelijk benaderende risicofunctionaal.

Een bijkomende moeilijkheid, eens men een benaderende risicofunctionaal concreet opstelt, is dat men in de optimalizatiestap te maken krijgt met ill-posedness [67], d.w.z. de oplossing (i) bestaat niet, (ii) is niet uniek of (iii) is niet stabiel (zie bvb. [18]). De remedie voor dit probleem dateert van de jaren 1960 en heet regularizatie [95, 136, 170]: het beperken van de complexiteit van het model, meestal via een extra regulerende parameter. Maar een gedegen onderbouw van het empirisch risico principe [183] toont dat de ill-posedness intrinsiek is, en dat het slechts verschoven werd naar de optimalizatiestap.

Een centraal resultaat in statistical learning toont namelijk aan dat regularizatie (eindige modelklasses) een nodige voorwaarde is opdat de generalizatiefout van het leeralgoritme naar nul zou gaan in de limiet naar oneindig veel data. De precieze details van de formulering van de generalizatiefout leren ons bijkomend dat het kiezen van de complexiteit (structuur van de modelklasses) cruciaal is voor de convergentie(snelheid). Dit suggereert dat men principieel moet minimizeren over geneste modelklasses (capaciteitscontrole). Aan de ene kant daalt het empirische risico met stijgende complexiteit, terwijl aan de andere kant de generalizatiefout (verschil tussen echte en beschouwde empirische risico) stijgt met complexiteit. Een afweging dient te worden gemaakt (structureel risico principe), geïllustreerd in Figuur 0.2.



Figuur 0.2: Illustratie van het principe van gestructureerde risico minimalizering. De bovengrens op de generalizatiefout hangt af van de model complexiteit.

Een belangrijk algoritme dat het structureel risico principe implementeerde was de lineaire classificator van Vapnik [181] in 1963 om een optimaal hypervlak te vinden tussen twee lineair scheidbare klasses: het is een lineaire beslissingfunctie die de klasses scheidt en de grootste marge laat tussen de vectoren van de twee klasses. Het was opmerkelijk dat dit hypervlak werd bepaald door slechts een fractie van de data vectoren, de zogenaamde *support vectoren*.

Pas in 1992 werd dit algoritme gegeneralizeerd naar een niet-lineaire classificator [17]. In principe verwisselde men het inwendig product door een inwendig

product afkomstig van een *niet-lineaire kernel functie*, terwijl de input vectoren naar een hoog-dimensionele *kenmerken ruimte* werden afgebeeld. Deze *kernel truuk* introduceert enorm veel flexibiliteit, en beperkt toch nog de complexiteit van de classificator en de oplossing.

De modelklasse die met dit speciale inwendig produkt overeenstemt bleek een ideale implementatie te zijn voor de combinatie van niet-lineariteit en structureel risico minimalizeren tegelijkertijd: de Reproducerende Kernel Hilbert ruimte (RKHS). Het is een ruimte die geïnduceerd wordt door een kernelfunctie k en daardoor enkel begrensde functies bevat. De kernelfunctie k neemt vaak de interpretatie aan van een similariteitsfunctie tussen de punten, bepaalt de niet-lineariteit van het lerend algoritme en veronderstelt dat de punten $\mathbf{x}$ afgebeeld worden op hoog-dimensionale kenmerk vectoren $\varphi(\mathbf{x})$. De pointe is dat inwendige producten tussen de afgebeelde punten nooit expliciet moeten berekend worden, maar omzeild worden doordat het gelijk is aan de kernel evaluatie van de originele input vectoren. Natuurlijk moet het algoritme wel met inwendige producten kunnen uitgedrukt worden, maar dit laat nog voldoende vrijheid.

Een belangrijk resultaat van de regularizatie methode via een RKHS is dat men kan aantonen dat de optimale oplossing bestaat in een eindig-dimensionele deelruimte, opgespannen door de kenmerk vectoren (nonparametric representer theorema [151]). Een ander sterk resultaat is dat ook de universele approximatie eigenschap, nl. dat elke continue functie willekeurig dicht kan benaderd worden, is gegarandeerd als de dimensie van de RKHS oneindig en de kernel positief definiet zijn.

En ten slotte, in 1995, stelden Cortes and Vapnik [29] een veralgemenende niet-lineaire classificator voor, voor niet-scheidbare klasses. Vapnik generalizeerde deze Support Vector Machine (SVM) ook voor het schatten van niet-lineaire reële functies [183], met hoofdzakelijk de ϵ-gebaseerde kostfunctie. De SVM machines en het kernel paradigma hebben een nieuw onderzoeksveld opengelegd, met nieuwe vragen, met enorme ruimte voor uitbreidingen en veralgemeningen.

Kernel gebaseerde algoritmes hebben vele gewenste eigenschappen en zijn erg succesvol in leerproblemen in vergelijking met andere methodes. Het bracht vooral een verschuiving naar convexe kostfuncties op gang, waardoor een unieke en globale oplossing mogelijk werd, met transparante primair-duale interpretaties. Kernel methodes hebben een kleinere gevoeligheid van de kostfunctie en generalizatiefout aan hoge inputdimensies. Deze modellen kunnen veelal uitgedrukt worden in een beperkt aantal support vectoren (spaarsheid) en niet-lineariteit kan op een flexibele manier geïntegreerd worden in bestaande algoritmes. De SVM methodologie vormt een brede basis waarop nieuwe uitbreidingen kunnen gebouwd worden. In het volgende hoofdstuk vertrekken we van kleinste kwadraten gebaseerde kost functies voor regressie- en classificatieproblemen.

3. Kleinste Kwadraten Kernel Machines

Een aanpassing van de Support Vector Machine (SVM) is de Least Squares Support Vector Machine (LS-SVM), uitgewerkt door Suykens in [167], die de SVM formulering vereenvoudigde zonder enige van de voordelen op te geven (behalve spaarsheid). Beschouwt men een afhankelijke variabele variabele y, onderhevig aan ruis, en data $\{(\mathbf{x_i}, y_i)\}_{i=1}^n \in \mathbb{R}^p \times \mathbb{R}$ gegeven, dan gaat deze machine uit van een lineair model met bias in een RKHS [165]:

$$y = \mathbf{w}^T \varphi(\mathbf{x}) + b,$$

waar de kenmerken afbeelding $\varphi : \mathbb{R}^p \to \mathcal{H}_k$ een functie is in de r-dimensionele Reproducerende Kernel Hilbert ruimte (RKHS) [5] $\mathcal{H}_k$ met de geassocieerde kernel functie $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j)$. Een typische keuze is de Gaussiaanse kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/h^2)$, waar h een (positive) kernel bandbreedte parameter voorstelt; of de polynomiale kernel $(\mathbf{x}_i^T \mathbf{x}_j + a)^d$ van orde $d \in \mathbb{N}$ en constante $a \in \mathbb{R}_0^+$. De gewichten in de vector $\mathbf{w}$ en de bias b worden geschat door een minimalizering van de kleinste kwadraten kostfunctie kostfunctie in de *primaire ruimte*:

$$\begin{aligned} &\min_{\mathbf{w},b,\mathbf{e}} J(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^n e_i^2 \\ &\text{subject to} \quad y_i = \mathbf{w}^T\varphi(\mathbf{x}_i) + b + e_i, \quad i = 1, \ldots, n \end{aligned}$$

Dit objectief bestaat uit een kleinste som van fouten in het kwadraat term (om de functie aan de data aan te passen, fitting) en een regularizatieterm om de benaderende functie glad te maken (om overfitting te vermijden). De relatieve belangrijkheid van deze twee termen wordt bepaald door de positieve constante γ. Het werken met de expliciete uitdrukking voor φ wordt omzeild door een kernel te kiezen en over te gaan op het duale probleem via de Lagrangiaan

$$L(\mathbf{w}, b, \mathbf{e}; \boldsymbol{\alpha}) = J(\mathbf{w}, b, \mathbf{e}) - \sum_{i=1}^n \alpha_i(\mathbf{w}^T\varphi(\mathbf{x}_i) + b + e_i - y_i), \tag{0.1}$$

met de α_i's de Lagrange vermenigvuldigers (duale variabelen). De oplossing resulteert dan in een linear stelsel van vergelijkingen [167]:

$$\left[\begin{array}{c|c} 0 & \mathbf{1}^T \\ \hline \mathbf{1} & K + \gamma^{-1} I_n \end{array}\right] \left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{y} \end{array}\right], \tag{0.2}$$

waarbij $\mathbf{1}$ een kolomvector van enen is, en $\mathbf{y}$ een vector met alle output waarden. De elementen van de symmetrisch positief-definiete kernel matrix K zijn $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Typisch is er een modelselectie procedure vereist om de twee hyperparameters (γ, h) te schatten, in geval van een Gaussiaanse kernel. Wanneer deze waarden zijn gekend, kan de LS-SVM approximator geëvalueerd

worden in elk punt $\mathbf{x}$ door

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T\varphi(\mathbf{x}) + b = \sum_{i=1}^{n} \alpha_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b.$$

Als alternatieve benadering van de regularizatie beschouwen wij nu hetzelfde lineair basismodel in de RKHS, maar nu met de restrictie van de regressieparameters tot een deelruimte met dimensie $s \ll n$, met als kolomruimte range$(V) = \{\mathbf{w}^{(d)} | \mathbf{w}^{(d)} = V\boldsymbol{\alpha}^{(s)}$ voor alle $\boldsymbol{\alpha}^{(s)}\}$, zodat we een met (0.2) vergelijkbaar stelsel van vergelijkingen verkrijgen:

$$\begin{bmatrix} 1 & (\Phi V\mathbf{1})^T \\ \Phi V\mathbf{1} & V^T\Phi^T\Phi V + \frac{1}{\gamma}V^TV \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T\mathbf{y} \\ V^T\Phi^T\mathbf{y} \end{bmatrix}, \tag{0.3}$$

met Φ de matrix met de kenmerken in de kolommen. De oplossing geeft de schatter van de vorm

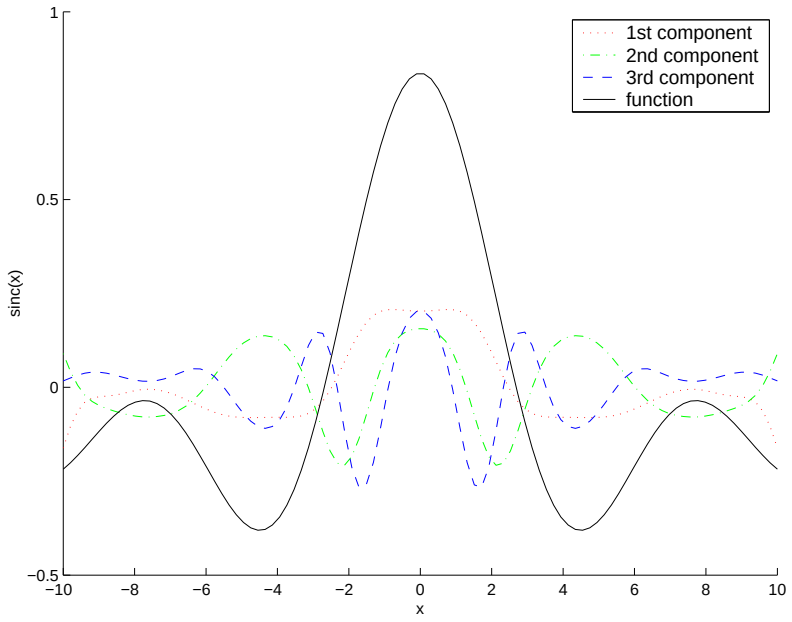
$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{n}(\sum_{j=1}^{s} a_{ij}w_j)k(\mathbf{x_i}, \mathbf{x}) = \sum_{i=1}^{n} \alpha_i \; k(\mathbf{x_i}, \mathbf{x}), \tag{0.4}$$

met a_{ij} de elementen van een $n \times s$ matrix A die de coefficiënten voorstellen waarmee een lineaire combinatie van de regressoren wordt samengesteld. We kunnen nu verschillende keuzes gaan maken voor de keuze van de matrix A, die overigens wordt bepaald in de duale ruimte, terwijl de regressiecoefficienten in de primale ruimte bepaald worden.

Enkele voor de hand liggende criteria om nuttige projectierichtingen te vinden zijn de Principale Componenten (minimalizatie van de interne correlatie tussen de variabelen), de Canonieke Componenten (minimalizatie van de externe correlaties) en richtingen die gebaseerd zijn op een combinatie van deze componenten. We tonen aan dat de uitwerking van deze gevallen, vertrekkend van een kleinste kwadraten benadering met restrictie, leidt tot regressie gebaseerd op kernel Principale Componenten Analyse (KPCA), Kernel Canonieke Correlatie Analyse (KCCA) en Kernel Partial Least Squares (KPLS). Figuur 0.3 toont hoe deze covariantiemethodes nuttige componenten voor de regressie opleveren.

In dit raamwerk gaan we dieper in op het KPLS geval en motiveren het, in contrast tot het louter gebruik van de kernel truuk, met een primair-duale formulering en afleiding [84], analoog aan de LS-SVM aanpak. Verder geven we ook de kernel versies van enkele PLS varianten uit de literatuur en voerden we vergelijkende experimenten uit op enkele gekende datasets, met algemeen goede resultaten voor de regressies en de conclusie dat inflatie gebaseerde PLS het minst aantal componenten benodigt om dezelfde regressieresultaten te bereiken [83].

Dat regressie met restrictie flexibele modellen levert tonen we eveneens aan op enkele voorbeelden uit de chemische industrie (BASF project [48]) en de energiesector (ELIA project [47]), waar ze goede performantie behalen.



Figuur 0.3: Visualisatie van de eerste drie componenten berekend met KCCA voor een sinc functie zonder ruis. The eerste kenmerkcomponenten bevatten duidelijk de karakteristieken van de onderliggende functie.

Ten slotte tonen we aan hoe men met de KCCA deelruimte de niet-lineaire identificatie van Hammerstein-Wiener systemen kan bekomen [71]. De niet-lineariteit van KCCA, een goede keuze van de kernel functie, en de combinatie met de LS-SVM maakt het mogelijk om het toestandsruimte en de niet-lineariteiten te schatten. We illustreren de methode op enkele niet-triviale voorbeelden.

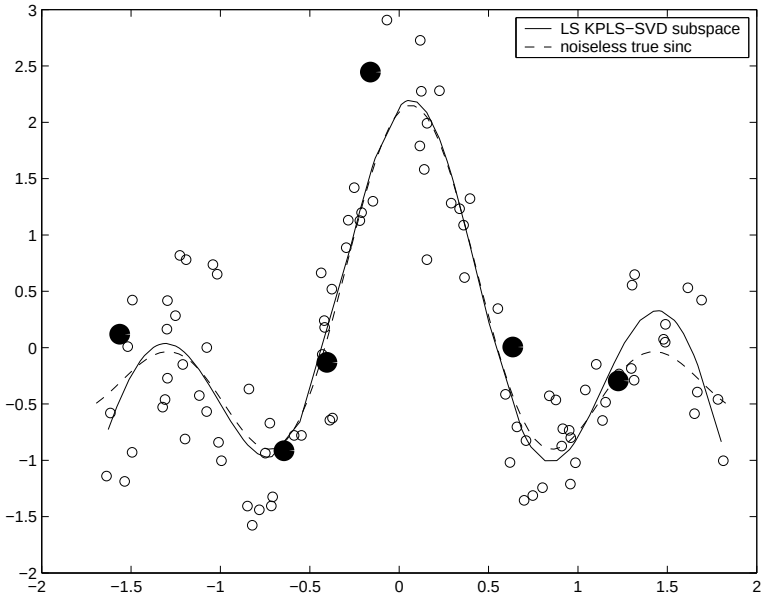
4. Gereduceerde Set Methodes

In Hoofdstuk 4 breiden we het kleinste kwadraten model, gebaseerd op restrictie tot een deelruimte, verder uit. Naast het voordeel dat het aantal parameters $s \ll n$ bij de *training* van deze machines beperkt kan worden, voegen we controle toe over het aantal termen $m \ll n$ dat het functievoorschrift (0.4) telt, nl. spaarsheid. Dit heeft het bijkomende voordeel dat de *evaluatie* van het model ook snel kan gebeuren. Vooral in het geval waarin de gegeven dataset zeer groot is, en de gangbare methodes een te lange rekentijd (bvb. polynomiaal in het aantal datapunten n) nodig hebben, is dit een zeer gewenste oplossing.

Op zeer natuurlijke wijze kunnen we aantonen dat de uitbreiding neerkomt op het werken met een gereduceerde subset van m basisvectoren waarin de s regressievectoren geprojecteerd worden in de RKHS. Door de deelruimte decompositie $V = \Phi_m^T A$ met Φ_m een kenmerken matrix in de m input vectoren bekomen we een bijkomende vereenvoudiging in het stelsel (0.3), met $\Phi V = \Phi\Phi_m^T A = K_{nm}A$, wat resulteert in een kernel regressiemodel in slechts m termen [87]:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \left(\sum_{j=1}^{s} a_{ij}\alpha_j^{(s)} \right) k(\mathbf{x}_i, \mathbf{x}) + b \equiv \sum_{i=1}^{m} \beta_i \; k(\mathbf{x}_i, \mathbf{x}) + b.$$

Een visuele voorstelling van dit model gebaseerd op slechts een klein aantal support vectoren is getoond in Figuur 0.4.



Figuur 0.4: Visualizatie van het LS KPCA subruimte regressiemodel, dat leidt tot een spaarse representatie, op een artificieel sinc data set voorbeeld. De subruimte wordt hier slechts opgespannen door 6 punten, gemarkeerd met een volle stip.

Aan de basis van dit veralgemeend regressiemodel vinden we een diepe connectie met het impliciet aanwenden van de Nyström benadering voor de kenmerk vectoren en de Fixed-Size LS-SVM [165]. In deze zin veralgemenen we het in Hoofdstuk 3 opgezette regressiemodel, maar door verschillende keuzes te maken voor V en A kunnen we bestaande methodes uit de literatuur plaatsen

binnen een algemeen kader. Bvb. het gebruik van $V = \Phi^T$ in het model met radiale basis kernels geeft de RBF-netwerk modellen [138]. Ook kernel Ridge Regressie modellen [150], met verbanden in Regularizatienetwerken [68, 56] en Gaussiaanse Processen [129]. Wanneer precies de kernel spaarsheid wordt beoogd, kiest men een gereduceerd set van basisvectoren $V = \Phi_m^T$ en dan bekomen we opnieuw een speciaal geval van bestaande modellen [160, 143]. Met bvb. $V = P_m\Phi^T = \Phi_m^T(\Phi_m\Phi_m^T)^{-1}\Phi_m\Phi^T$ bekomt men de bekende Nyström lage-rang benadering voor de kernel matrix K: $\tilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$.

Het voorgestelde regressiemodel laat toe om ook kernel Partial Least Squares en kernel Canonical Correlation analysis voor regressie nu met een spaarse representatie uit te drukken, zodat ze ook aangewend kunnen worden in de context van grote datasets, met weinig verlies aan accuraatheid [86]. Verder werden deze regressiemethodes ook geïllustreerd en empirisch vergeleken op enkele benchmark data sets met uitstekende resultaten [54].

5. Het introduceren van spaarsheid

In kernel gebaseerde classificatie en functie schatting is de spaarsheid (d.w.z. een beperkt aantal van kernel termen, in vergelijking met het totaal aantal beschikbare trainingspunten) van de machine een belangrijk probleem omdat het een snellere evaluatie van nieuwe datapunten toelaat. De overblijvende punten worden meestal aangeduid als support vectoren.

In de standaard SVM [184] is de spaarsheid ingebouwd dankzij de ϵ-insensitieve kostfunctie die fouten tot nul afvlakt voor punten binnenin een 'tube' rond de geschatte kromme. Dit resulteert in een kwadratisch programmeringsprobleem. In LS-SVM [167] is een kwadratische kostfunctie gebruikt en is het optimalizatieprobleem herleid tot het oplossen van een stelsel van lineaire vergelijkingen. Daarom dient de LS-SVM gecombineerd te worden met een apart criterium dat de spaarsheid introduceert.

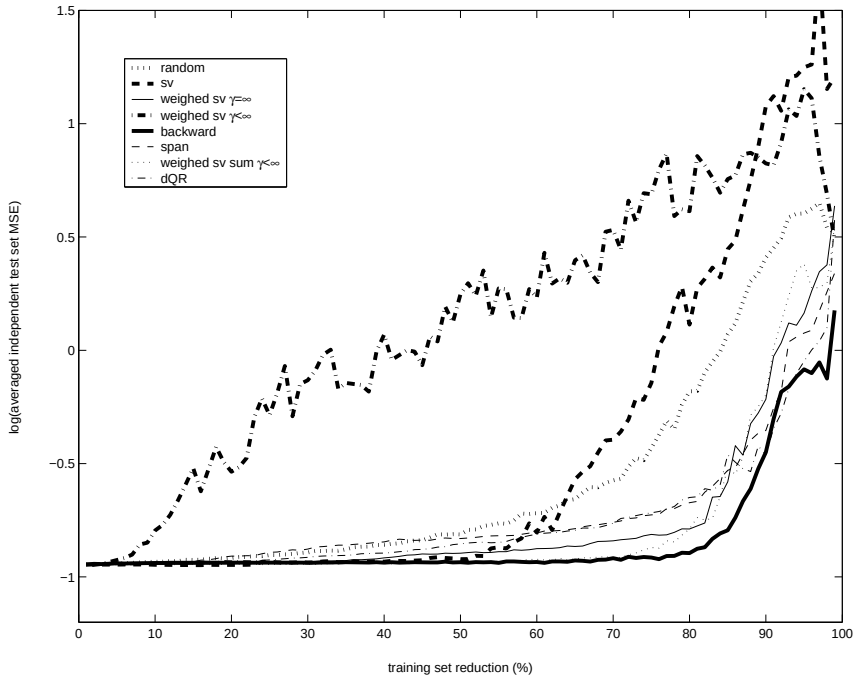
Een simpele aanpak om spaarsheid te introduceren is gebaseerd op het gesorteerde spectrum van de support vector waarden (d.w.z. de gewichten van de oplossing van het stelsel) [163]. Vanuit de LS-SVM oplossing volgt een redelijke keuze voor het wegsnoeien van datapunten met een lage foutenbijdrage in de kostenfunctie, men gaat in de standaard aanpak punten met een minimale coefficient weglaten (methode 1). Een recente publicatie [39] sofisticeert dit snoeialgoritme door het wegen van de support vector waarden. Het datapunt met de met de kleinste fout, geïntroduceerd na de eliminatie ervan, wordt geselecteerd om weggelaten te worden. Er wordt beweerd dat deze methode het standaard schema [163] zou overtreffen, maar het bereik van de vergelijking was slechts beperkt tot een voorbeeld waar de ruis op de data weggefilterd was. We suggereren hier een andere *verbeterde* selectie van het te snoeien punt

gebaseerd op het door de auteurs afgeleide criterium (methode 2).

Snoeien is nauw gerelateerd aan subset selectie, het kiezen van relevante datapunten of variabelen om het spaarse model mee op te bouwen. Snoeien veronderstelt dat het model, te beginnen vanop de volledige training data set, op iteratieve wijze wordt gereduceerd, in een achterwaartse manier, terwijl subset selectie meestal in een voorwaartse wijze gebeurt. Vele subset selectieschema's organizeren hun zoek op een exhaustieve wijze [160]. We zullen op twee van zulke algoritmes ingaan. Aan de ene kant een (supervised) methode gebaseerd op de QR decompositie [121, 27], wat we kunnen gebruiken om punten weg te laten waarvan de georthogonalizeerde componenten het minst correlatie vertonen met de uitgang (methode 3). En aan de andere kant een (unsupervised) methode gebaseerd op de best beaderende deelruimte [13, 24], wat we kunnen gebruiken om punten weg te laten die het minste similariteit vertonen met de deelruimte (methode 4).

In de experimenten met snoeien beogen we een vergelijking van de regressieperformantie tussen de twee snoeimethodes en de twee subset selectieprocedures door een reeks van testen uit te voeren met (i) evaluatie op de onafhankelijke test set (als geijkte basis om te vergelijken) en (ii) het beschouwen van random en achterwaarts snoeien om respectievelijk een ondergrens voor de performantie en een objectieve maat te hebben. In Figuur 0.5 tonen we hoe de performantie zich typisch gedraagt wanneer punten gesnoeid worden met verschillende selectiecriteria.

Als uitbreiding op snoeien beschouwen we verder de LS-SVM in een *on-line context*, waar we nu ook het *aangroeien* in rekening brengen. We beschouwen de LS-SVM in termen van een relatief klein, vast set van basisvectoren, in een on-line lus waar we iteratieve updating (inclusie van een nieuw punt) wordt afgewisseld met downdating (exclusie) naargelang een selectiecriterium op de punten in de gereduceerde set. In het bijzonder verzamelen we drie updating methodes (extensie, approximatie en recursie) en drie downdating methodes (afkapping, approximatie, recursie) in een gestructureerd schema en tonen aan hoe dit laatste op natuurlijke wijze selectiecriteria induceren gebaseerd op foutmaten in zowel data- als kenmerkenruimte. Door het combineren van de verschillende componenten met een van de bepaalde selectie mechanismen, kan men iteratieve, on-line LS-SVM algoritmes construeren, die een efficiente representatie kunnen bereiken, gebaseerd op een geselecteerde fractie van overblijvende support vectoren. Verscheidene strategieën worden vergeleken op enkele benchmark data sets.



Figuur 0.5: Gemiddelde kwadratische fout op een onafhankelijke test set versus de reductie van de grootte van de training set (in %) voor een 1-dimensionele sinc functie, met ruisniveau $\sigma^2 = 0.2$.

6. Updating en Tracking van de Dominante Kernel Eigenruimte

Een hele reeks van methodes (bvb. Kernel Principale Componenten Analyse (KPCA) [155], Fixed-Size Least Squares Support Vector Machines (FS-LSSVM) [165], de-noising [153, 120, 97, 147], data modellering [174], etc.) steunen, direct of indirect, op de dominante m eigenvectoren van de kernel matrix K, waar $m \ll n$. Het gebruikelijke middel om deze eigenruimte te berekenen is de Singuliere Waarden Ontbinding (SVD) omwille van de grote numerieke nauwkeurigheid [73]. Het nadeel van de standaard routines voor de batch berekening van een SVD is dat ze $O(n^3)$ operaties vereisen (of $O(mn^2)$, bvb. in deflatie schema's [73]).

Het is duidelijk dat voor de toepassing op grote datasets (zoals bvb. in beeldverwerking of patroonherkenning) de kernel methodes, met hun sterk vermogen om met niet-lineariteiten om te gaan, computationeel gezien erg beperkt zijn. Voor grote datasets neemt een eigenwaardenontbinding van K eenvoudigweg teveel tijd in beslag. Maar ook geheugen is een probleem

aangezien typisch $O(n^2)$ geheugen-eenheden vereist zijn. Bijkomend moet soms de berekening vele malen herhaald worden als het een niet-stationair probleem betreft. En sommige toepassingen vereisen exclusief adaptieve algoritmes. Een echt on-line algoritme voor de extractie van kernel principale componenten werd nog niet gerapporteerd tot dusver, hoewel het probleem belangrijk is en pogingen om het probleem op te lossen momenteel ondernomen worden [146, 101, 104].

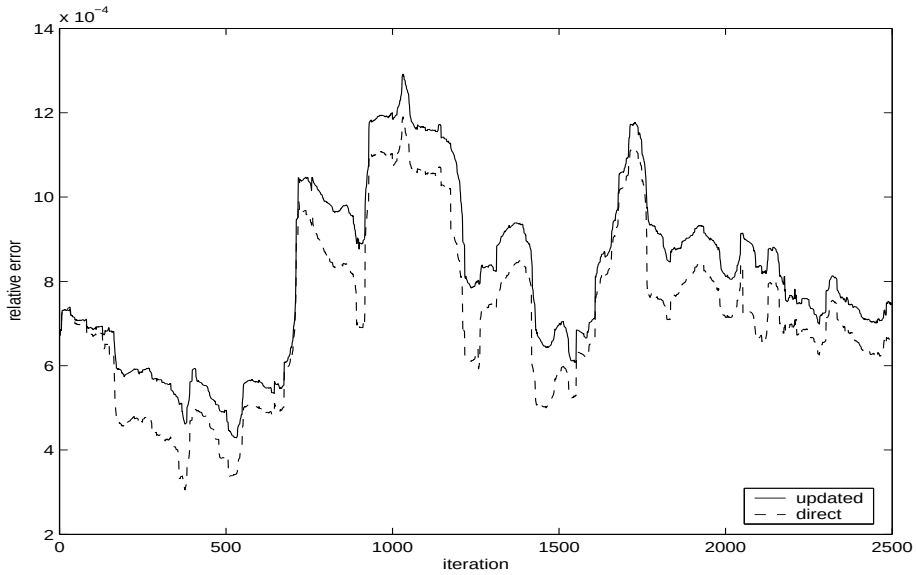
De literatuur van de laatste drie decennia toont dat er aanzienlijke moeite en aandacht besteed is aan deze problemen, met een gemeenschappelijke noemer genaamd SVD updating [20, 19, 127, 25, 108, 7], waarbij iteratief de vorige eigenruimte wordt herbruikt om de nieuwe te berekenen als er een nieuwe data observatie binnenkomt. In de meeste van deze methodes gebeurt het updaten incrementeel met het toevoegen van een of meerdere rijen (of kolommen) aan de betreffende matrix.

Nochthans, in de bijzondere context van kernel algoritmes, waar de matrix dimensies overeenkomen met het aantal trainingsdatapunten, kunnen de updating schema's niet rechttoe-rechtaan toegepast worden. De kernel matrix expandeert namelijk in *zowel* rij- als kolomdimensie wanneer een punt toegevoegd wordt. Voor zover onze huidige kennis reikt, hebben we de behandeling van dit geval nog niet kunnen terugvinden.

In Hoofdstuk 6 presenteren we een snel on-line updating en downdating procedure voor het berekenen van de dominante eigenruimte in geval van een *groeiende* symmetrische vierkante matrix, met een rekencomplexiteit van $O(nm^2)$ en geheugen vereisten van $O(nm)$ per iteratie, en met zeer beperkt verlies van nauwkeurigheid [82]. We ontwikkelden een numeriek algoritme om de eigenruimte te berekenen op basis van een gegeven beste-rang schatter en opeenvolgende minimale SVD en QR decomposities. Voor het geval van deelruimte tracking, waar men de dimensie van de eigenruimte constant houdt, stellen we ook een efficiënt downsizing mechanisme voor. Experimenten tonen dat ons algoritme een numeriek stabiele en nauwkeurige benadering oplevert voor eigenwaarden en eigenvectoren, met een excellente overeenkomst met de met batch algoritmes berekende resultaten [81]. In Figuur 0.6 tonen we een typisch voorbeeld van een accurate tracking van een eigenruimte met onze methode t.o.v. de batch berekening.

7. Veralgemeende Hogere-Orde Rayleigh Quotient Iteratie

In Hoofdstuk 7 begeven we ons op het veld van de multilineare algebra, dat betrekking heeft op hogere-orde tensoren. In contrast tot matrices, kunnen tensoren gezien worden als entiteiten waarvan de elementen aangeduid woren met meer dan twee indices. In de literatuur worden deze hogere-orde



Figuur 0.6: Relatieve fout versus iteratiestap, getoond voor de tracking van een m-dimensionele eigenruimte en de rang-m eigenruimte bekomen door een batch SVD, voor de getrackte 500×20 kernel matrix van de Abalone data set. The twee geplotte curves volgen elkaar nauw, wat een indicatie is van een goede benadering van de eigenruimte.

equivalenten van vectoren (eerste orde) en matrices (tweede orde) hogere-orde tensoren, multidimensionele arrays of multiway arrays genoemd.

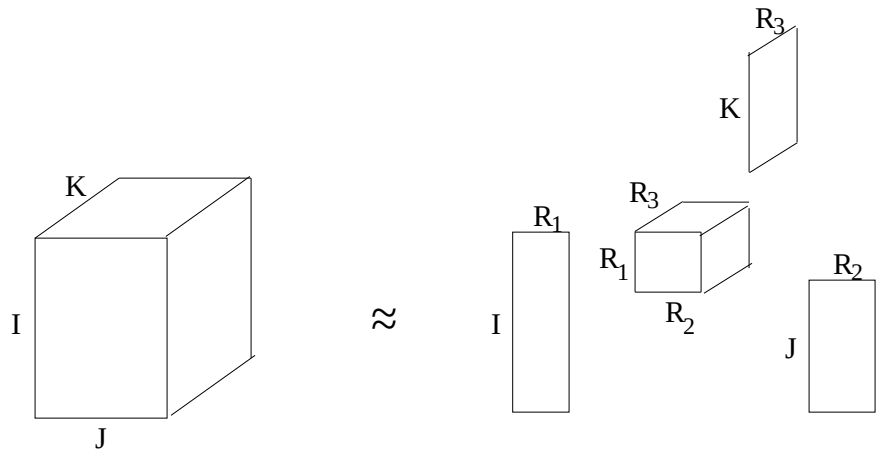
Voor heel wat toepassingen in wetenschappen en ingenieurstechnieken blijkt het bestaande raamwerk van vectoren en matrices onvoldoende of onaangepast. Er is dikwijls een nood aan meerdere dimensies en een representatie met een hogere dichtheid. In deze zin, breiden (of veralgemenen) tensoren de klassieke matrix algebra uit. Bijgevolg brengt de uitbreiding ook een nood aan de transfer van de bestaande resultaten en algoritmes met zich mee. Deze uitdaging en vaak niet-triviale taak vormt de focus van lopend onderzoek in het domein van de multilineaire algebra.

Een belangrijk voorbeeld van een typisch onderwerp is dimensionaliteitsreductie, waar het het doel is om invariante subruimtes voor systemen af te leiden, die beschreven worden door vaak hoog-dimensionele matrices. Met betrekking tot de numerieke aspecten van Principale Componenten Analyse (PCA) wordt wereldwijd wetenschappelijk onderzoek ondernomen. Dit heeft geleid tot krachtige routines (direct of iteratief) voor de berekening van de Eigenwaarde Ontbinding (EVD), Singuliere Waarden Ontbinding (SVD), dominante subruimtes, etc. van hoog-dimensionele matrices.

In dit hoofdstuk pogen we bij te dragen tot dit onderzoeksveld. Op het gebied van dimensionaliteitsreductie focussen we op de Rayleigh quotient iteratie om de eigenvector van een matrix te bekomen. We bouwen verder op twee publicaties in de literatuur die dit algoritme op verschillende wijze veralgemenen. Aan de ene kant stelden Absil *et al.* een algoritme voor dat de dominante eigen*ruimte* van een matrix levert [2]. Aan de andere kant stelden Zhang *et al.* een algoritme voor om de dominante eigenvector *van een tensor* te bekomen [200]. Wij hebben hun resultaten gecomplementeerd met het meest algemene geval: het berekenen van de dominante eigenruimte van een tensor.

In het bijzonder hebben we een Hogere-Orde Grassmann-Rayleigh Quotient Iteratie (HOGRQI) afgeleid voor de berekening van de beste-rang-(R_1, R_2, R_3) benadering van hogere-orde tensoren [41]. In Figuur 0.7 geven we een visuele voorstelling voor een derde-orde tensor.

De convergentie is kwadratisch en elke iteratiestap behoeft slechts het oplossen van een vierkant stelsel van lineaire vergelijkingen. Dit is een grote verbetering ten opzichte van de algoritmes gemeld in [43], waarbij de convergentie ten hoogste linear is en elke iteratie het partieel oplossen van een aantal Singuliere Waarde Ontbindingen met zich meebrengt. We leiden analytisch gesloten formules af, geven een Matlab implementatie voor het HOGRQI algorithm, en illustreren de toepassingsmogelijkheden in een Onafhankelijke Componenten Analyse (ICA) context.



Figuur 0.7: Visualizatie van de beste (R_1, R_2, R_3)-rang benadering van een $I \times J \times K$ tensor. We veralgemeenden de berekening van een eigenvector van een vierkante matrix d.m.v. de Rayleigh Quotient Iteratie tot de berekening van de gezamenlijk dominante k-mode deelruimte van een tensor.

8. Conclusies en verder onderzoek

In Hoofdstuk 8 geven we een overzicht van onze bijdragen, de conclusies en stippen we enkele suggesties aan voor verder onderzoek.

In grote lijn situeert dit onderzoek zich op het gebied van Gegevensontginning en Kennisextractie. Hierbinnen bakenen we het onderzoeksterrein verder af tot kernel gebaseerde algoritmes in het domein van Artificiële intelligentie en Lerende Systemen.

Kort samengevat, verrichtten wij onderzoek rond vijf thema's: (i) we leiden een primair-duale interpretatie af voor Kernel Partial Least Squares en reiken een unificerend interpretatiekader aan voor een verzameling van reeds bestaande kleinste kwadraten gebaseerde kernel regressie methodes in een Reproducerende Kernel Hilbert ruimte, (ii) we voorzien deze methodes van een spaarse formulering via de Nyström benadering, wat toelaat om kernel regressie op zeer grote data sets toe te passen met beperkt verlies van nauwkeurigheid, (iii) we verkennen mogelijke up/downdating strategieën voor Least Squares Support Vector Machines, met geïnduceerde data punt selectiecriteria, (iv) we ontwikkelen een nieuw tracking algoritme om de dominante eigenruimte van de kernel matrix efficient te benaderen (v) we leiden een veralgemeende Grassmann-Rayleigh Quotient Iteratie af voor de berekening van de beste rang-(R1,R2,R3) benadering van hogere-orde tensoren, met kwadratische convergentie.

Onderzoek eindigt nooit, en in deze zin is deze doctoraatsthesis slechts een momentopname van werk verricht in een beperkte periode. Er zijn zeker nog bijkomende vraagstukken en uitdagingen die verder onderzoek behoeven.

We noemen enkele open onderzoeksonderwerpen op. Het kiezen van de optimale basis voor de projectie van de kernel representanten. Het kiezen van de juiste kernel functie en zijn parameters. Alternatieve subsampling technieken. Een doorgedreven empirische studie van de on-line mogelijkheden met LS-SVM. Het aanwenden van kernel methodes in andere onderzoeksdomeinen. Het updaten en tracken van de eigenruimte behorende bij de *kleinste* eigenwaarden. Het mee in rekening brengen van de normalizatie in onze definitie van de Rayleigh Quotient. De berekening van een specifieke eigenruimte behorend bij een eigenwaarde, i.p.v. de dominante eigenruimte.

Contents

Voorwoord **i**

Abstract **iii**

Korte inhoud **v**

Nederlandse samenvatting **vii**

Contents **xxv**

1 Introduction **1**

1.1 Data Mining, knowledge extraction and machine learning 1

1.2 Motivation 2

1.3 Structure of the thesis 5

1.4 Contributions 8

2 The Kernel Paradigm in Machine Learning **11**

2.1 Learning from examples 11

2.2 Risk 13

2.2.1 Empirical Risk 14

2.2.2 Structural risk 16

2.2.3 Non-linearity 17

2.3 Reproducing Kernel Hilbert Space 20
2.3.1 Regularization and nonlinearity together embedded 20
2.3.2 The feature map 21
2.3.3 Representer theorem 23
2.4 Support Vector Machines 24
2.4.1 Linear classification of separable classes 24
2.4.2 (Non)linear classification of nonseparable classes 24
2.4.3 Linear function estimation of ε-accurate values 26
2.4.4 (Non)linear function estimation of non-ε-accurate values . 28
2.5 Model selection and assessment 29
2.6 Conclusions 32

3 Least Squares Kernel Machines 33
3.1 Least Squares Support Vector Machines 33
3.1.1 Model assumptions 34
3.1.2 Model solution 34
3.2 Subspace regression in reproducing kernel Hilbert space 36
3.3 Subspace construction 38
3.3.1 Minimization of within-space correlation 38
3.3.2 Maximization of between-space correlation 39
3.3.3 Intermediate criteria 40
3.4 Introducing the kernel function 42
3.4.1 Kernel Principal Component Analysis 42
3.4.2 Kernel Canonical Correlation Analysis 43
3.4.3 Kernel Partial Least Squares 44
3.4.4 KPLS derived in a primal-dual optimization perspective . 45
3.5 Illustrative examples 47

3.5.1 Sinc toy data set 47
3.5.2 Boston housing data set 48
3.5.3 Chemical data set 52
3.5.4 Electrical Power data set 52
3.6 An application in Subspace Identification 55
3.6.1 Introduction 55
3.6.2 The subspace intersection algorithm 56
3.6.3 Hammerstein-Wiener subspace intersection 58
3.6.4 Illustrative examples 61
3.7 Conclusions 67

4 Reduced Set Methods **69**

4.1 Subset Based Least Squares Subspace Regression in RKHS . . . 69
4.2 The Nyström approximation 71
4.2.1 Approximation of eigenfunctions 71
4.2.2 Feature approximation based on the complete training data set 72
4.2.3 Feature approximation based on a training data subset . . 73
4.3 The restricted subspace regression model 75
4.3.1 Least Squares regression 76
4.3.2 Restriction to a subspace 77
4.3.3 Choice of basis 78
4.3.4 A complementary view 79
4.3.5 Eigenvectors for the subspace 81
4.4 Experiments 81
4.4.1 Artificial data example 82
4.4.2 Real world data examples 83
4.4.3 Large scale example 87

4.5 A comparative benchmark study . . . 90
4.5.1 The 'Silver Box' Identification Problem . . . 90
4.5.2 Experimental Results . . . 92
4.6 Conclusions . . . 97

5 Introducing Sparsity 99

5.1 Pruning . . . 100
5.2 Pruning methods . . . 101
5.2.1 Minimal coefficients . . . 101
5.2.2 Weighed support values . . . 101
5.2.3 Orthogonal error minimizing regressors . . . 104
5.2.4 Approximating representers . . . 105
5.3 Experiments with Pruning . . . 106
5.4 Selection criteria for online LS-SVM's . . . 110
5.5 Updating . . . 111
5.5.1 Extension . . . 111
5.5.2 Approximation . . . 112
5.5.3 Recursive . . . 112
5.5.4 Update criteria . . . 113
5.6 Downdating . . . 115
5.6.1 Downdate criteria . . . 115
5.6.2 Truncation . . . 117
5.6.3 Approximation . . . 119
5.6.4 Recursive . . . 123
5.7 Experiments with online LS-SVM . . . 126
5.7.1 Algorithm . . . 126
5.7.2 Comparison . . . 127

5.8 Conclusions 133

6 Updating and Tracking the Dominant Kernel Eigenspace 135

6.1 Introduction 135

6.2 Updating and downdating 137

6.2.1 Updating 137

6.2.2 Downdating 139

6.3 Downsizing 140

6.4 Experiments 143

6.4.1 Eigenspace Tracking 143

6.4.2 Eigenspace Updating 149

6.5 Conclusions 152

7 Generalized Higher-Order Rayleigh Quotient Iteration 153

7.1 Introduction 154

7.1.1 The eigenvector of a matrix 155

7.1.2 The eigenspace of a matrix 156

7.1.3 Notation and definitions 157

7.2 The dominant k-mode vector of a tensor 159

7.2.1 The generalized Rayleigh quotient 159

7.2.2 The Newton iteration 161

7.3 The dominant k-mode space of a tensor 163

7.3.1 Higher-Order Grassmann-Rayleigh Quotient Iteration . . 164

7.3.2 The Generalized Rayleigh Quotient 165

7.3.3 A central property of the Rayleigh quotient 165

7.3.4 The Newton iteration 166

7.3.5 Convergence rate 168

7.4 Computing the set of linear equations 169

7.4.1 Calculus rules and definitions 169
7.4.2 Explicit expression of the linear equations 171
7.5 Example 174
7.6 Conclusions 177

8 Conclusions **179**
8.1 Summary 179
8.2 Further research 182

Chapter 1

Introduction

This Chapter is a general introduction that (i) situates this work in a broad context, (ii) gives the main motivations in an intuitive language, (iii) summarizes the structure of the thesis and (iv) gives an overview on the specific contributions.

1.1 Data Mining, knowledge extraction and machine learning

The last decades may well be regarded as an information-age. We all have experienced the revolution in availability and exchange of information. Several causes can be discerned that drive the rapid developments: (i) world-wide integration of the microprocessor, (ii) increase of processor power (cfr. Law of Moore), (iii) exponential growth rate of the internet usage (saturation has not yet been reached), (iv) integration of information technology in all sectors, (v) globalisation of economy with organisations growing on world-scale, (vi) reduced costs of data acquisition, cheap and reliable measurement equipment becomes available due to mass-production, (vii) reduced costs of data storage, cheap memory media become available with high capacities.

It is unmistakably clear that new challenges arise for both business and scientific users in the processing of this huge amount of data [59]. There is a serious gap between information and knowledge and many opportunities lie there for progress and improvement. The main motivation is to explore these possibilities to harvest the benefits, in the realm of positivism, with the argument that better knowledge allows for better decision-taking.

Research and development in extracting the knowledge from data in the context

of large databases, is commonly referred to as *data mining* and *knowledge discovery*. These fields have experienced as well a tremendous growth, since the beginning of the 1990s. Also in the industry these data-driven branches become more and more a necessary component in the decision support architecture where they complement other (simpler) techniques like queries, reporting, online analytical processing (OLAP), business objects, data visualization and traditional statistical analysis.

The target in data mining is also to extract useful information from large scale data in a quick, reliable and transparant manner. By the application of methods and particular algorithms one tries to derive relations between variables or to identify functions that allow to predict, classify or represent regularities in the data. Typically knowledge extraction delivers automatically descriptive and predictive hypotheses to the user, whereas in classical approaches the user has to query the data with his own hypotheses. In Figure 1.1 we depict this process.

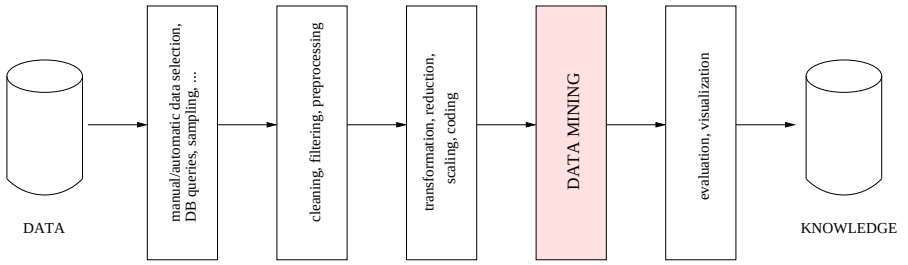


Figure 1.1: A conceptual picture of the process steps involved in knowledge extraction.

The explosive growth of databases in companies, governments and organisations are exceeding the ability to process and interprete the data. Techniques in data mining typically draw upon various methods from *artificial intelligence, machine learning and statistics*, and efforts are required to leverage and extend the existing methods which are better capable of discovering complex nonlinear relations in large amounts of data. Consequently, many universities are starting up new research laboratories to create a response to this need. In this track, this work is a product of such a research program that aims at engineering methods to learn from data.

1.2 Motivation

This work is centered around automatic learning from a given collection of pairwise examples (input-output samples). Clear examples are classification and

function estimation. The history of this goal is long-standing and a multitude of methods could be chosen as the focus of attention in order to investigate. But it are of course the latest developed topics and (still) unresolved issues that drive the subject choice. Nowadays the landscape of artificial intelligence and machine learning is heavily influenced by the class of models known as *kernel methods*. In this work we mainly focus on kernel based methods.

Kernel methods are a new generation of techniques that are founded on sound theoretical grounds. The first class of methods that implemented the theory were Support Vector Machines (SVM). They are general approximators (depending on the kernel) for the (nonlinear) relation that underlies the given sample pairs. Their successful performance and elegance have been drawing the attention of many researchers. Among other paradigms (like neural networks, Gaussian Processes) they became a worthy adverse with strong capabilities. They are e.g. conceptually simple and very transparant, they possess a unique and global solution and they are less sensitive to high input dimensionality. But most of all they created a new playground with room for extensions and improvements. This work is an attempt to contribute in that matter.

The intuitive idea behind kernel methods is that the data are nonlinearly mapped into a higher dimensional feature space, related to a nonlinear kernel function (see Figure 1.2). In general such a procedure has the advantage that inseparable data becomes linearly separable. But the disadvantage is that computations with high-dimensional mapped data points (i.e. very long column vectors with millions of entries) come at too large computational costs -except for *dot products* of the kernel map, since these in particular coincide with a mere kernel function evaluation. Consequently, only dot product based algorithms can be kernelized or used in the feature space, but it is mostly sufficient: for a very large class of algorithms the solution in the original data space can finally be expressed in terms of the solution in the feature space. In Chapter 2 these intuitive notions are more formally explained.

In the other chapters we have dealt with the following limitations and questions around which we investigated and tried to make a contribution to their solution:

- Presently many methods are called kernel machines once they are endowed with a kernel based dot product: in that way they are limited to the ad hoc use of the so-called kernel trick. But it does not imply that they are motivated according to the principled primal-dual optimization approach of SVMs. Can we enhance some of these methods in a more appropiate manner, with this conceptual interpretation?

- Methods become kernelized and start to live their own life. It is however useful to take a broad viewpoint and notice that some least squares kernel methods can be interpreted as instances of a general algorithm. Can we provide such a general framework for some kernel methods?

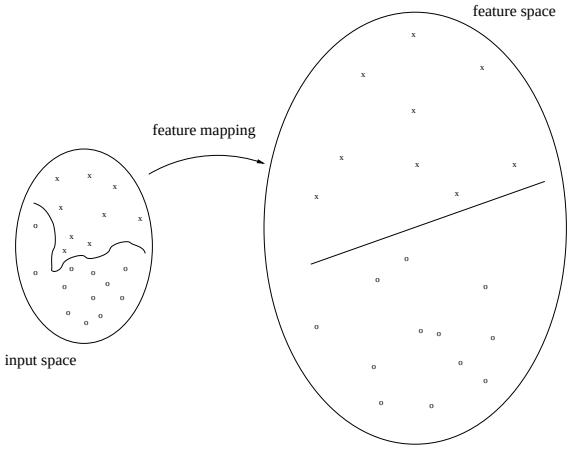


Figure 1.2: Mapping of the input space to a high-dimensional feature space where a linear separation is made, which corresponds to a nonlinear separation in the original input space.

- The concept of Support Vector Machines (SVM) to introduce nonlinearity through the use of kernels has a very broad applicability, and its proliferation into other domains is often fruitful, but still too limited. Can we employ kernels in e.g. the related field of subspace identification and enhance one of its classical methods?

- The final solution of SVMs is only dependent on a subset of the training data, it is sparse. This property is desirable because it makes the evaluation of the SVM at new data highly performant. Can we introduce sparsity in some kernel regression methods?

- It is a limitation that the Least Squares Support Vector Machine (LS-SVM) solution depends on all data points (it is not sparse). Can we improve on existing methods to introduce sparsity in LS-SVM? Can we derive other selection criteria for pruning LS-SVMs? Can we derive useful selection criteria for growing LS-SVMs?

- Many important methods rely upon the principal components of the kernel matrix. But for large data sets the batch computation is severely limiting their application. Can we efficiently compute the eigenspace of the kernel matrix in a faster recursive/incremental manner?

- The Rayleigh Quotient Iteration (RQI) algorithm for obtaining an eigenvector of a matrix has been long known and is appreciated for its favorable convergence properties. Generalizations exist both to compute

the eigen*space* of a matrix and to compute the eigenvector of a *tensor*. Can we, via Rayleigh Quotient Iteration, compute the eigenspace of tensors?

1.3 Structure of the thesis

This thesis is subdivided in six main chapters with each an introduction, a body and conclusions. All chapters have been written in such a way that they are as self-contained as possible. The symbols and notations have been kept intuitive and have been clarified when not self-explaining. Chapters 3-4-5-6 share the common background exposed in Chapter 2, while Chapter 7 is more independent of it. In general, each chapter can be seen as making a logical extension of the previous chapter, such that a causal chain is formed. This structure is depicted in Figure 1.3.

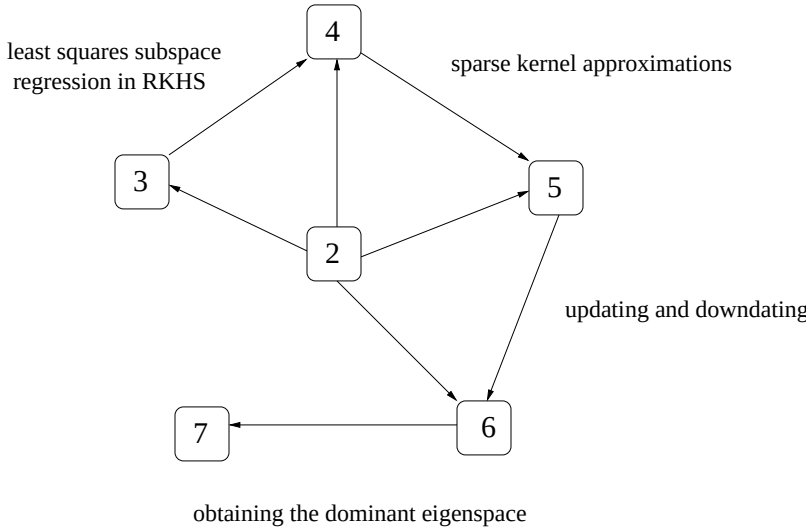


Figure 1.3: Structural overview of the thesis. The arrows indicate dependencies between the chapters. The titles denote the common theme that links the chapters together, so that they form a logical chain.

The contents of the chapters are organized as follows.

- Chapter 1 gives an introduction and describes the background of this manuscript in general. In condensed terms, the main motivations are

sketched and an overview is given of the content and the concrete contributions.

- In Chapter 2 we zoom in on the basic concepts of machine learning, setting the scene for the next chapters with an overview of the paradigm in which kernel models are rooted. We follow here mainly the statistical learning point of view, the cornerstone of Vapnik's Support Vector Machines.

  In Section 2.1 we decompose the elements of learning by examples. In Section 2.2 the performance measure is highlighted, where the empirical risk is recognized as the inference principle, that has to be extended to structural risk in order to reach feasibility of the solution. Another aspect of performance is the capability of modeling nonlinearities, that by means of Support Vector Machines historically led to the introduction of kernels. In Section 2.3 these two aspects are found to be naturally unified in the particular structure of Reproducing Kernel Hilbert Spaces. In Section 2.4 we give an overview of the Support Vector Machines (SVM), that accelerated the proliferation of the kernel paradigm. In Section 2.5 we discuss approaches to determine the hyperparameters of kernel models through the concepts of model selection and assessment.

- In Chapter 3 we focus on the important class of methods of Least Squares Support Vector Machines (LS-SVM) which are based on a least squares cost function, with added regularization. Next to LS-SVM regression with an extra penalty on the norm of the parameter vector, we place the class of kernel regression with restriction of the subspace.

  In Section 3.1 our study sets off with the exemplary Least Squares Support Vector Machines (LS-SVM). In Section 3.2 we connect to this class of penalized kernel models by considering the decomposition of least squares regression models in feature space with regularization by projection into a subspace. In Section 3.3 it is shown to form a natural framework in which methods like Kernel Principal Component Analysis (KPCA), Kernel Canonical Correlation Analysis and Kernel Partial Least Squares (with variants) fit for a advantageous choice of subspace. In Section 3.4 we consider these algorithms in the feature space context and derive additionally a novel motivation of KPLS, from a primal-dual optimization perspective. In Section 3.5 we illustrate the models on some data sets and demonstrate some applications. In Section 3.6 we go into an application of KCCA in Subspace Identification for the nonlinear identification of Hammerstein-Wiener systems.

- In Chapter 4 we extend the least squares subspace regression framework of previous chapter by introducing an approximation of the subspaces computed on a reduced part of the training data set. This has the important advantage of obtaining a sparse kernel expansion and allowing for handling large data sets.

In Section 4.2 we deal with the Nyström approximation for features to recognize three least squares models. In Section 4.3 we introduce least squares regression in feature space, with parameters restricted to a subspace and a choice of basis vector subset for projection. In Section 4.3.5 we give an overview of some alternative eigenspaces with which we can endow the model. In Section 4.4 we illustrate the different methods by some results on an artificial and real-world data sets, including a large scale example. In Section 4.5 we demonstrate the application of our class of models in a comparative study on the Silver Box benchmark data.

- In Chapter 5 we explore possible pathways to introduce also, like in previous chapter for subspace kernel regression, sparsity in the LS-SVM model. On the one hand we perform a study of pruning methods and on the other hand we analyze error measures and induced selection criteria to assess updating and downdating of the LS-SVM in an online setting.

  In Section 5.2 we explain the basic principles of four pruning capable methods and propose an alternative selection criterion for the method of de Kruif [39]. In Section 5.3 we discuss the results of experiments on a series of UCI benchmark data sets. In Section 5.4 we generalize pruning and motivate the use of LS-SVM in an online setting. In Section 5.5 different approaches of updating are considered, with corresponding errors in data and feature space. In Section 5.6 we consider downdate errors in data and feature space, different approaches of downdating and the induced selection criteria. In Section 5.7 we illustrate and compare some useful combinations of the methods.

- In Chapter 6 we elaborate on an updating and downdating procedure for the efficient computation of the dominant principal components of the kernel matrix, that are of central importance in many kernel methods.

  In section 6.2 we propose an updating and downdating scheme for an existing eigenspace. In section 6.3 we develop an efficient way to downsize the row dimension of an updated eigenspace and we outline a high-level algorithm for eigenspace tracking. In section 6.4 we present experiments on some typical benchmark data sets to demonstrate that accuracy and stability of the eigenvalues and eigenvectors are preserved during and after iteration.

- In Chapter 7 we connect to the previous chapter with the computation of the dominant eigenspace of a generalization of matrices, namely higher-order tensors.

  In Section 7.1 we set the scene, consider some iterative methods for refining an eigenvector of a given matrix, briefly draw some concepts of Absil [2] to generalize the Rayleigh Quotient Iteration for obtaining eigenspaces of matrices and settle notation and prepare definitions in order to introduce higher-order concepts. In Section 7.2 we summarize the approach of Zhang [200] in the generalization of the Rayleigh Quotient Iteration for

obtaining the dominant k-mode eigenvector of tensors. In Section 7.3 we discuss the line of reasoning to attain a fruitful definition of the Rayleigh Quotient in the context of obtaining the dominant eigenspace of a tensor, we generalize the Rayleigh Quotient, prove a crucial property of the RQ, develop the formal equations for a Rayleigh(-Newton) iteration, formulate the high-level algorithm and demonstrate the quadratic convergence rate. In Section 7.4 we develop explicit expressions for the set of equations in our RQI algorithm, after reviewing some calculus rules and extra definitions. In Section 7.5 we show an example and mention the relevance of our GRQI method in Independent Component Analysis.

- In Chapter 8 we summarize the results obtained in this thesis and give an outlook on additional research questions.

1.4 Contributions

In this section we summarize the main contributions obtained in this thesis.

- *A Primal-dual optimization formulation of Kernel Partial Least Squares.* Taking the LS-SVM as the prototypical example of a penalized modelling paradigm based on primal-dual representations, we originally endowed the Kernel Partial Least Squares (KPLS) technique with a primal-dual motivation. We also showed how this method, together with other covariance based criteria, like Kernel Principal Component Analysis (KPCA) and Kernel Canonical Correlation Analysis (KCCA), fit naturally in a general framework of kernel regression that penalizes by reduction of projections into a particular subspace. Further, we kernelized and compared different KPLS variants and applied KCCA in a nonlinear Subspace Identification context.
  - [83] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Kernel PLS variants for regression. In *Proceedings of the 11th European Symposium on Artificial Neural Networks (ESANN2003)*, pp. 203–208, Bruges, Belgium, April 2003.
  - [84] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subspace regression in Reproducing Kernel Hilbert Space. In *Proceedings of the 13th System Identification Symposium (SYSID2003)*, pp. 839–842, Rotterdam, Nederland, August 2003.
  - [71] I. Goethals, L. Hoegaerts, V. Verdult, J.A.K. Suykens, and B. De Moor. Hammerstein-Wiener subspace identification using kernel Canonical Correlation. *Submitted for publication*, 2005.
- *Subset based Least squares Subspace Regression in Reproducing Kernel Hilbert Space.* We proposed an extension of the restricted least squares subspace regression in a RKHS. In addition to the reduction to a certain subspace we show now how reduction to a subset of the basis

of representers can offer a *controlled sparse representation* for the kernel expansion of least squares kernel regression models. This extension is based on the Nyström approximation for features and permits to perform kernel regression on very large data sets with very limited loss of accuracy. These models have been demonstrated successfully on a series of real-world data sets and obtained also one of the best test set results on a benchmark problem.

  - [86] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor. Primal space sparse kernel partial least squares regression for large scale problems. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2004)*, Budapest, Hungary, July 2004.
  - [87] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subset based least squares subspace regression in RKHS. *Neurocomputing*, vol. 63, January 2005, pp. 293-323.
  - [54] M. Espinoza, K. Pelckmans, L. Hoegaerts, J.A.K. Suykens, and B. De Moor. A comparative study of LS-SVMs applied to the Silver Box identification problem. In *Proceedings of the 6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004)*, Stuttgart, Germany, September 2004.

- *Introducing sparsity.* In comparison to the standard Support Vector Machines (SVM), LS-SVM only requires solving a linear system, but it lacks sparseness in the number of solution terms because it has not a built-in simultaneous resolution for this problem. Therefore LS-SVM has to be combined with an apart criterion that introduces sparsity. Pruning is a viable remedy. We suggested an important improvement on a pruning method from the literature [39] and made an empirical contribution by comparing some existing techniques [85]. Further we considered up/downdating strategies for the LS-SVM in an online setting, for which we derived a taxonomy of induced data point selection criteria.

  - [85] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. A comparison of pruning algorithms for sparse least squares support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing (ICONIP 2004)*, Calcutta, India, November 2004.

- *Efficiently updating and tracking the dominant kernel eigenspace.* The dominant set of eigenvectors of the symmetric kernel Gram matrix is used in many different kernel methods. In case of large data sets its batch computation becomes overly time consuming. We proposed an efficient incremental approach for fast calculation of the dominant kernel eigenbasis, which allows to track the kernel eigenspace dynamically. We developed a numerical algorithm to compute the approximation based on a given best rank estimate and successive minimal SVD and QR decompositions. Experiments show that our updating scheme delivers a numerically stable and accurate approximation for eigenvalues and eigenvectors at every iteration in comparison to the batch algorithm.

  - [82] L. Hoegaerts, L. De Lathauwer, J.A.K. Suykens, and J. Vandewalle. Efficiently updating and tracking the dominant kernel eigenspace. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2004)*, Leuven, Belgium, July 2004.
  - [81] L. Hoegaerts, L. De Lathauwer, I. Goethals, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Efficiently Updating and Tracking the Dominant Kernel Principal Components. *Submitted for publication*, 2005.

- *Generalized higher-order Grassmann-Rayleigh quotient iteration.* We derive a Grassmann-Rayleigh Quotient Iteration for the computation of the best rank-(R_1, R_2, R_3) approximation of higher-order tensors [41]. It consists of a generalization to tensors of the Rayleigh Quotient Iteration (RQI) for the computation of an invariant subspace of a given matrix [2]. It also generalizes the RQI for the best rank-1 approximation of higher-order tensors [200]. We developed a numerical algorithm to compute the approximation, based on solving a linear set of equations only, and with a quadratic rate of convergence.

  - [41] L. De Lathauwer, L. Hoegaerts, and J. Vandewalle. A Grassmann-Rayleigh quotient iteration for dimensionality reduction in ICA. In *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, Granada, Spain, September 2004.

Chapter 2

The Kernel Paradigm in Machine Learning

In this chapter we describe basic concepts tied with historic events in order to sketch the setting of kernel methods in machine learning. It is a general overview of the framework on which kernel models are based. We start from the distinguished elements that constitute learning from examples and focus on the performance measure that drives the inference mechanism. Adopting the statistical learning point of view, we learn how the principle of empirical risk alone fails to yield a satisfactory solution and how the structural risk principle provides a remedy. Historically, it took then some time before the structural risk principle became finally linked with a strategy of generalizing linear models with nonlinear capabilities through kernel functions. Very naturally, the combination turns out to be embedded in the structure of Reproducing Kernel Hilbert spaces, providing a solid framework for the class of kernel-based Support Vector Machines.

2.1 Learning from examples

The elements of learning consist of the key words task, performance and experience. A learner is supposed to improve from experience (or examples) with respect to some class of tasks and a performance measure [123]. Machine learning considers the problem of constructing a good learner (described by a model or algorithm) for the learning of an (unknown) relation from a set of given examples.

By *examples* we mean pairs of elements of two sets. Since any element can be

reduced to an ordered set of one or more quantitative properties, one mostly represents elements by means of vectors of real numbers. This choice makes mathematical manipulation easy.

By *relation* we mean a functional dependency between elements from the two sets: every input vector is connected to a target vector. Let us thus be given a dataset, consisting of n examples,

$$D = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \tag{2.1}$$

where $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}, \mathbf{z}_i = (\mathbf{x}_i, y_i)$ and $y = f(\mathbf{x})$. The separate components of the vector $\mathbf{x}_i$ are called variables. In statistics one describes the example pairs as samples drawn randomly and independently from an unknown joint distribution of two random variables X and Y, which is related to the marginal and conditional distribution by the relation $F(x, y) = F(x)\ F(y|x)$. We will restrict ourselves to the case where samples with y-values (response examples) are available and thus deal only with supervised learning (in contrast to unsupervised learning).

By *model* we mean a flexible prescription, a conceptual representer g for the relation, in the form of a function. Mostly one parametrizes the model with a parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^s$ that determines the function completely when fixed. In this manner we can specify easily a whole set of models when we consider the parameters as free variables. Remark that also assumptions (whether implicit or explicit) are considered as part of the model specification.

In this context one discerns on the one hand parametric models: when one fully knows the functional form, except for its parameters. In statistics this means that the underlying distribution of responses in the population from which the sample is drawn, is known. And on the other hand, one speaks of non-parametric models when one doesn't know anything about the functional form. In statistics this means that the underlying distribution of responses in the population from which the sample is drawn, is not known, but only the distribution of the variates themselves.

By *learning* we mean the process of approximating the relation: inference. Clearly, there are multiple ways to achieve that goal and many methodologies have been developed over time. Neglecting the western ancient history of qualitative inference, the two oldest, general, important examples are (*i*) hypothesis testing and maximum likelihood maximization [61] from regular statistics or the frequentist school and (*ii*) occam's razor, prior expliciting and posterior maximization [113] from Bayesian statistics or the Bayesian school.

Around 1950, lots of specialized paradigms were developed, constituting the entire spectrum between practice and theory. To mention some well-known classes: decision trees, instance-based learning, rules learning, reinforcement learning, neural networks, computational learning, function approximation theory, etc. [123, 14]. Piece by piece these paradigms have been playing major

and minor roles in the arena of machine learning, with varying success. In that respect, the last decade is now being dominated by *statistical learning theory*.

The latter has gained a lot of support in recent years through the seminal work of Vapnik [183], that attempts to synthesize and refocus on the essence that underlies the plethora of machine learning methodologies, in a most general, analytical, rigorous and conceptual manner. The development of the pragmatic consequences from Vapnik's work have considerably influenced the last decade of research. This work draws largely upon that inheritance and it will serve as our starting point and view.

In the statistical learning framework, inference comes down to choosing an appropriate model from a given set G of possible functions. To assess the quality of a model, and also the convergence rate of the learning, we need a performance measure.

2.2 Risk

When a model gives accurate predictions of a target value to a corresponding input vector, we say the model performs well, or that it generalizes well. To be unambiguous, a performance measure must be represented by a one-dimensional scalar function on the set of functions G, which is called a functional

$$R : G \to \mathbb{R} : g(\cdot) \mapsto R(g(\cdot)). \tag{2.2}$$

The optimality of the quality criterion can always be reformulated into a minimization problem:

$$g^* = \min_{\mathbf{z} \in D, g \in G} R(g(\mathbf{z})). \tag{2.3}$$

If the set G is continuous and the performance functional is explicitly known, it is considered to be the subject of the calculus of variations. It considers solutions to the problem of finding from a given set G of functions over a continuous domain the optimal one. The issue there is mainly to propose a best search method.

Yet, here we consider a *discrete* and *finite* domain of examples. And more, the examples are *stochastic*, they have a probability of occurence. The examples are instances of a random variable Z and they are drawn independently and randomly from an unknown, but fixed, distribution $F(\mathbf{z})$. These constraints complicate the formulation of a solution considerably.

As a function of a random variable, the performance functional becomes a random variable too, and will contain the observation explicitly as argument, represented by $L(\mathbf{z}, g(\mathbf{z}))$. The most natural scalar to take for the performance

is then the expectation value:

$$R(g(\mathbf{z})) = \int L(\mathbf{z}, g(\mathbf{z}))\ dF(\mathbf{z}). \tag{2.4}$$

In a practical setup one parameterizes the functions g in G by a parameter vector α such that $g(\mathbf{z}, \alpha)$ determines uniquely one function in G such that the functional becomes

$$R(\alpha) = \int L(\mathbf{z}, \alpha)\ dF(\mathbf{z}). \tag{2.5}$$

One adopts the convention to let the best performance correspond to the minimum. Therefore this function represents the expected loss, *generalization error*, or risk, and is called the risk functional, and L is a loss function.

2.2.1 Empirical Risk

Since the probability distribution $F(\mathbf{z})$ is unknown, one cannot compute the risk directly. Instead we will have to resort to a set of assumptions that motivate a valid explicit form as an approximation to the risk. Because we infer from limited data to obtain an explicit functional as an approximation for the true general functional, the risk minimization problem is said to be solved according to an *induction principle*.

A well-known example thereof is the *empirical risk minimization principle*. It assumes that each example has the same probability $1/n$ and approximates the integral by only a sum over the n examples:

$$R_{emp}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{z}_i, \alpha). \tag{2.6}$$

Three specific, important learning problems are in fact such instances of this general problem of risk minimization:

- *Density estimation.* The derivative $p(\mathbf{x})$ of a continuous (probability) distribution function $F(\mathbf{x})$ is a density function. The loss function is often taken as $L(\mathbf{z}, \alpha) = -\ln\ p(\mathbf{x}, \alpha)$ because it can be proven that it brings $p(\mathbf{x}, \alpha)$ in a L_1 sense close to the true density $p(\mathbf{x})$. It is also the negative of the maximum likelihood function encountered in statistics.
- *Pattern recognition.* In classification a pattern $\mathbf{x}$ must be assigned to one class y of a total of k classes. So $\mathbf{x} \in \mathbb{R}^p$, $y \in \{0, 1, ..., k-1\}$ and $\mathbf{z} = (\mathbf{x}, y)$. The loss function in a natural and most simple form, can be an indicator function $L(\mathbf{z}, \alpha) = I(y, g(\mathbf{x}, \alpha))$ which returns the value zero if $g(\mathbf{x}) = y$ and one otherwise. Empirical risk minimization with this function is minimizing the number of erronously classified examples. Other more sophisticated loss functions can be defined of course.

- *Regression estimation.* In regression, a point $\mathbf{x}$ is related to a scalar y by a function. It is a pattern recognition problem with infinitely many classes; yet, one can also see classification as a regression of the discrimation function (the boundary between the classes).

  The loss function in this case is often the L_2 norm of the prediction $g(\mathbf{z})$ and the target y: $L(\mathbf{z}, \alpha) = (y - g(\mathbf{x}, \alpha))^2$. Minimization of the risk with this loss function brings $g(\mathbf{x}, \alpha)$ L_2-close to the expectation of the conditional probability $F(y|\mathbf{x})$, and is therefore called regression. Empirical risk minimization with this function is minimizing the L_2-norm of the difference between prediction and target values on examples.

  Of course other loss functions may be considered. The good news is that one can use the loss function that suits best the regression problem at hand. If the samples were generated by an underlying function f with additive noise such that $y_i = f(\mathbf{x}_i) + p(e_i)$ with density $p(e_i)$, then the optimal loss function in a maximum likelihood sense is simply $L(\mathbf{z}, \alpha) = -\ln p(y - f(\mathbf{x}))$. Thus if we use the above L_2-norm, we make an implicit assumption of an additive Gaussian noise model. In Table 2.1 we give some examples of other loss functions.

  But the bad news is that (i) the error density is hardly ever known or very difficult to derive from data without a model (chicken-egg situation), and (ii) a loss function based on the exact error density leads to difficult optimization problems or is nonconvex. In practice one tries to find a risk functional that has favorable computational optimization characteristics and is also as close as possible to the most optimal one from a given set of functions.

Of course machine learning is a very broad field that conveys many more challenges. Sometimes the goal is not prediction, but rather explanation, selection of best variables, decomposition into certain desired components, compression of information, clustering, etc. Nevertheless these different problems can often be described within this framework. The empirical risk minimization principle with all its consequential and deep intricacies is basically at the heart of the present day machine learning practice and theory.

	loss function $c(e)$	**density function** $p(e)$
L_1	$\lvert e\rvert$	$\frac{1}{2}\exp\left(-\lvert e\rvert\right)$
L_2	$\frac{1}{2}e^2$	$\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{e^2}{2}\right)$
Vapnik	$\lvert e\rvert_\varepsilon := \begin{cases} 0 & \text{if } \lvert e\rvert \le \varepsilon \\ \lvert e\rvert - \varepsilon & \text{otherwise} \end{cases}$	$\frac{1}{2(1+\varepsilon)}\exp\left(-\lvert e\rvert_\varepsilon\right)$
Huber	$\begin{cases} \frac{1}{2a}e^2 & \text{if } \lvert e\rvert \le a \\ \lvert e\rvert - \frac{a}{2} & \text{otherwise} \end{cases}$	$\propto \begin{cases} \exp\left(-\frac{e^2}{2a}\right) \\ \exp\left(\frac{a}{2} - \lvert e\rvert\right) \end{cases}$

Table 2.1: Some examples of common loss functions.

2.2.2 Structural risk

Although the empirical risk principle is a practical working methodology, in its general formulation it is suffering from ill-posedness. Loosely speaking, a problem is said to be ill-posed if a solution (i) does not exist, (ii) is not unique or (iii) is not stable. Unstability of the learning machine is the main cause mostly, and it means that small deviations in parameters, cause large deviations in the solution. For more details we refer the reader to [18].

The ill-posedness of learning seems to be an inherent property. On the one hand, even if one reduces the risk to explicit forms, like above, it must be dealt with afterwards in the optimization step [67]. But on the other hand, it also reoccurs when an alternative approach is followed. Indeed, maximal knowledge on each of the above three learning problems would be obtained if one could get at the joint distribution $F(x, y) = F(y|x)\ F(x)$ or their densities. Why not start right away from the definitions of these and then make approximations and leave out any (empirical) risk minimization? Unfortunately, it leads again to ill-posedness. The definitions of $F(x, y), F(y|x)$ and $F(x)$ are in fact equations of the general form $Af = F$, with A a continuous linear integral operator, f in G, and F the distribution function. Solving for f comes down to a true identification of the underlying relation for any possible input [183].

The remedy for the latter ill-posed problem dates back to the early 1960s and was named regularization. It is generally due to Tikhonov, Philips and Ivanov. They proposed three different regularization approaches that are essentially equivalent.

- Minimize the empirical risk subject to a bound on the complexity [95]

$$\arg\min_{f \in G} R_{emp}(f, \mathbf{z}) \quad \text{subject to } \|f\|^2 \leq \tau \in \mathbb{R}_0^+. \tag{2.7}$$

- Minimize the complexity subject to a bound on the empirical risk [136]

$$\arg\min_{f \in G} \|f\|^2 \quad \text{subject to } R_{emp}(f, \mathbf{z}) \leq \mu \in \mathbb{R}_0^+. \tag{2.8}$$

- Minimize the trade-off between complexity and empirical risk [170, 171]

$$\arg\min_{f \in G} R_{emp}(f, \mathbf{z}) + \lambda\ \|f\|^2 \quad \text{with } \lambda \in \mathbb{R}_0^+ \tag{2.9}$$

The thorough theoretical underpinnings for the empirical Risk principle show that it is ill-posed itself [183]. A basic question of statistical learning theory is to find out the necessary and sufficient conditions under which the true minimum $R(\alpha_0)$ (for infinitely many data) is close to the actually obtained $R(\alpha_n)$ (based on n data instances). Through an appropiate induction principle

for limited data, one tries to find a most thight bound on the difference between the considered and the smallest possible risk, such that with probability $1 - \eta$:

$$R(\alpha_0) \leq R(\alpha_n) + \epsilon \left(\sqrt{\frac{h}{n}}, \eta \right), \tag{2.10}$$

where the generalization bound ϵ is an increasing function depending on the finite number of data n, η and characteristics of the set of functions, related in a non-trivial manner to a sophisticated complexity/flexibility measure h on the considered sets.

A related question concerns consistency; one imposes for the learning process that if the number of examples n goes to infinity, that the generalization bound ϵ should go to zero. The asymptotic study of necessary and sufficient conditions under which consistency holds, reveals the specifics of the dependencies in the bound. In their search for the best algorithm, Vapnik and Chervonenkis let themselves be led by their theoretical results on nonasymptotic and distribution-free bounds and their resulting rates of convergence.

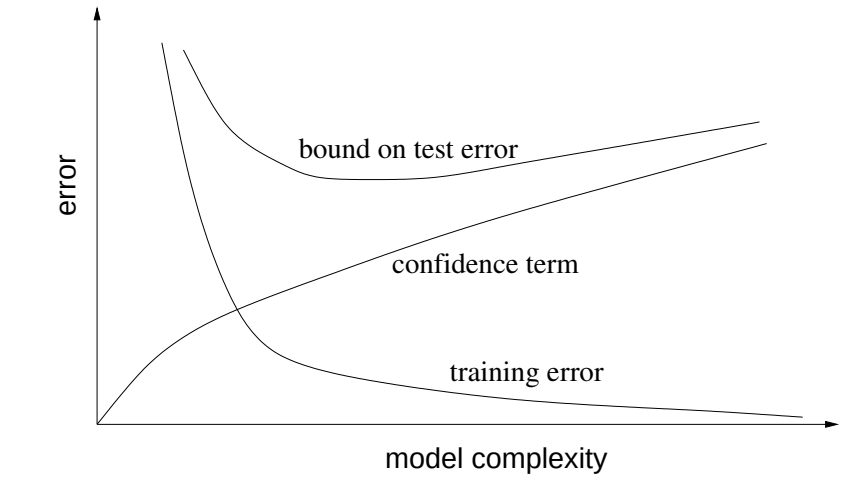
First of all, they concluded that restricting the set of possible functions G is necessary for the consistency of risk minimization. It comes down to working with sets that have finite complexity h. But secondly, (2.10) says that next to empirical risk minimization, also complexity must be tuned. Vapnik therefore considers the minimization over structures: nested function sets $G_1 \subset G_2 \subset \ldots \subset G_i \subset G_{i+1} \subset \ldots \subset G$, where each set G_i has a finite capacity h_i larger than the previous spaces such that $h_1 < h_2 < \ldots < h_i < h_{i+1} < \ldots < h$. On the one hand the empirical risk decreases with increasing complexity, while on the other hand the generalization error bound increases with complexity. A tradeoff for the generalization bound must thus be made. This is illustrated in Figure 2.1.

The latter strategy has been termed *Structural Risk Minimization*. Vapnik concluded that in order to find the best solution for a finite amount of data, one needs to make a compromise between Empirical Risk minimization and the capacity of the learning algorithm (to implement arbitrary functions). Capacity control on the class of functions is an essential ingredient in order to resolve the ill-posedness of Empirical Risk minimization.

2.2.3 Non-linearity

A second important ingredient of learning algorithms is that they need to be able to deal with non-linearities in the relation. We take at the same time a brief historical view with some highlights in machine learning.

Linear functions have been investigated for the last twohundred years. Pearson [133] is often seen as one of the first to publish a rigorous treatment on



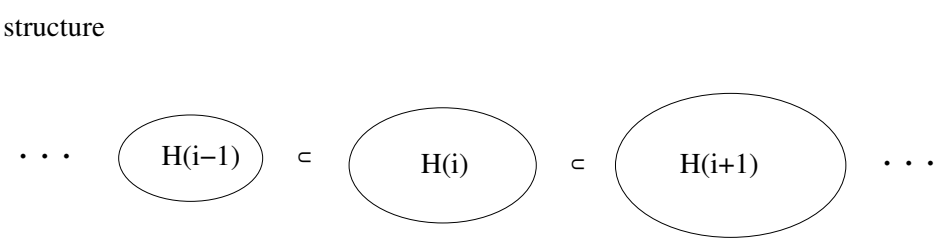


Figure 2.1: Principle of structural risk minimization, illustrating that the generalization bound depends on the model complexity.

correlation and regression in 1896. Since then, the developments gained steadily momentum with the formalization of statistics over the period of 1920 to 1950, with subjects like multiple regression, parametric models and maximum likelihood [61]. The invention of the computer created extra impuls both on practical and theoretical grounds. Subjects like density estimation and non-parametric models benefited greatly.

A lot of theoretical work on the potential of machine's capabilities to learn took place in the early days of the computer. In the field of artificial intelligence first studies of linear classifiers (perceptrons) appeared [145, 144]. One of the most influential works was the textbook by Minsky and Papert [122] in 1969 in which they investigate whether or not it is realistic to expect machines to learn complex tasks. They found that simple, biologically motivated learning systems

called perceptrons were incapable of learning an arbitrarily complex problem. This negative result was demotivating active research in the field for the next ten years.

Almost twenty years later, in 1986, the work by Rumelhart et al. [149] renewed interest in the problem of machine learning. The paper presented an efficient, locally optimal learning algorithm for the class of neural networks, a direct generalization of perceptrons. Since then, an enormous number of papers and books have been published about extensions and empirically successful applications of neural networks.

It was over the period of 1950-1995 that the Vapnik-Chervonenkis (VC) Theory was developed, which would eventually culminate in a paradigm called statistical learning theory [183]. One of the cornerstone viewpoints is that when a problem has to be solved, one has to try to avoid solving a more general problem as an intermediate step. So, when a classification boundary between two classes has to be found, the computation of a probability density is to be avoided, according to Vapnik.

In line with this idea, Vapnik [181] proposed in 1963 a linear classifier for finding a hyperplane optimally dividing two (linearly separable) classes, which does not depend on a probability estimation. This optimal hyperplane is a linear decision boundary which separates the two classes and leaves the largest margin between the vectors of the two classes. It was observed that the optimal hyperplane is determined by only a small fraction of the data points, the so-called *support vectors*. The support vector classifier training algorithm is a procedure to find these vectors.

Furthermore, it was shown that the probability of making an error depends only on the number of these support vectors (and therefore the complexity of the classifier), the number of training errors, and the number of training vectors. In the distributions-free procedure the dimensionality of the input space is of no importance [185]. At the end of the 1960's the empirical risk minimization theory for the classification problem was constructed [180] and later also for the regression case [182]. Also the adaptation of ill-posed problems to machine learning, ideas of capacity control, structural risk principle, Vapnik-Chervonenkis dimension and regularization were tied together in the mid 1970s [183].

It took till 1992 when Boser, Guyon and Vapnik [17] generalized the method into a non-linear classifier. They basically replaced the dot product by a *non-linear kernel function* and proposed to transform the vectors from the input space to some high-dimensional *feature space*. This *kernel trick* allows for more difficult problems to be solved and increases the feature space dimensionality, but hardly influences the classifier complexity or the computations of the algorithm. The introduction of the kernel function for ordinary dot products is an idea due to [3]. In Section 2.3 we will go deeper into the characterisation of the non-linearity

through the kernel function.

In 1995, Cortes and Vapnik [29] suggested a modified margin idea that allows for mislabeled examples and thus extended the non-linear classifier for the case of non-separable classes. Vapnik generalized SVM for estimating non-linear real-valued functions [183]. The proposed learning methods based on statistical learning theory are considered as one of the most important contributions and prepared the onset of a Support Vector Machine era.

2.3 Reproducing Kernel Hilbert Space

2.3.1 Regularization and nonlinearity together embedded

Mostly one desires for the set of functionals G a vector space, that is closed under addition, scalar multiplication and has convenient properties like associativity and distributivity. In order to measure neighbourhoods and distances, G should also be equipped with an inner product. A convenient candidate often is found in *Hilbert spaces*, which are endowed with sufficient topological structure. The $\mathcal{L}_2(\mathcal{X})$ Hilbert space of square integrable functions over input space $\mathcal{X}$ is a popular example.

The question is now how to define a Hilbert space of functions such that we can unify the requirements of (i) non-linearity and (ii) the structural risk minimization principle. Hints for appropiate conditions are in fact available from the above regularization principles (2.7)-(2.9), where the key is to bound functions. The $\mathcal{L}_2(\mathcal{X})$ space for example would not do since it contains many functions with highly irregular behaviour, taking values at infinity.

It turns out that simply choosing general functionals that are bounded is enough to establish those two demands. More formally it means that for each $f \in \mathcal{H}$ and each $\mathbf{x} \in \mathcal{X}$ there exists a constant $c_{\mathbf{x}}$ (independent of f), such that

$$|f(\mathbf{x})| \leq c_{\mathbf{x}} \|f\|. \tag{2.11}$$

But this property is equivalent to saying that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ [190]. Such a RKHS is associated with a positive-definite kernel function

$$k : X \times X \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j), \tag{2.12}$$

and according to the Riesz representation theorem, there exists an evaluation representer $k(\mathbf{x}, \cdot) \in \mathcal{H}_k$ of f at $\mathbf{x}$ such that

$$\langle k(\mathbf{x}, \cdot), f \rangle = f(\mathbf{x}). \tag{2.13}$$

Remark that the kernel function is symmetric, real-valued and coincides with an inner product function and that

$$\langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \rangle = k(\mathbf{x}_i, \mathbf{x}_j), \tag{2.14}$$

so it reproduces itself. Remark that one obtains a RKHS on $\mathbb{R}^p$ under the condition that the kernel is positive definite and vice versa [5]. Some examples of commonly used kernels are given in Table 2.2.

kernel	$k(\mathbf{x}_i, \mathbf{x}_j)$	hyperparameters
linear	$\mathbf{x}_i^T \mathbf{x}_j$	
polynomial	$(\mathbf{x}_i^T \mathbf{x}_j + a)^d$	$d \in \mathbb{N}, a \in \mathbb{R}_0^+$
Gaussian	$\exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2}\right)$	$\sigma \in \mathbb{R}_0^+$
sigmoid	$\tanh(\kappa\ \mathbf{x}_i^T \mathbf{x}_j + d)$	$\kappa, d \in \mathbb{R}$

Table 2.2: Some examples of commonly used kernels. The linear kernel is the classic dot product. The Gaussian kernel is also sometimes named Radial Basis Function (RBF) kernel. By use of the sigmoid kernel one can describe neural network Perceptrons.

Remark 1. The back door through which one can introduce nonlinearity is by a proper choice of the kernel k, which provides a similarity measure between pairs of data points. This becomes more clear when we mention the implicit *feature mapping* that is hidden in the replacement of the classical dot product by a kernel product (kernel trick). One can associate to each $\mathbf{x} \in \mathbb{X}$ a mapping

$$\varphi : \mathbb{X} \to \mathcal{H}_k : \mathbf{x} \mapsto k(\mathbf{x}, \cdot), \tag{2.15}$$

which can be evaluated at $\mathbf{x}'$ to give

$$\varphi(\mathbf{x})(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}'). \tag{2.16}$$

So by working with mapped data points (features), the dot products in $\mathcal{H}_k$ (feature space) of the data points will become nonlinearly computed values. Hence a non-linear algorithm is obtained. At least, as long as one can express all computations of the original algorithm as inner products.

Remark 2. In [55] it is proven that the capacity of the set of functions $\{f \in \mathcal{H}_k | \|f\| < c\}$ depends on c, with c a positive constant. By relaxing the norm $\|f\| < c_i$, for $\{c_i\}_{i=1}^q$ a monotonically increasing sequence of positive constants, a nested sequence of function spaces $H_1 \subset H_2 \subset \ldots \subset H_i \subset H_{i+1} \subset \ldots \subset H_q$ is created in the RKHS $\mathcal{H}_k$ with increasing capacity. And this is exactly what one wanted to achieve in order to implement the Structural Risk minimization principle.

2.3.2 The feature map

The Mercer-Hilbert-Schmidt theorem reveals more about the nature of the feature map φ by stating that for each positive definite kernel there exists an orthonormal set $\{\phi_i\}_{i=1}^d$ with non-negative λ_i such that we have the following

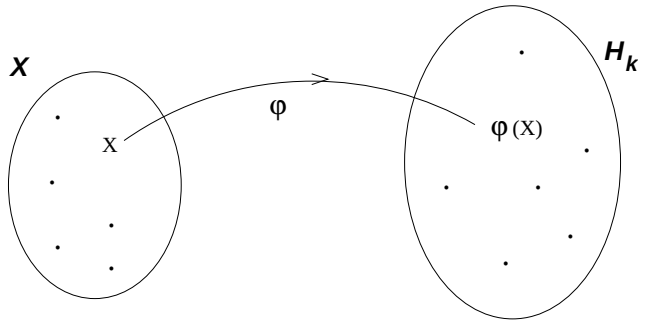


Figure 2.2: The central idea for kernel algorithms in the learning theory context is to change the representation of a data point into a higher-dimensional mapping in a reproducing kernel Hilbert space (RKHS) by means of a kernel function. When appropiately chosen, the kernel function with arguments in the original space corresponds to a dot product with arguments in the RKHS. This allows to circumvent working explicitly with the new representation as long as one can express the computation in the RKHS as inner products.

spectral decomposition [5]:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} \lambda_i \ \phi_i(\mathbf{x})\phi_i(\mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle, \tag{2.17}$$

where $d \leq \infty$ is the dimension of of the RKHS, and λ_i and ϕ_i are the eigenvalues and eigenvectors of the kernel operator, defined by the integral equation

$$\int k(\mathbf{x}, \mathbf{x}')\phi_i(\mathbf{x})p(\mathbf{x})\mathrm{dx} = \lambda_i \phi_i(\mathbf{x}'), \tag{2.18}$$

on $L_2(\mathbb{X})$. Consequently one can then express any functional with respect to the eigenbasis, $f(\mathbf{x}) = \sum_{i=1}^{d} a_i \phi_i$. This allows to formulate an inner product in $\mathcal{H}_k$ in terms of expansion coefficients:

$$\langle f, g \rangle = \langle \sum_{i=1}^{d} a_i \phi_i, \sum_{j=1}^{d} b_j \phi_j \rangle = \sum_{i=1}^{d} \frac{a_i b_i}{\lambda_i}, \tag{2.19}$$

where $\langle \phi_i, \phi_j \rangle = \delta_{ij}/\lambda_i$. A proper scaling of the basis vectors ϕ_i with factor $\sqrt{\lambda_i}$ will transform the inner product to its most simple canonical form of an Euclidean dot product so that $k(\mathbf{x}, \mathbf{x}') = \varphi(\mathbf{x})^T \varphi(\mathbf{x}')$. Furthermore one can express each feature $\varphi(\mathbf{x})$ in this explicit basis so that the φ mapping can be identified with a $d \times 1$ feature *vector*

$$\varphi(\mathbf{x}) = \left[\sqrt{\lambda_1} \ \phi_1(\mathbf{x}) \quad \sqrt{\lambda_2} \ \phi_2(\mathbf{x}) \quad ... \quad \sqrt{\lambda_d} \ \phi_d(\mathbf{x}) \right]^T. \tag{2.20}$$

For regression purposes, just like one builds up a $n \times p$ data matrix X from the $\mathbf{x}_i$ inputs, one constructs with the mapped data points $\varphi(\mathbf{x})$, an $n \times d$ *feature matrix* :

$$\Phi = \begin{pmatrix} \varphi^T(\mathbf{x}_1) \\ \varphi^T(\mathbf{x}_2) \\ \vdots \\ \varphi^T(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_1}\,\phi_1(\mathbf{x}_1) & \sqrt{\lambda_2}\,\phi_2(\mathbf{x}_1) & \dots & \sqrt{\lambda_n}\,\phi_d(\mathbf{x}_1) \\ \sqrt{\lambda_1}\,\phi_1(\mathbf{x}_2) & \sqrt{\lambda_2}\,\phi_2(\mathbf{x}_2) & \dots & \sqrt{\lambda_n}\,\phi_d(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\lambda_1}\,\phi_1(\mathbf{x}_n) & \sqrt{\lambda_2}\,\phi_2(\mathbf{x}_n) & \dots & \sqrt{\lambda_n}\,\phi_d(\mathbf{x}_n) \end{pmatrix}. \tag{2.21}$$

2.3.3 Representer theorem

An important result for the regularization method in a RKHS is that the optimal solution can be shown to live in a finite-dimensional subspace, spanned by the features. This has been stated in the nonparametric representer theorem:

Suppose we are given a nonempty set $\mathcal{X}$, a positive definite real-valued kernel k on $\mathcal{X} \times \mathcal{X}$, a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a strictly monotonically increasing function g on $[0, \infty[$, an arbitrary cost function $c : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$ and a RKHS $\mathcal{H}_k$ associated with k. Then any $f \in \mathcal{H}_k$ minimizing the generalized risk functional

$$c\left((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_n, y_n, f(\mathbf{x}_n))\right) + g(\|f\|) \tag{2.22}$$

admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot). \tag{2.23}$$

Originally, with the mean squared error cost function

$$c\left((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_n, y_n, f(\mathbf{x}_n))\right) = \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 \tag{2.24}$$

and $g(\|f\|) = \lambda \|f\|^2 (\lambda > 0)$, the theorem is due to Whaba [102]. The generalization of c to non-quadratic cost functions was stated by Cox [31]. The generalization to $g(\|f\|)$ was reported by Schölkopf [151].

The significance of this theorem is that it shows that a whole range of learning algorithms have optimal solutions that can be expressed as expansions in terms of the training examples. Another complementary strong result is that also the Universal Approximation Property is guaranteed if the dimension $d = \infty$ and the kernel k is positive definite. The Gaussian kernel of Table 2.2 is an example. This means that any continuous function can be approximated or, more formally, that $\mathcal{H}_k$ is dense in $C(\mathcal{X})$ [37].

2.4 Support Vector Machines

In this section we give a brief overview of Support Vector Machine formulations.

2.4.1 Linear classification of separable classes

The initial idea of Vapnik starts with a linear classification of separable classes [181]. Suppose we have given some set of data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, +1\}$. For separable classes there exist in general many solutions, that can be described as the set of $\mathbf{w}$ and b that satisfy, for all $i = 1, 2, \ldots, n$:

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 0 & \text{if} \quad y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq 0 & \text{if} \quad y_i = -1. \end{cases} \tag{2.25}$$

In order to find a unique solution, Vapnik suggested to require that both classes should be separated with *each* a hyperplane closest to its class, creating a certain margin between them, which is to be minimized. So, one considers instead, for all $i = 1, 2, \ldots, n$,

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq +1 & \text{if} \quad y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 & \text{if} \quad y_i = -1. \end{cases} \tag{2.26}$$

The margin distance equals $\frac{2}{\|\mathbf{w}\|_2}$ and in Figure 2.3 we show an illustration of such a solution. The full problem can be summarized as

$$\begin{cases} \min\limits_{\mathbf{w},b} \quad \frac{1}{2}\,\|\mathbf{w}\|_2^2 \\ \\ \text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \text{for all } i = 1, 2, \ldots, n. \end{cases} \tag{2.27}$$

The minimization of $\mathbf{w}^T\mathbf{w}$ is a similar regularization approach as (2.8) and bears resemblance to the use of a weight decay term in neural networks.

2.4.2 (Non)linear classification of nonseparable classes

A second step was to generalize the problem to non-separable classes. For example under the influence of noise, or due to other errors, some points will end up in regions of another class, with another label. In order to allow for these bad points, Vapnik [29] suggested to introduce a positive misclassification slack variable ξ_i for every point, which should be penalized. In this manner a natural relaxation of the margin classifier could be reached:

$$\begin{cases} \min\limits_{\mathbf{w},b,\xi} \quad \frac{1}{2}\,\|\mathbf{w}\|_2^2 \;+\; C\;\sum_{i=1}^n \xi_i \\ \\ \text{subject to} \quad \begin{array}{ll} y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, & \text{for all } i = 1, 2, \ldots, n, \\ \xi_i \geq 0 & \text{for all } i = 1, 2, \ldots, n. \end{array} \end{cases} \tag{2.28}$$

From an optimization point of view, this convex cost function is stated in the *primal* space, spanned by $\mathbf{w}$ and b. But via the method of Lagrange multipliers one can construct a Lagrangian cost function with Lagrange multipliers α_i (new variables, which span a *dual* space). By deriving the optimality conditions, and eliminating the primal variables, one finally obtains a quadratic programming (QP) problem in *dual* space which leads to a sparse solution with $s \leq n$ terms:

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{s} \alpha_i \; y_i \; \mathbf{x}_i^T \mathbf{x} + b\right). \tag{2.29}$$

Some important assets of the SVM model and the particular solution approach are

- The QP problem can be solved with standard optimization tools. But the computational cost (between linear and cubic in n) is high for large scale data (meaning $n > 1000$). Training is slow, although different remedies have been suggested in the literature (like chunking, sequential minimization, relaxation of constraints, etc., see for example [182, 172, 137, 116]).
- The learning problem has been solved such that there is a reproducible, *global* and *unique* solution. This is in contrast with neural networks, which were plagued by variable local minima optimization outcomes.
- The expansion of the solution (2.29) is in terms of a *subset* of the data points, namely s. This is a consequence of the fact that the margin depends on points near the boundary only, and other well-localized points have zero-valued Lagrange multipliers. The data points with non-zero Lagrange multipliers are called *support vectors*. One says that a *sparse* solution is obtained. This has the advantage that evaluation of new points is fast.
- Geometrical interpretation can be attached to the support vectors: they are located near the decision boundary.
- The size of the solution vector α grows with the number of data points n. Hence, in dual space a non-parametric approach is achieved, while it was a parametric problem in primal space. Especially in problems with high-dimensional input spaces it is better to be more dependent on the number of data points. On the other hand, for large data sets, it is better to solve the primal problem.

A third adaptation consisted of a generalization to *nonlinear* boundaries, that was achieved by implicitly mapping the data into a higher dimensional feature space, as proposed in [17]. The nonlinear separation surfaces in the data space are then corresponding to linear hyperplanes in feature space. Theoretically, this

construction assumes that the framework of RKHS is being employed, in terms as explained above. But in practice, only a simple substitution of $\mathbf{x}$ by $\varphi(\mathbf{x})$ is needed, and a proper kernel must be chosen. The algorithm does not need to be adapted, because it is already formulated in terms of dot products, so that the kernel trick can be applied directly, replacing $\mathbf{x}_i^T \mathbf{x}_j$ by $\varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$. The nonlinear representation of the model then becomes

$$y(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{s} \alpha_i \; y_i \; k(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{2.30}$$

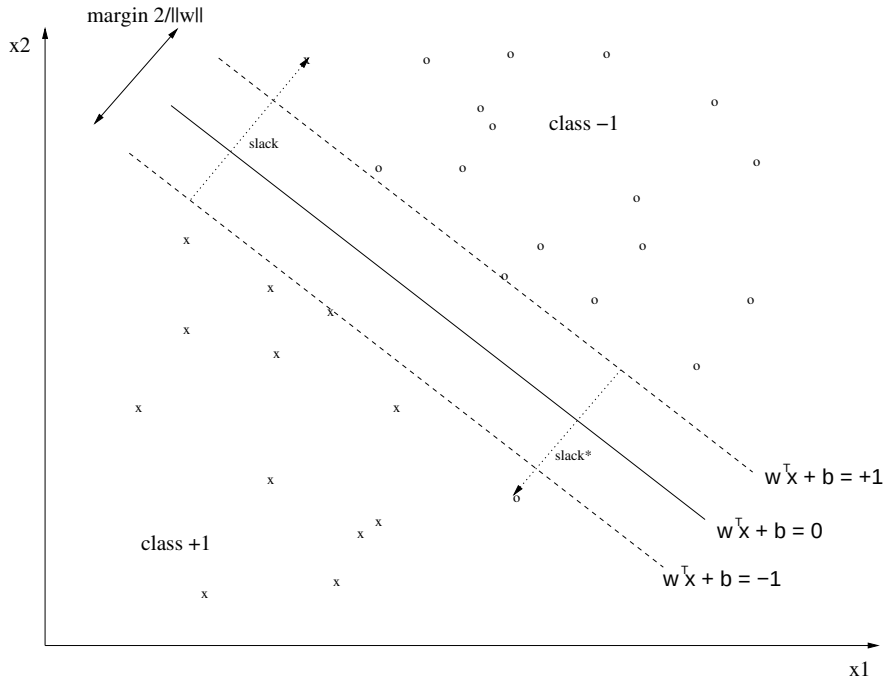


Figure 2.3: The linear Support Vector Machine classifier, maximizing the margin of the separating hyperplanes, with misclassification slack measures in case the data is non-separable.

2.4.3 Linear function estimation of ε-accurate values

In addition to classification, the support vector methodology has been introduced also for linear and nonlinear *function estimation problems* [183]. For this case, in fact the whole line of reasoning of the classification is simply repeated,

with some adaptation to a regression setting of course. First, one considers regression in the set of linear functions

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{2.31}$$

with n given training data points with input values $\mathbf{x}_i \in \mathbb{R}^p$ and output values $y_i \in \mathbb{R}$. For regression in the sense of Vapnik one employs the empirical risk form

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \mathbf{w}^T\mathbf{x}_i - b|_\varepsilon \tag{2.32}$$

with Vapnik's ε-insensitive loss function defined as

$$|y - f(\mathbf{x})|_\varepsilon := \begin{cases} 0 & \text{if} \quad |y - f(\mathbf{x})| \leq \varepsilon \\ |y - f(\mathbf{x})| - \varepsilon & \text{otherwise.} \end{cases} \tag{2.33}$$

The rationale behind this loss function is that it penalizes only points which differ from their real value by an amount larger then a (fixed) *accuracy* ε. Very analogously to the margin idea, the unique $\mathbf{w}$ and b will be determined by a 2ε-wide margin, a tube between 2 hyperplanes that envelope all the data. Again a regularizVapnik [181] proposed in 1963 a linear classifier for finding a hyperplane optimally dividing two (linearly separable) classes, which does not depend on a probability estimation. This optimal hyperplane is a linear decision boundary which separates the two classes and leaves the largest margin between the vectors of the two classes. It was observed that the optimal hyperplane is determined by only a small fraction of the data points, the so-called *support vectors*. The support vector classifier training algorithm is a procedure to find these vectors.

Furthermore, it was shown that the probability of making an error depends only on the number of these support vectors (and therefore the complexity of the classifier), the number of training errors, and the number of training vectors. In the distributions-free procedure the dimensionality of the input space is of no importance [185]. At the end of the 1960's the the empirical risk minimization theory for the classification problem was constructed [180] and later also for the regression case [182]. Also the adaptation of ill-posed problems to machine learning, ideas of capacity control, structural risk principle, Vapnik-Chervonenkis dimension and regularization were tied together in the mid 1970s [183].

It took till 1992 when Boser, Guyon and Vapnik [17] generalized the method into a a non-linear classifier. They basically replaced the dot product by a *non-linear kernel function* and proposed to transform the vectors from the input space to some high-dimensional *feature space*. This *kernel trick* allows for more difficult problems to be solved and increases the feature space dimensionality, but hardly influences the classifier complexity or the computations of the algorithm. The introduction of the kernel function for ordinary dot products is an idea is due to

[3]. In Section 2.3 we will go deeper into the characterisation of the non-linearity trough the kernel function.

In 1995, Cortes and Vapnik [29] suggested a modified margin idea that allows for mislabeled examples and thus extended the non-linear classifier for the case of non-separable classes. Vapnik generalized SVM for estimating non-linear real-valued functions [183]. The proposed learning methods based on statistical learning theory are considered as one of the most important contributions and prepared the onset of a Support Vector Machine era.

ation approach is taken on the norm of the parameter vector $\mathbf{w}$ so that estimation of a linear function is done by solving the following primal space problem:

$$\begin{cases} \min\limits_{\mathbf{w},b} & \frac{1}{2}\ \|\mathbf{w}\|_2^2 \\ \\ \text{subject to} & \begin{array}{ll} y_i - \mathbf{w}^T\mathbf{x}_i - b \leq \varepsilon, & \text{for all } i = 1, 2, \ldots, n, \\ \mathbf{w}^T\mathbf{x}_i + b - y_i \leq \varepsilon & \text{for all } i = 1, 2, \ldots, n. \end{array} \end{cases} \tag{2.34}$$

Of course, due to noise or errors -relative to the value for ε- certain points will be outside the 2ε margin, and the minimization problem (2.34) will become infeasible.

2.4.4 (Non)linear function estimation of non-ε-accurate values

Analogously, one will relax the constraints by introducing *misfit* slack variables $\xi_i, \xi_i^* \in \mathbb{R}_0^+$ for $i = 1, 2, \ldots, n$. The problem (2.34) is then modified into

$$\begin{cases} \min\limits_{\mathbf{w},\xi,\xi^*} & \frac{1}{2}\ \|\mathbf{w}\|_2^2 + C\ \sum_{i=1}^n (\xi_i + \xi_i^*) \\ \\ \text{subject to} & \begin{array}{ll} y_i - \mathbf{w}^T\mathbf{x}_i - b \leq \varepsilon + \xi_i, & \text{for all } i = 1, 2, \ldots, n, \\ \mathbf{w}^T\mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i^* & \text{for all } i = 1, 2, \ldots, n, \\ \xi_i, \xi_i^* \geq 0 & \text{for all } i = 1, 2, \ldots, n. \end{array} \end{cases} \tag{2.35}$$

The constant $C > 0$ determines the tolerated amount of deviation from the desired accuracy ε. Again the minimization can now be done in dual space, which leads to a quadratic programming problem in Lagrange multipliers α_i, α_i^* with a solution in terms of $s \leq n$ support vectors:

$$y(\mathbf{x}) = \sum_{i=1}^{s} (\alpha_i - \alpha_i^*)\ \mathbf{x}_i^T\mathbf{x} + b. \tag{2.36}$$

Similar remarks on the properties of the solution can be made like above for classification. Also the extension to nonlinearity is again done by application of

the kernel trick. The nonlinear representation of the regression model becomes

$$y(\mathbf{x}) = \sum_{i=1}^{s} (\alpha_i - \alpha_i^*)\ k(\mathbf{x}_i, \mathbf{x}) + b. \tag{2.37}$$

For a complete treatment we refer to the extensive literature [184, 98, 35, 154, 165]. In comparison with the nonlinear SVM classifier case, the nonlinear SVM regression case contains extra parameters. In the case of an RBF kernel, σ, C, and ε are considered as additional *tuning* parameters which do not follow from the solution of the QP problem. In the next section we discuss this issue.

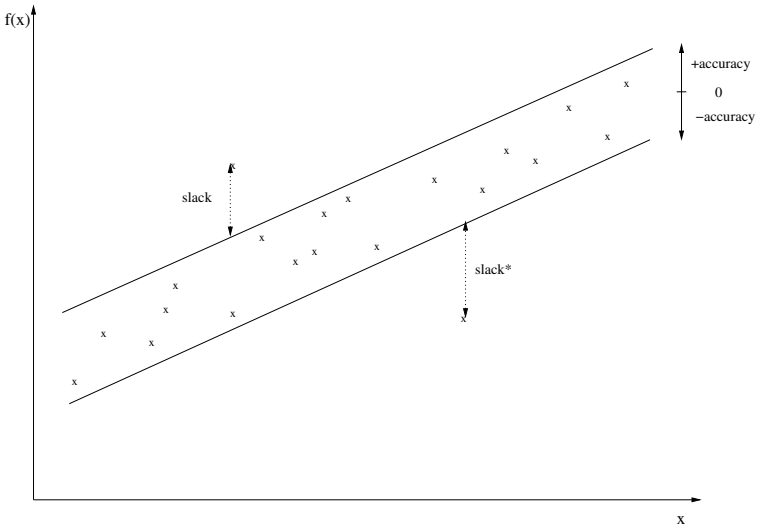


Figure 2.4: The linear function estimator of Vapnik, enveloping the data in a tube with the desired accuracy as radius and with misfit slack measures in case the data contains more variation than twice the accuracy.

2.5 Model selection and assessment

Many models have besides their model parameters also a (mostly smaller) number of extra parameters, which are not on the same level or stand in another relation to the problem formulation. Examples are the kernel parameter (RBF width h), number of components or regularization constant (C). Different hyperparameters lead in general to different models with different performances.

Choosing these hyperparameters in an optimal way is the goal of *model selection*. Once the best model is selected, model assessment is done by estimating prediction error on new data. For this purposes one can best decompose the modelling process in three parts, which is reflected in the utilization of the data.

One correspondingly divides the dataset into three parts: a training set, a validation set and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is chosen for assessment of the final chosen model, and ought not to be used in any parameter determination.

More specifically, the hyperparameters control the *bias-variance trade-off*, associated with model complexity. The best model is one that fits the training data well and generalizes best to test data. When the model is too simple, it results in a high bias (misfit on the training data) and low variance (difference between training and test data performance). But the opposite occurs when the model is too complex (too many degrees of freedom). Typically the training set error decreases with increasing model complexity, but at the same time generalization error (over the test set) will increase due to overfit (see Figure 2.5). Finding the trade-off between the variance and the bias is the goal of model selection.

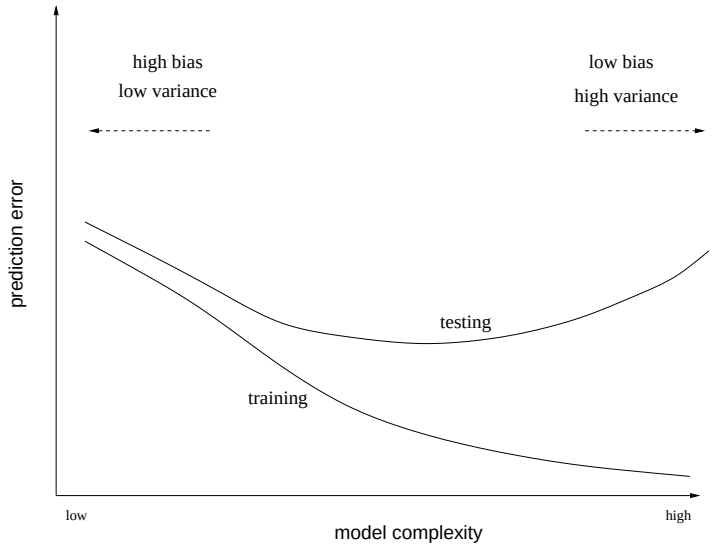


Figure 2.5: Test and training error as function of the model complexity.

Three classes of model selection methods are

- Cross-validation (CV) and bootstrap. These are both empirical methods for estimating generalization error based on resampling. In k-fold cross-validation [161], one divides the data into k subsets of (approximately) equal size. One trains the model k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute generalization error (for an estimate of the bias of a statistic one reserves the term 'Jackknife').

  If k equals the sample size, this is called 'leave-one-out' cross-validation. 'Leave-v-out' is a more elaborate and expensive version of cross-validation that involves leaving out all possible subsets of v cases. When k is small, training sample may be too small to effectively evaluate a model (bias is too high). When k is high, the training data sets are highly similar and the amount of computation is considerable. Studies tend to show that k=5 or 10 is a good compromise.

  In the simplest form of bootstrapping, instead of repeatedly analyzing subsets of the data, one repeatedly analyzes subsamples of the data. Each subsample is a random sample with replacement from the full sample, while typically 50 to 2000 subsamples might be used. There are many more sophisticated bootstrap methods that can be used not only for estimating generalization error but also for estimating confidence bounds for network outputs [53]. Better estimates of prediction error can be obtained than with CV, but at a much higher computational cost.

- Plug-in methods. One considers an approximation of the bias of a function estimator (of an unknown function). An approximate estimator is then plugged in to derive an explicit estimate for the bias (and hence mean squared error, which the optimal tuning parameters effectively minimize). For more information on this approach we refer to [76].

- Complexity criteria. The expected generalization error is in general a function of the variance of y, the variance of our estimate and the bias. The variance of y is beyond our control (irreducible error, mostly estimated by the training error). The two latter are functions of our estimator (with the hyperparameters). Complexity criteria directly estimate this part and then add it to the training error, while cross-validation/bootstrap are based on direct approximations of the generalization error. Examples of criteria are Mallows' C_p [115], Akaike's information criterion (AIC) [4], Bayes Information criterion (BIC) [159] and the Vapnik-Chervonenkis dimension [184].

For practical use in the kernel context data-driven methods like CV and complexity criteria are preferred over plug-in estimators. The disadvantage of CV is that is requires long computation times. The disadvantage of complexity criteria is that they require estimates of the error variance and thus already a reasonable correctly working model, and lead to too conservative estimates,

while CV does not. In this work, being mostly in a data-rich situation, we mainly make use of cross-validation in the selection of the learning parameters.

2.6 Conclusions

We close this chapter by remarking that Vapnik's well-motivated learning machines have set the scene for a myriad of new questions and extensions, that provided a fertile breeding-ground for new research. Its influence and dominance grew quickly, once the versatility and fundaments became known at the end of the 1990s. SVM's caused a paradigm shift for a lot of good reasons:

- Benchmarking and comparisons with other techniques showed that the Support Vector Machines were outperforming in many domains the best alternative techniques [175]. Soon, the SVM had absorbed previous nonlinear paradigms like e.g. multilayer perceptrons [14].
- It brought about a shift towards algorithms that were having a *convex* loss function, with the convenient benefit of a unique and global solution with transparant primal-dual interpretations.
- The (relatively stronger) independence of the dual solutions of the input dimension offered a way to relax the curse of dimensionality (i.e. when the cost and the error will blow up as dimensionality goes up due to sparseness of data).
- The function can be expanded in terms of a limited number of support vectors, sparseness of solution.
- The kernel trick and reintroduction of the RKHS framework made it possible that other existing methods could be transfered into a kernel context in order to extend them with nonlinear capabilities.

The SVM methodology introduced the kernel paradigm in machine learning and laid out a firm, general basis on which new extensions can be built. In the next chapter we will connect to this framework and consider mainly least squares based cost functions for regression and classification problems.

Chapter 3

Least Squares Kernel Machines

In this chapter we focus on an important class of kernel machines that minimize the cost function in a least squares sense, with added regularization. The Least Squares Support Vector Machine (LS-SVM) will serve as the exemplary starting point. In a complementary way, we consider a decomposition of the least squares regression problem in the feature space that regularizes by reduction of projections onto a subspace. We show that known methods, like KPCA, KPLS and KCCA, fit naturally in this framework.

We then explore the kernel PLS case and motivate it with an original primal-dual formulation [84], which allows to motivate and interpret the algorithm in a principled way, in contrast to ad hoc application of the kernel trick. Further we state kernel versions of some different PLS variants and make a comparison between their kernel version [83]. Finally, in an application in Subspace Identification we show how the KCCA subspace admits the nonlinear identification of Hammerstein-Wiener systems [71]. The methods are each time illustrated and compared on a number of illustrative examples.

3.1 Least Squares Support Vector Machines

Vapnik's SVM resulted in a wave of new research activity, where new variations and further improvements are being realized. One of the important modifications to the Vapnik SVM classifier was the LS-SVM classifier originally proposed by Suykens in [167], which simplified the SVM formulation without losing any of its advantages (except sparsity). In this section we elaborate on LS-SVM and

emphasize its primal-dual interpretation.

3.1.1 Model assumptions

We consider the case where there is noise in the dependent variable y. We assume data $\{(\mathbf{x_i}, y_i)\}_{i=1}^n \in \mathbb{R}^p \times \mathbb{R}$ have been given, drawn independently and identically from an unknown distribution according to

$$y_i = m(\mathbf{x}_i) + e_i, \tag{3.1}$$

where $e_i \in \mathbb{R}$ are assumed to be i.i.d. random errors with zero mean and finite variance, $m(\mathbf{x}_i)$ belongs to the space of real-valued smooth functions and represents the conditional mean of y_i for $\mathbf{x} = \mathbf{x}_i$. LS-SVM [165] assumes a model of the form

$$y = \mathbf{w}^T \varphi(\mathbf{x}) + b, \tag{3.2}$$

where the feature map $\varphi : \mathbb{R}^p \to \mathcal{H}_k$ is a function into the r-dimensional Reproducing Kernel Hilbert space (RKHS) [5] $\mathcal{H}_k$ with an associated kernel $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j)$. A common choice is the Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/h^2)$, where h is a (positive) kernel width parameter; or the polynomial kernel $(\mathbf{x}_i^T \mathbf{x}_j + a)^d$ with order $d \in \mathbb{N}$ and constant $a \in \mathbb{R}_0^+$ (see Table 2.2). The mapping k provides a similarity measure between pairs of data points, should fulfill the Mercer condition of positive definiteness and is supposed to capture the nonlinearity, while the model remains linear in the parameters [154].

The weights in $\mathbf{w}$ and bias b are to be estimated by minimizing an error cost function in *primal space*:

$$\begin{aligned} &\min_{\mathbf{w},b,\mathbf{e}} J(\mathbf{w}, b, \mathbf{e}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{n} e_i^2 \\ &\text{subject to} \quad y_i = \mathbf{w}^T\varphi(\mathbf{x}_i) + b + e_i, \quad i = 1, \ldots, n \end{aligned} \tag{3.3}$$

The objective consists of a smallest sum of squared errors term (to fit to the training data) and a regularization term to smoothen the approximation (to compensate overfitting). The relative importance of these terms is determined by the positive real constant γ. In the case of noisy data one avoids overfitting by taking a smaller value of γ.

3.1.2 Model solution

Working with the explicit expression for φ is avoided by taking a Mercer kernel in relation to φ for the dual problem (in the Lagrange multipliers). The

optimization objective becomes the Lagrangian

$$L(\mathbf{w}, b, \mathbf{e}; \boldsymbol{\alpha}) = J(\mathbf{w}, b, \mathbf{e}) - \sum_{i=1}^{n} \alpha_i (\mathbf{w}^T \varphi(\mathbf{x}_i) + b + e_i - y_i), \tag{3.4}$$

where the α_i's are Lagrange multipliers, also called *dual variables*. One solves by deriving the optimality conditions

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i \varphi(\mathbf{x_i}) \tag{3.5}$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i = 0 \tag{3.6}$$

$$\frac{\partial L}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i \quad i = 1, \ldots, n \tag{3.7}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow e_i = \mathbf{w}^T \varphi(\mathbf{x}_i) + b + e_i - y_i = 0 \quad i = 1, \ldots, n. \tag{3.8}$$

Elimination of the variables $\mathbf{e}, \mathbf{w}$ through substitution naturally leads to a solution expressed solely in terms of inner products $\varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$, which results in a linear system [167]:

$$\left[\begin{array}{c|c} 0 & \mathbf{1}_n^T \\ \hline \mathbf{1}_n & K + \gamma^{-1} I_n \end{array}\right] \left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{y} \end{array}\right], \tag{3.9}$$

where $\mathbf{1}_n$ is a column vector of ones, and $\mathbf{y}$ a vector with all the training target values. The entries of the symmetric positive definite kernel matrix K equal $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The role of the (potentially infinite-dimensional) $r \times 1$ weight vector $\mathbf{w}$ in primal space is conveniently taken over by a directly related $n \times 1$ weight vector $\boldsymbol{\alpha}$ in the dual space. For the solution of the parameters of the above linear system we have the explicit expressions

$$\begin{aligned} \hat{b} &= \frac{\mathbf{1}_n^T \left(K + \gamma^{-1} I_n\right)^{-1} \mathbf{y}}{\mathbf{1}_n^T \left(K + \gamma^{-1} I_n\right)^{-1} \mathbf{1}_n} \\ \hat{\boldsymbol{\alpha}} &= \left(K + \gamma^{-1} I_n\right)^{-1} \left(\mathbf{y} - \mathbf{1}_n \hat{b}\right). \end{aligned} \tag{3.10}$$

To apply LS-SVM for *classification* problems the formulation is only slightly adapted. While in function estimation $y \in \mathbb{R}$ depends continuously on $\mathbf{x}$, in classification the aim is basically to predict to which binary class y (suppose +1 and -1) a future object $\mathbf{x}$ belongs, on the basis of a set of given examples $\{(\mathbf{x_i}, y_i)\}_{i=1}^n \in \mathbb{R}^p \times \{-1, +1\}$. Instead one thus formulates a primal space error cost function

$$\begin{cases} \min\limits_{\mathbf{w}, b, \mathbf{e}} \quad \mathbf{w}^T \mathbf{w} + \gamma \sum_{i=1}^{n} e_i^2 \\ \\ \text{subject to} \quad y_i \left(\mathbf{w}^T \varphi(\mathbf{x}_i) + b\right) = 1 - e_i \quad \text{for} \quad i = 1, \ldots, n. \end{cases} \tag{3.11}$$

and solves it in the dual space in an analog manner as above, wich results in a linear system [167]:

$$\left[\begin{array}{c|c} 0 & \mathbf{y}^T \\ \hline \mathbf{y} & \Omega + \gamma^{-1} I_n \end{array}\right] \left[\begin{array}{c} \mathrm{b} \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1}_n \end{array}\right], \tag{3.12}$$

where $\Omega = \mathrm{diag}(\mathbf{y})\ K\ \mathrm{diag}(\mathbf{y})$. The binary class LS-SVM classifier has also been extended to multiclass problems [168, 177].

Typically there is a model selection procedure required (see previous chapter) for the determination of two hyperparameters (γ, h) in the case of a Gaussian kernel. Once these are fixed, the LS-SVM approximator can be evaluated at any point $\mathbf{x}$ by

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \mathbf{w}^T \varphi(\mathbf{x}) + b \\ &= \sum_{i=1}^{n} \alpha_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b \\ &= \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b. \end{aligned} \tag{3.13}$$

As one can notice the LS-SVM model is a kernel expansion where the coefficients are the solution of a $n \times n$ square linear system of equations. This implies that if the matrix is of full rank the dual solution is global and unique [165], which is always ensured via the use of the regularization term.

The computational complexity is independent of the input dimension p and independent of the dimension of the feature space, typically mild operation times are achievable, square or cubic in the size of the training data n and the time of a kernel evaluation.

Related models of regularization networks, Gaussian processes and Kriging have been considered without the bias term b [138]. LS-SVMs have been proposed as a class of kernel machines with a primal-dual formulation for kernel Fisher Discrimant Analysis, kernel Ridge Regression, kernel Principal Component Analysis, kernel Canonical Correlation Analysis, recurrent networks and optimal control [165]. The class of standard SVM's, in the sense of Vapnik, have been extended in this way using primal-dual optimization methodology, in contrast to some kernel methods which cannot be interpreted in this manner.

3.2 Subspace regression in reproducing kernel Hilbert space

We focus on least squares regression models in the kernel context. Instead of regularization via an extra penalty term in the cost function, we now consider

regularization by subspace reduction. We first introduce a general context of kernel regression where merely another choice is made for the subspace. We then discuss (kernel) principal component analysis and (kernel) canonical correlation analysis. We focus further on (kernel) partial least squares and give a least squares support vector machine style derivation with a primal-dual optimization problem formulation. Finally we deal with alternative KPLS-related variants and compare.

The goal is to obtain a regression estimate of the underlying function, given the data $\{(\mathbf{x_i}, \mathbf{y_i})\}_{i=1}^n \in \mathbb{R}^p \times \mathbb{R}^q$. Here, we consider the standard linear multivariate regression model in feature space

$$Y = \Phi W + E, \tag{3.14}$$

where Y represents a $n \times q$ matrix of observations of the dependent variable, Φ is a $n \times d$ matrix of regressors $\{\varphi(\mathbf{x_i})\}_{i=1}^n$, W is the unknown $d \times q$ matrix of regression coefficients and E is a $n \times q$ matrix of errors with zero-mean Gaussian i.i.d. values of equal variance σ^2 (unknown). We will assume all mapped data variables have been mean-centered.

A first difficulty in this setup, is that d may be very large compared to the number of data points n, even $d = +\infty$ potentially. In order to *restrict* the infinite number of regression coefficients, one can introduce a projection of the data into a subspace of finite dimension $s \ll d$, so as to obtain an overdetermined system. Hence the name subspace regression, which is a form of regularization [162], leading to a smaller number of regression coefficients and thus effectively reducing the complexity capacity of the model. If we gather the basis vectors $\{\mathbf{v}_i\}_{i=1}^s$ of the subspace in the columns of a $d \times s$ transformation matrix V, we can express the $\varphi(\mathbf{x_i})$ in the new coordinates $z_i = \varphi(\mathbf{x_i})^T\mathbf{v}$, so that in matrix notation $Z = \Phi V$, where Z is a $n \times s$ matrix of transformed regressors (scores). If $s = d$ then a mere basis change is performed, but when $s < d$ we project into a real subspace and the confined regression model becomes

$$Y = ZW + E, \tag{3.15}$$

where W is now a $s \times q$ matrix of regression coefficients.

The particular choice of the basis vectors, which will affect the quality of the model, is here assumed to be given and its determination on the basis of the data will be discussed in section 3.3. To estimate the unknown true model parameters, the elements of W, we choose here to minimize the error E in least squares sense:

$$\min_W \|Y - ZW\|_2^2. \tag{3.16}$$

A second difficulty is that in general the elements of matrix Φ are unknown because the explicit expression for φ is not available. An elegant and optimal approach is to make use of the so-called kernel trick [154]. The need for a direct expression is typically avoided by formal manipulation towards scalar products

so that kernels come into play. The matrix Φ may be unknown, but the $n \times n$ elements of $\Phi\Phi^T = K$ are easily computable. In the above setup, this can be achieved by introducing a decomposition $V = \Phi^T A$. This allows to write $Z = \Phi V = \Phi\Phi^T A = KA$ and our model equation becomes:

$$Y = (KA)W + E. \tag{3.17}$$

Here the $n \times n$ kernel matrix K can be easily calculated and the computation of the $n \times s$ matrix A will be discussed in detail in section 3.3. So in order to obtain an estimate of our $n \times q$ matrix Y, we have to find the $s \times q$ matrix W of regression coefficients. In the univariate case ($q = 1$) we can write $Y = \mathbf{y}$ as

$$f(\mathbf{x}) = \sum_{i=1}^{n}(\sum_{j=1}^{s} a_{ij}w_j)k(\mathbf{x_i}, \mathbf{x}) = \sum_{i=1}^{n} \alpha_i \; k(\mathbf{x_i}, \mathbf{x}). \tag{3.18}$$

Since we assumed that the data are centered, we need to adjust the expression $\varphi(\mathbf{x})$ with $\varphi(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n}\varphi(\mathbf{x_i})$ everywhere. For the kernel matrices this has the consequence of replacing K by [155]

$$K := (I_n - \frac{1}{n}1_n 1_n^T)K(I_n - \frac{1}{n}1_n 1_n^T) \tag{3.19}$$

where I_n is the identity matrix and 1_n a vector of ones.

3.3 Subspace construction

Basically, we need to find directions in the variable space that form a new basis, upon which we can project. Several criteria can be chosen to arrive at a subspace to confine the regression. We take an overview of two important criteria and an intermediate one.

3.3.1 Minimization of within-space correlation

A more common name for within-space correlation is multicollinearity, the degree of covariance between the data vectors in the variables space. It occurs often when using multiple regression on data that one has collected that one has no full control over the design of the experiment. A high degree of multicollinearity produces unacceptable uncertainty (large variance) in regression coefficient estimates [99, 119].

The commonly used tool for the purpose of multicollinearity reduction is Principal Component Analysis (PCA). PCA is a widely used feature extraction technique. Formally, the new features are simply linear combinations of original variables, weighed according to eigenvectors. Intuitively, they form

(hyper)planes of closest fit to systems of points in space. As such, principal components were for the first time mentioned in a regression motivated least squares sense [133]. Thirty years later they were reintroduced in a factor analysis context [90]. Temporarily, PCA is undoubtedly a very important method which has taken a fundamental place amongst linear methods in the toolbox of data processing communities and statistics. For a broad and thorough overview of this technique a standard reference is [99], from which we briefly draw basics here.

PCA is mostly stated as a problem in which one maximizes the variance of the new variables $\mathbf{s} = \mathbf{v}^T\mathbf{x}$:

$$\max_{\mathbf{v}} \text{var } (\mathbf{v}^T\mathbf{x}) = \mathbf{v}^T C_{xx}\mathbf{v} \tag{3.20}$$

subject to

$$\begin{aligned} \|\mathbf{v}\| &= 1 \\ V^T V &= I_p, \end{aligned} \tag{3.21}$$

with $C_{xx} = X^T X$ the $p \times p$ sample covariance matrix. This involves a diagonalization procedure which requires solving an eigenvalue problem

$$C_{xx}\mathbf{v} = \lambda\mathbf{v}. \tag{3.22}$$

From the viewpoint of dimensionality reduction one can also describe PCA as the search for a best fitting subspace in a least squares sense. So PCA is equivalent to successive minimization of

$$J_{\text{PCA}}(\mathbf{v}) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{v}\mathbf{v}^T\mathbf{x}_i\|^2, \tag{3.23}$$

subject to the constraints (3.21).

3.3.2 Maximization of between-space correlation

On the other hand, the goal is maximization of between-space correlation. For the purpose of prediction one wishes to select subspace input vectors that have strong correlation with the target vectors.

The specific method that implements this criterion is Canonical Correlation Analysis (CCA) [69]. CCA is concerned with describing linear dependencies between sets of variables. The method was originally formulated by Hotelling [91, 92] and is nowadays included in most textbooks on multivariate statistics [117].

CCA considers the $p \times q$ sample covariance matrix $C_{xy} = X^T Y$ between two spaces. To minimize the cross-covariances, again diagonal elements are

maximized

$$\max_{\mathbf{v},\mathbf{w}} \ \mathrm{corr}(\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y}) = \frac{\mathbf{v}^T C_{xy}\mathbf{w}}{\sqrt{\mathbf{v}^T C_{xx}\mathbf{v}}\ \sqrt{\mathbf{w}^T C_{yy}\mathbf{w}}} \tag{3.24}$$

subject to

$$\begin{aligned} \mathrm{var}(\mathbf{v}^T\mathbf{x}) &= \mathbf{v}^T C_{xx}\mathbf{v} = 1 \\ \mathrm{var}(\mathbf{w}^T\mathbf{y}) &= \mathbf{w}^T C_{yy}\mathbf{w} = 1. \end{aligned} \tag{3.25}$$

Essentially this requires the solution of the system

$$C_{xy}\mathbf{w} = \lambda\ C_{xx}\mathbf{v} \tag{3.26}$$

$$C_{yx}\mathbf{v} = \lambda\ C_{yy}\mathbf{w}. \tag{3.27}$$

The new bases in both spaces are chosen such that the vector components (projections) of all data maximally coincide. From a least squares perspective, one successively minimizes

$$J_{\mathrm{CCA}}(\mathbf{v}, \mathbf{w}) = \sum_{i=1}^{n} \|\mathbf{v}^T\mathbf{x}_i - w^T\mathbf{y}_i\|^2 \tag{3.28}$$

subject to the constraints (3.25). Thus in CCA the difference between the cosine of angles of lines in both spaces is minimized.

3.3.3 Intermediate criteria

The two above criteria can be taken as two extremes, and an optimal subspace choice may involve a trade-off. The Partial Least Squares (PLS) [196] method can be positioned in between these two. PLS is a multivariate technique that delivers an optimal basis in x-space for y onto x regression. Reduction to a certain subset of the basis introduces a bias, but reduces the variance. This method was developed between 1962 and 1982 by H. Wold and co-workers [198, 197]. Since then the model was in several directions enhanced and extended, discussing its statistical properties and objective function [89, 64, 80, 118, 38, 112, 62].

In general, PLS is based on a maximization of the covariance between successive linear combinations in x and y space, $\langle \mathbf{v}, \mathbf{x} \rangle$ and $\langle \mathbf{w}, \mathbf{y} \rangle$, where coefficient vectors are normed to unity and constrained to be orthogonal in x space:

$$\max_{\mathbf{v},\mathbf{w}} \mathrm{cov}\ (\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y}) = \mathbf{v}^T C_{xy}\mathbf{w} \tag{3.29}$$

subject to

$$\begin{aligned} \|\mathbf{v}\| = 1 = \|\mathbf{w}\| \\ V^T V = I_p. \end{aligned} \tag{3.30}$$

Solutions can be obtained by using Lagrange multipliers, which leads to solving the following system

$$C_{xy}\mathbf{w} = \lambda\mathbf{v} \tag{3.31}$$
$$C_{yx}\mathbf{v} = \lambda\mathbf{w}. \tag{3.32}$$

As a least squares cost function, PLS turns out to be a sum of the least squares formulation of each of the above methods:

$$J_{\text{PLS}}(\mathbf{v},\mathbf{w}) = J_{\text{PCA}}(\mathbf{v}) + J_{\text{CCA}}(\mathbf{v},\mathbf{w}) + J_{\text{PCA}}(\mathbf{w}) \tag{3.33}$$

subject to the constraints (3.30). Indeed, by simplifying this expression, we obtain the covariance term only: the two variance minimizations of the CCA criterion are compensated by the PCA variance maximizations.

The above PLS version may be called PLS-SVD since the variables $\mathbf{v},\mathbf{w}$ are the integral eigenspaces of $C_{xy}C_{yx}$. Imposing other constraints, results in other PLS variants. From the vast plethora of possible modified PLS constructs that can be found in the literature [162], we include here a few transparant ones:

- The original version of Wold, PLS-WA [196], computes consecutively the first left and right singular vectors of $(X^{(r)})^T Y^{(r)}$, after which each time the data matrices are deflated (projected into the complement of the space spanned by the previous found new variables, or scores, denoted $\mathbf{a}$ and $\mathbf{b}$):

$$\begin{array}{rcl} X^{(r+1)} & = & X^{(r)} - \mathbf{a}_r(\mathbf{a}_r^T\mathbf{a}_r)^{-1}\mathbf{a}_r^T X^{(r)} \quad \text{with} \quad \mathbf{a}_r = X^{(r)}\mathbf{v}_r, \\ Y^{(r+1)} & = & Y^{(r)} - \mathbf{b}_r(\mathbf{b}_r^T\mathbf{b}_r)^{-1}\mathbf{b}_r^T Y^{(r)} \quad \text{with} \quad \mathbf{b}_r = X^{(r)}\mathbf{w}_r. \end{array} \tag{3.34}$$

- The directly adapted most used variant of PLS-WA has resulted in PLS2 (multivariate) or PLS1 (univariate), where y-space is being deflated with the x-space score instead [38]:

$$\begin{array}{rcl} X^{(r+1)} & = & X^{(r)} - \mathbf{a}_r(\mathbf{a}_r^T\mathbf{a}_r)^{-1}\mathbf{a}_r^T X^{(r)}, \\ Y^{(r+1)} & = & Y^{(r)} - \mathbf{a}_r(\mathbf{a}_r^T\mathbf{a}_r)^{-1}\mathbf{a}_r^T Y^{(r)}. \end{array} \tag{3.35}$$

  This variant is well-suited to the asymmetry of the regression context, where the variables are considered as independent and dependent [89].

- We included PLS-U, which is PLS-WA, not with an orthogonality constraint, but more strongly, uncorrelatedness with the previously found coefficients [141]:

$$V_r^T C_{xx} V_r = I_p, \tag{3.36}$$
$$W_r^T C_{yy} W_r = I_q. \tag{3.37}$$

  By Lagrange multipliers one arrives after some lengthy calculation again at an eigenproblem with deflations:

$$\begin{array}{rcl} X^{(r+1)} & = & X^{(r)} - (C_{xx}V_r)((C_{xx}V_r)^T(C_{xx}V_r))^{-1}(C_{xx}V_r)^T X^{(r)}, \\ Y^{(r+1)} & = & Y^{(r)} - (C_{yy}W_r)((C_{yy}W_r)^T(C_{yy}W_r))^{-1}(C_{yy}W_r)^T Y^{(r)}. \end{array} \tag{3.38}$$

- The PLS least squares interpretation allows to add immediately two more variants. Leaving out the compensating PCA term in x space, we obtain PLSx from CCA, with consequently $C_{xx} = I_p$. By symmetry, also a PLSy version can be obtained. These versions may be useful especially when dealing with many dummy variables, such that the PCA contribution does not make much sense [11].

For a generic overview, partly inspired by [16], of the criteria for the methods PCA, CCA and some PLS variants, we refer to Table 3.1.

	covariance maximization	least squares minimization
PCA	$\max_{\mathbf{v}} \frac{\text{var}(\mathbf{v}^T\mathbf{x})}{(\mathbf{v}^T\mathbf{v})}$	$\min_{\mathbf{v}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{v}\mathbf{v}^T\mathbf{x}_i\|^2$
PLS	$\max_{\mathbf{v},\mathbf{w}} \frac{[\text{cov}(\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y})]^2}{(\mathbf{v}^T\mathbf{v})(\mathbf{w}^T\mathbf{w})}$	$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \|\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i\|^2 + \|\mathbf{v}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{y}_i\|^2 + \|\mathbf{w}\mathbf{w}^T\mathbf{y}_i - \mathbf{y}_i\|^2$
PLSx	$\max_{\mathbf{v},\mathbf{w}} \frac{[\text{cov}(\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y})]^2}{(\mathbf{v}^T\mathbf{v})\text{var}(\mathbf{w}^T\mathbf{y})}$	$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \|\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i\|^2 + \|\mathbf{v}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{y}_i\|^2$
PLSy	$\max_{\mathbf{v},\mathbf{w}} \frac{[\text{cov}(\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y})]^2}{\text{var}(\mathbf{v}^T\mathbf{x})(\mathbf{w}^T\mathbf{w})}$	$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \|\mathbf{v}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{y}_i\|^2 + \|\mathbf{w}\mathbf{w}^T\mathbf{y}_i - \mathbf{y}_i\|^2$
CCA	$\max_{\mathbf{v},\mathbf{w}} \frac{[\text{cov}(\mathbf{v}^T\mathbf{x}, \mathbf{w}^T\mathbf{y})]^2}{\text{var}(\mathbf{v}^T\mathbf{x})\text{var}(\mathbf{w}^T\mathbf{y})}$	$\min_{\mathbf{v},\mathbf{w}} \sum_{i=1}^{n} \|\mathbf{v}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{y}_i\|^2$

Table 3.1: PCA, CCA and 3 PLS variants summarized in their (co)variance and least squares formulation.

3.4 Introducing the kernel function

In this section we will briefly apply the typical approach used for obtaining a kernel version of the above covariance criteria. These closely related methods deliver us meaningful $n \times s$ subspaces spanned by the column space of the loading matrix A (where we assume eigenvectors $\{\mathbf{a}_i\}_{i=1}^{s}$ ordered corresponding to nondecreasing values of the eigenvalues λ_i).

3.4.1 Kernel Principal Component Analysis

Ongoing research is involved in the generalization of PCA to a nonlinear form, yielding a variety of approaches [45, 65, 79, 49]. In this section we deal with

PCA in feature space that is able to incorporate nonlinearities, named kernel Principal Component Analysis, and which was originally introduced in [155].

Basically, a transfer of PCA to the kernel context can be described as a mapping of data points and performing linear PCA in Feature space. So, instead of the data points $\mathbf{x}_i$ and the design matrix X from data space, we just deal with the features $\varphi(\mathbf{x}_i)$ and the feature matrix Φ in the kernel version. Starting from criterion (3.22) we need to solve the eigenproblem

$$\Phi^T\Phi\ \mathbf{v} = \lambda\ \mathbf{v}. \tag{3.39}$$

By lack of explicit expression of Φ, one resorts to the application of the kernel trick to bring kernels into play. As was noted in the original paper [155], this is possible because the above solutions $\mathbf{v}$ are exactly linear combination of the $\varphi(\mathbf{x}_i)$ with coefficients $\Phi\mathbf{v}$. Thus we may state $\mathbf{v} = \Phi^T\mathbf{a}$ and perform also a left multiplication with Φ to obtain

$$\Phi\Phi^T\Phi\Phi^T\ \mathbf{a} = \lambda\Phi\Phi^T\ \mathbf{a}. \tag{3.40}$$

which yields, using the fact that K is positive definite, the main result (an eigendecompostion) of KPCA

$$K\ \mathbf{a} = \lambda\ \mathbf{a}. \tag{3.41}$$

For a more principled derivation, with a *primal-dual approach*, we refer to [166], where PCA was originally reported in a support vector machine formulation.

3.4.2 Kernel Canonical Correlation Analysis

Starting from criterion (3.24), we can proceed likewise in feature space, but now with $\mathbf{v}$ and $\mathbf{w}$ as $d\times 1$ feature space vectors. To arrive at calculation with kernels instead of feature vectors one typically expands the new basis vectors as follows

$$\mathbf{v} = \sum_{i=1}^{n} \mathbf{a}_i\ \varphi(\mathbf{x}_i) = \Phi^T A \tag{3.42}$$

$$\mathbf{w} = \sum_{i=1}^{n} \mathbf{b}_i\ \varphi(\mathbf{y}_i) = \Phi^T B. \tag{3.43}$$

By substitution in (3.24) and use of some algebra one arrives at the following criterion

$$\max_{\mathbf{v},\mathbf{w}}\ \mathrm{corr}(\mathbf{v}^T\varphi(\mathbf{x}), \mathbf{w}^T\varphi(\mathbf{y})) = \max_{\mathbf{a},\mathbf{b}} \frac{\mathbf{a}^T K_{xx} K_{yy}\mathbf{b}}{\sqrt{\mathbf{a}^T K_{xx}\mathbf{a}}\ \sqrt{\mathbf{b}^T K_{yy}\mathbf{b}}} \tag{3.44}$$

where $[K_{xx}]_{ij} = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ and $[K_{yy}]_{ij} = \varphi(\mathbf{y}_i)^T\varphi(\mathbf{y}_j)$ are the kernel Gram matrices and $\mathbf{v}$ and $\mathbf{w}$ were divided by their norm. Taking then derivatives with

respect to $\mathbf{a}$ and $\mathbf{b}$ and setting to zero leads to the following coupled system of equations

$$\begin{aligned} K_{yy}\mathbf{b} &= \lambda K_{xx}\mathbf{a} \\ K_{xx}\mathbf{a} &= \lambda K_{yy}\mathbf{b}. \end{aligned} \tag{3.45}$$

Again, a principled primal-dual derivation was proposed in the context of primal-dual LS-SVM formulations, which leads to a more extended 'regularized' KCCA variant [165].

3.4.3 Kernel Partial Least Squares

By direct substitution of (3.42)-(3.43) in the PLS criterion (3.29) and use of some algebra one arrives at the following criterion

$$\max_{\mathbf{a},\mathbf{b}} \frac{\mathbf{a}^T K_{xx} K_{yy} \mathbf{b}}{\sqrt{\mathbf{a}^T\mathbf{a}}\,\sqrt{\mathbf{b}^T\mathbf{b}}}. \tag{3.46}$$

Taking then derivatives with respect to $\mathbf{a}$ and $\mathbf{b}$ and equating to zero leads to the following coupled system of equations

$$\begin{aligned} K_{yy}\mathbf{b} &= \lambda \mathbf{a} \\ K_{xx}\mathbf{a} &= \lambda \mathbf{b}. \end{aligned} \tag{3.47}$$

This variant is the nonlinear counterpart of PLS-SVD, for the other variants from Section 3.3, we have the following expressions:

- For KPLS-WA we can substitute X by Φ_1 and Y by Φ_2. But because of the unknown elements of Φ, we cannot directly obtain the SVD of $(\Phi_1^r)^T\Phi_2^r$. The NIPALS-PLS algorithm [196] allows to circumvent this issue and delivers the $\mathbf{a}$ and $\mathbf{b}$ as first singular vectors of $K_{xx}^{(r)}K_{yy}^{(r)}$ and $K_{yy}^{(r)}K_{xx}^{(r)}$. Then the deflation expressions become for PLS-U at step r:

$$\begin{aligned} K_{xx}^{(r+1)} &= K_{xx} - K_{xx}^2 A_r (A_r^T K_{xx}^3 A_r)^{-1} A_r^T K_{xx}^2 \\ K_{yy}^{(r+1)} &= K_{yy} - K_{yy}^2 B_r (B_r^T K_{yy}^3 B_r)^{-1} B_r^T K_{yy}^2. \end{aligned} \tag{3.48}$$

  The same is valid for KPLS-WA (which resembles the kernel version of PLS1 [148]) where we have:

$$\begin{aligned} K_{xx}^{(r+1)} &= K_{xx}^{(r)} - \mathbf{a}_r\mathbf{a}_r^T K_{xx}^{(r)} - K_{xx}^{(r)}\mathbf{a}_r\mathbf{a}_r^T + \mathbf{a}_r\mathbf{a}_r^T K_{xx}^{(r)}\mathbf{a}_r\mathbf{a}_r^T \\ K_{yy}^{(r+1)} &= K_{yy}^{(r)} - \mathbf{b}_r\mathbf{b}_r^T K_{yy}^{(r)} - K_{yy}^{(r)}\mathbf{b}_r\mathbf{b}_r^T + \mathbf{b}_r\mathbf{b}_r^T K_{yy}^{(r)}\mathbf{b}_r\mathbf{b}_r^T. \end{aligned} \tag{3.49}$$

- The kernel version of PLS2 and PLS1 has been extensively studied in [148].

- For KPLSx and KPLSy we point out that its solutions can be considered in the optimization context as special cases of the regularized KCCA variant of (3.45) if one fixes the positive parameters (ν_1, ν_2) (given below in (3.58)) respectively as $(1, 0)$ and $(0, 1)$. In fact even KPLS-SVD can be considered as a subcase thereof, by setting $(\nu_1, \nu_2) = (0, 0)$.

For a generic overview of the criteria for the methods KPCA, KCCA and some KPLS variants, we refer to Table 3.2. In the next subsection we present for KPLS the principled primal-dual derivation, proposed in the context of primal-dual LS-SVM formulations [84].

	Linear (primal)		Kernel (dual)	
	M	Q	L	R
(K)PCA	S_{xx}	I	K_{xx}	I
(K)PLS	$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}$	$\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$	$\begin{bmatrix} 0 & K_{xx}K_{yy} \\ K_{yy}K_{xx} & 0 \end{bmatrix}$	$\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$
(K)PLSx	$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}$	$\begin{bmatrix} I & 0 \\ 0 & C_{yy} \end{bmatrix}$	$\begin{bmatrix} 0 & K_{xx}K_{yy} \\ K_{yy}K_{xx} & 0 \end{bmatrix}$	$\begin{bmatrix} I & 0 \\ 0 & K_{yy}^2 \end{bmatrix}$
(K)PLSy	$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}$	$\begin{bmatrix} C_{xx} & 0 \\ 0 & I \end{bmatrix}$	$\begin{bmatrix} 0 & K_{xx}K_{yy} \\ K_{yy}K_{xx} & 0 \end{bmatrix}$	$\begin{bmatrix} K_{xx}^2 & 0 \\ 0 & I \end{bmatrix}$
(K)CCA	$\begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}$	$\begin{bmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{bmatrix}$	$\begin{bmatrix} 0 & K_{xx}K_{yy} \\ K_{yy}K_{xx} & 0 \end{bmatrix}$	$\begin{bmatrix} K_{xx}^2 & 0 \\ 0 & K_{yy}^2 \end{bmatrix}$

Table 3.2: (K)PCA, (K)CCA and 3 (K)PLS-SVD variants summarized in their generalized eigenvalue formulation, where $M\,[\mathbf{v}; \mathbf{w}] = \lambda\, Q\,[\mathbf{v}; \mathbf{w}]$ represents the linear version with solutions in primal space and $L\,[\mathbf{a}; \mathbf{b}] = \lambda\, R\,[\mathbf{a}; \mathbf{b}]$ the nonlinear kernel version, with solutions in dual space.

3.4.4 KPLS derived in a primal-dual optimization perspective

In this section, we will give an alternatively motivated approach for KPLS, along the same line of [166], where KPCA was originally reported in a support vector machine formulation and [176] for the kernel KCCA case.

A more principled approach can be taken by starting from the PLS least squares formulation (3.33). This primal cost criterion aims at optimizing the coefficient vectors $\mathbf{v}$ and $\mathbf{w}$, searching simultaneously for the maximal projection of a data point in x space, maximal covariation with the corresponding projection of the point in y space, and maximal projection of a data point in y space. Since the coefficients could become arbitrarily large, these are typically constrained or at

least regularized. By simplifying expression (3.33) we obtain $\langle \mathbf{v^T x}, \mathbf{w^T y} \rangle$ and by adding (soft) regularization, we arrive at the following primal form problem:

$$\begin{aligned} \max_{\mathbf{v},\mathbf{w},\mathbf{e},\mathbf{r}} \quad & J_{PLS}(\mathbf{v},\mathbf{w},\mathbf{e},\mathbf{r}) = \gamma \sum_{i=1}^{n} e_i r_i - \frac{1}{2}\mathbf{v^T v} - \frac{1}{2}\mathbf{w^T w} \\ \text{such that} \quad & e_i = \mathbf{v^T}\varphi(\mathbf{x_i}) \quad i = 1,\ldots,n \\ & r_i = \mathbf{w^T}\varphi(\mathbf{y_i}), \quad i = 1,\ldots,n \end{aligned} \tag{3.50}$$

with hyperparameter $\gamma \in \mathbb{R}_0^+$. Introducing α_i, β_i as Lagrange multiplier parameters, the Lagrangian is written as

$$\begin{aligned} L(\mathbf{v},\mathbf{w},\mathbf{e},\mathbf{r};\boldsymbol{\alpha},\boldsymbol{\beta}) = \gamma \sum_{i=1}^{n} e_i r_i - \frac{1}{2}\mathbf{v^T v} - \frac{1}{2}\mathbf{w^T w} \\ - \sum_{i=1}^{n} \alpha_i (e_i - \mathbf{v^T}\varphi(\mathbf{x_i})) - \sum_{i=1}^{n} \beta_i (r_i - \mathbf{w^T}\varphi(\mathbf{y_i})). \end{aligned} \tag{3.51}$$

A given optimization problem has a corresponding dual formulation. The number of constraints in the original problem becomes the number of variables in the dual problem. One optimizes the Lagrangian with respect to the variables which gives the following optimality conditions:

$$\frac{\partial L}{\partial \mathbf{v}} = 0 \rightarrow \mathbf{v} = \sum_{i=1}^{n} \alpha_i \ \varphi(\mathbf{x_i}) = \Phi^T A \tag{3.52}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{n} \beta_i \ \varphi(\mathbf{y_i}) = \Phi^T B \tag{3.53}$$

$$\frac{\partial L}{\partial e_i} = 0 \rightarrow \gamma \ r_i = \alpha_i \quad i = 1,\ldots,n \tag{3.54}$$

$$\frac{\partial L}{\partial r_i} = 0 \rightarrow \gamma \ e_i = \beta_i \quad i = 1,\ldots,n \tag{3.55}$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \rightarrow e_i = \mathbf{v^T}\varphi(\mathbf{x_i}) \quad i = 1,\ldots,n \tag{3.56}$$

$$\frac{\partial L}{\partial \beta_i} = 0 \rightarrow r_i = \mathbf{w^T}\varphi(\mathbf{y_i}) \quad i = 1,\ldots,n. \tag{3.57}$$

By elimination of the variables $\mathbf{e}, \mathbf{r}, \mathbf{v}, \mathbf{w}$, we can simplify the dual problem further, which results into the very same system (3.47).

We point out that this KPLS solution can be considered in the optimization context as a special case of the 'regularized' KCCA variant which was proposed within the context of primal-dual LS-SVM formulations in [165]

$$\begin{aligned} K_{yy} \ \beta &= \lambda \ (\nu_1 K_{xx} + I) \ \alpha \\ K_{xx} \ \alpha &= \lambda \ (\nu_2 K_{yy} + I) \ \beta, \end{aligned} \tag{3.58}$$

if one chooses the parameters $\nu_1 = \nu_2 = 0$. The 'regular' KCCA was originally reported by [105] and [6], in an independent component analysis (ICA) context. Remark also that the subspace interpretation of regression in feature space allows to employ KCCA for the particular purpose of function estimation, in contrast to its classical use for yielding visualization axes or canonical angles.

In summary, these three closely related methods, by partial solution of the (generalized) eigenproblems (3.41), (3.47) and (3.45), deliver us useful subspaces spanned by eigenvectors $\{\alpha_i\}_{i=1}^m$ (ordered corresponding to nondecreasing values of the eigenvalues λ_i). Of course other subspaces may be chosen according to other principles. It is an entire subject on itself, often referred to as *exploratory projection pursuit*, that aims at finding linear combinations of variables so that data exhibit clusters with respect to those variables [63]. Finally the subpace allows to introduce the kernel matrix K in equation (3.17) in order to obtain a kernel regression model.

3.5 Illustrative examples

3.5.1 Sinc toy data set

To demonstrate some characteristics of these subspace regression methods, we apply it first to a sinc function. We started from a noiseless dataset. We considered a domain dataset of 100 equally-spaced points in the interval [-10,10]. The corresponding output values were centralized. We used a Gaussian kernel $\exp(-\|\mathbf{x_i} - \mathbf{x_j}\|_2^2/h^2)$ with width parameter $h = 1$ (and for kCCA always $\nu_1 = \nu_2 = 1$).

In Figure 3.1 the first three components of KPCA, KPLS and KCCA qualitatively show a good correlation with the target function. We show also the eigenspectrum of the components which indicate the needed number of components in order to represent the function. Non-equally-spaced sampling causes the components to be more irregular and oscillatory, while prediction will be less performant in undersampled regions and near boundaries. In Figure 3.2 we show an example of KCCA subspace regression on the same data, but with added Gaussian noise with standard deviation $\sigma = 0.2$. The other two methods give likewise component profiles and prediction results.

We compared for 100 trials on sinc data sets (noise added with $\sigma = 0.2$) the number of components m for the three methods. Parameters $m \in \{1, 2, .., 20\}$ and $h \in [0.01, 5]$ were determined by 10-fold cross-validation (CV). On average the three methods perform equally well in terms of mean square error (MSE) on an independent test data set, with comparable variance. When lowering the number of components further, the best bias-variance trade-off is for KCCA and then KPLS. From the boxplots in Figure 3.3 we see that KCCA needs on

average less components. This is also reflected in the non-increasing ordered eigenvalue spectrum that has large first eigenvalues, most quickly decaying for KCCA. As for the parameters ν_1 and ν_2 we may conclude that the regression result is fairly insensitive to ν_2, but that large values of ν_1 cause overfitting. The use of other kernels, like the polynomial or the sigmoidal kernel, did not produce such good results as the Gaussian kernel.

We compared for 100 randomizations on sinc data sets (noise added with $\sigma = 0.2$) with results shown in Table 3.3. The number of components $m \in \{1, 2, .., 30\}$ was determined by 10-fold cross-validation (CV), $h = 0.15$ was fixed. On average all methods perform equally well in terms of mean square error (MSE) on an independent test data set, with comparable variance. KPLS1 performs best in terms of lowest number of components, followed by KCCA and KPLSy. The distinct feature of KPLS1 is that in fact each time the data matrices in y space are deflated onto the components of the x space (instead of on previously found component in y space). The use of other kernels, like the polynomial or the sigmoidal kernel, did not produce better results. The results are similar to LS-SVMs for regression (kernel ridge regression or regularization network [56] with additional bias term) using the LS-SVMlab software http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

method	nr of components	Mean Square Error
KPCA	25 ± 4	0.3875 ± 0.0914
KPLS-SVD	19 ± 6	0.3888 ± 0.1014
KPLS-WA	20 ± 5	0.3801 ± 0.0863
KPLS-U	24 ± 5	0.3778 ± 0.0812
KPLSx	19 ± 6	0.3864 ± 0.0972
KPLSy	15 ± 7	0.3908 ± 0.0861
KPLS1	3 ± 1	0.3825 ± 0.0879
KCCA	11 ± 7	0.3903 ± 0.0445

Table 3.3: Comparison of KPLS variants and KCCA, which correspond to forms of subspace regression in RKHS with different choices of matrix A in (3.52). Shown are the number of selected components and test set performance for 100 randomizations on the sinc data set.

3.5.2 Boston housing data set

We also tested the methods on the larger, real-world dataset benchmark of Boston Housing data and observed again comparable performance. The Boston Housing data set [77] consists of 506 cases having $p = 13$ input variables. The aim is to predict the housing prices. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 400$ and a test set of size $n_t = 106$. From the literature [156] we adopted the commonly used value of $h^2 = 0.15p = 3.9$ as fixed Gaussian kernel width for comparison reasons. We performed 10-fold CV to estimate the number of components $m \in$

$\{1, 2, ..., 300\}$. After CV the best model was evaluated on an independent test set. The randomisation trials were repeated 10 times. See table 3.4 for details.

On average all methods perform equally well in terms of mean square error (MSE) on an independent test data set, with comparable variance. The results are similar and not significantly different from LS-SVMs for regression (kernel ridge regression or regularization network with additional bias term) using the LS-SVMlab software http://www.esat.kuleuven.ac.be/sista/lssvmlab/. KPLS1 performs best in terms of lowest number of components, followed by KPLS-U and KPLS-WA. The particular deflation scheme in the KPLS1 model is responsible for this, and therefore this model may be preferred when dealing with large datasets. In the next chapter we will present additional experiments and at the same time show a more efficient approach to extend the capabilities of the subspace regression model when data sets are large.

method	nr of components	Mean Square Error
KPCA	220 ± 35	0.2292 ± 0.0525
KPLS-SVD	272 ± 14	0.2327 ± 0.0479
KPLS-WA	209 ± 24	0.2701 ± 0.0467
KPLS-U	191 ± 18	0.2166 ± 0.0832
KPLSx	234 ± 11	0.2864 ± 0.0395
KPLSy	216 ± 16	0.2108 ± 0.0681
KPLS1	176 ± 38	0.2027 ± 0.0879
KCCA	245 ± 30	0.2230 ± 0.0445

Table 3.4: Boston Housing data set. Obtained parameters and independent test set performance of the various subspace regression methods using a Gaussian kernel with $h = \sqrt{3.9}$.

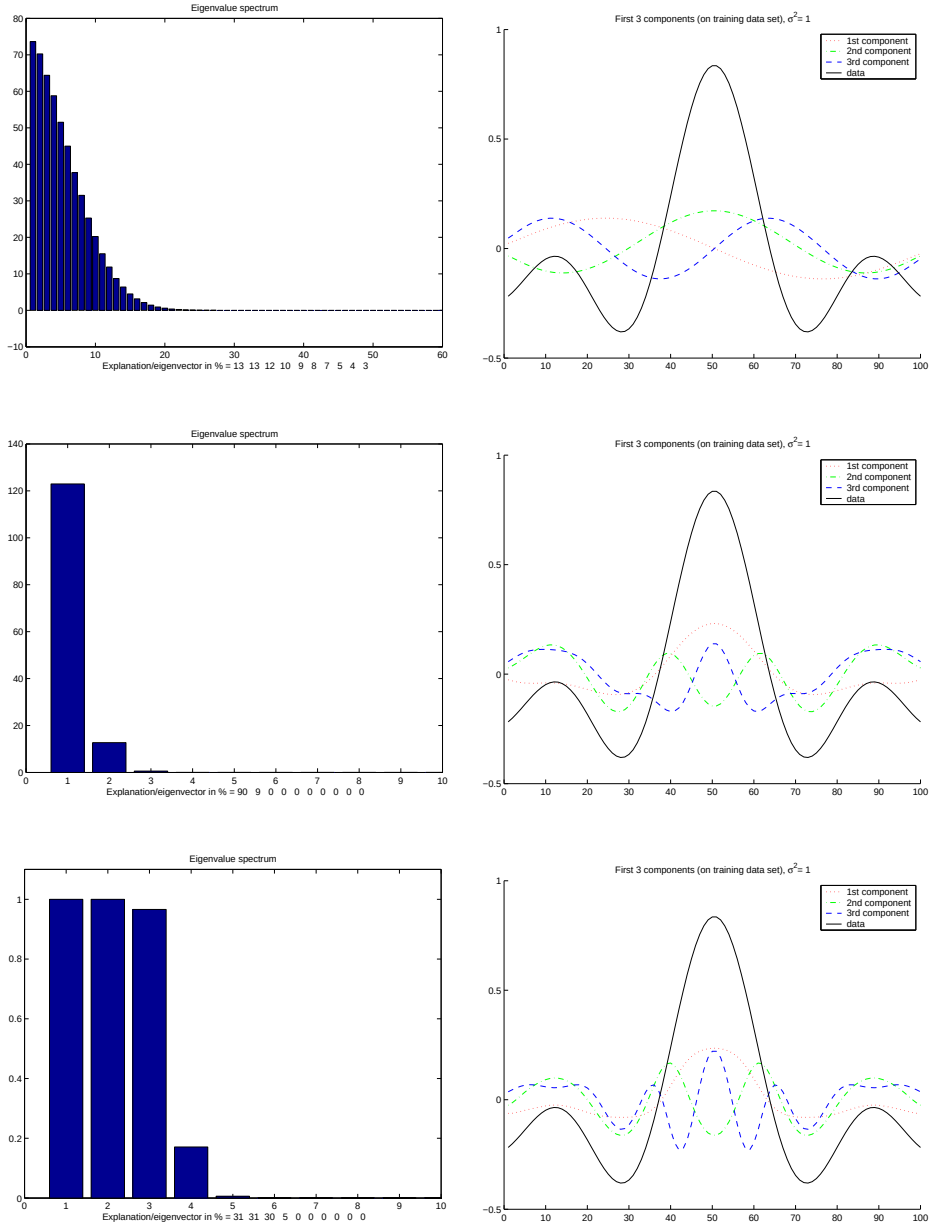


Figure 3.1: Eigenvalue spectra (left) and shape of the first 3 components (right) by KPCA (upper), KPLS (middle), KCCA (bottom) for a noiseless sinc function. Notice how KPLS and KCCA need fewer components than KPCA. The first few feature components contain clearly the main characteristics of the underlying function.

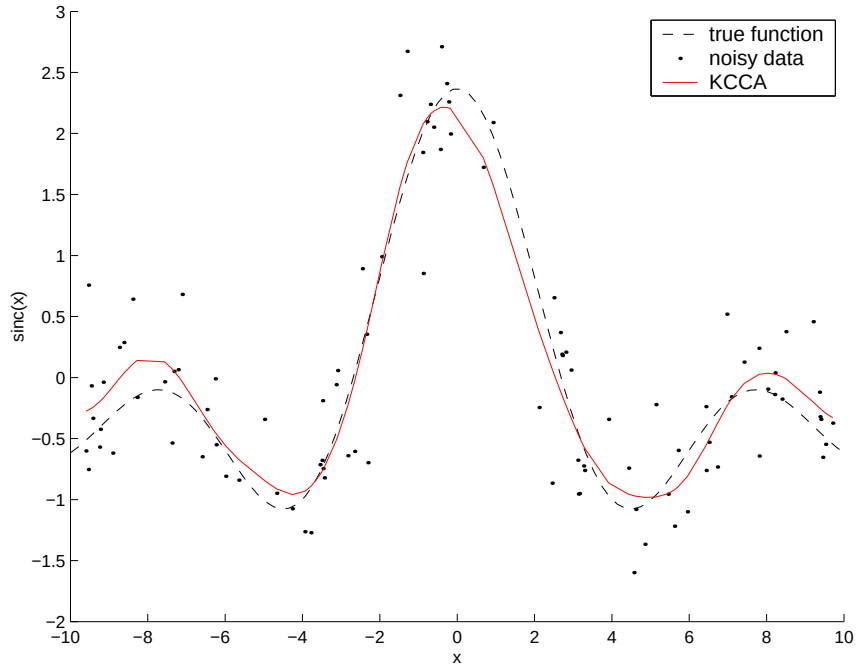


Figure 3.2: Example of prediction obtained by KCCA subspace regression on a noisy sinc ($\sigma = 0.2$, $m = 4$, $h = 1$). Good prediction is achieved, considering the noise.

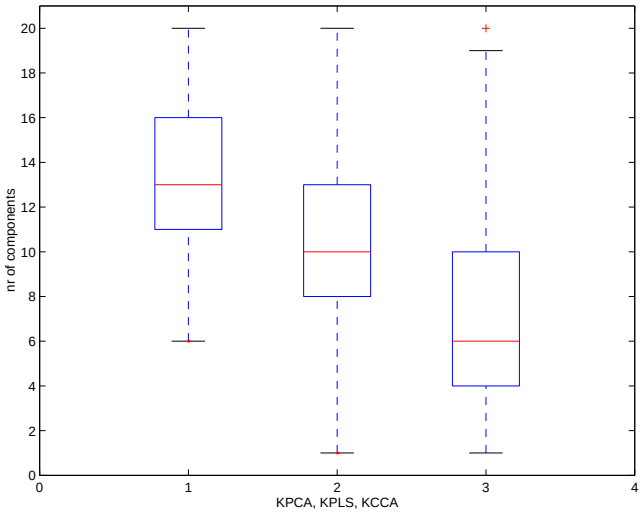


Figure 3.3: Example of boxplots of number of components for KPCA, PLS-SVD, KCCA respectively, on a noisy sinc ($\sigma = 0.2$ and fixed $h = 1$).

3.5.3 Chemical data set

We applied the subspace kernel regression models on data from a process in the chemical industry. The author took part in a project [48] between 3 partners: ESAT-KULeuven, ISMC NV Haasrode and the chemical company BASF NV in Antwerp. The goal was the 'Design of a softsensor for the chemical process industry' based on state-of-the-art data mining technology. With softsensor one means a model that replaces the 'hard' sensor in a monitor process in order to decrease its (maintenance) cost, to lower the frequency of timeconsuming and costly lab-analyses, to obtain a higher production quality and guarantee more security.

The provided data set consisted of 88 input variables (like temperature, pressure, concentrations, etc.), 1 output (a concentration ratio of 2 chemical substances), with a size of circa 50.000 data points (25% of the data was kept as a validation set).

Typical data mining preparation steps have been performed like data corrections, Matlab format conversion, filtering, outlier detection, peak shaving, trend elimination, transforming inputs and rough input selection. The most difficult task, however, was the characterisation of the precise nature of the underlying process. Is the system static or dynamic, is the behaviour linear or non-linear? In each of the possible cases there are each time alternative modelling techniques available (e.g. Ordinary least squares (OLS), Subspace models [179], recurrent Neural networks, Support Vector Machines [35], etc.). Experiments with each of the model classes have been performed. Robustness and accuracy were the most important industrially desired requirements. Linear models with 33 inputs were shown to be very sensitive to overfitting with large parameter variance. Experiments with input selection methods showed that the number of inputs could be brought further down to 9 via Automatic Relevance Determination (ARD) [114]). One of the best models was delivered by KCCA based kernel regression, with s=64 components after 10-fold cross validation on the number of components and the RBF kernel parameter h^2=0.46, of which a visual plot is shown in Figure 3.4 .

3.5.4 Electrical Power data set

We applied the subspace kernel regression models on data from a process in the energy industry. In a second project [47] a cooperation was set up with the Belgian network administration company (ELIA NV) that controls energy transport between producer and consumer. The goal was the 'Modelling, scenario and risk analysis of peak power capacities in the Belgian high-voltage electricity network'. The knowledge extraction process had to deliver especially explicative information.

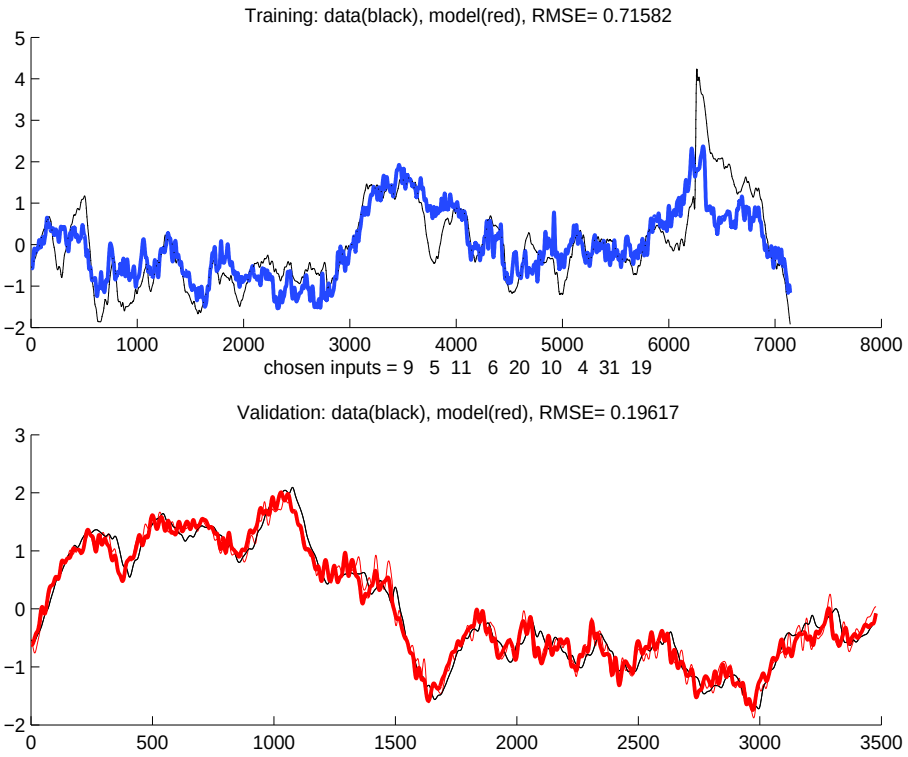


Figure 3.4: A demonstration of a best result -with the KCCA subspace regression in RKHS- on a real-life chemical process data set from industry. The thin line is the output signal, the thick line is the model on the training set (top) and the prediction on the validation set (bottom). For clarity of the picture we only showed each time a subset of the bigger data set.

A database was provided that consisted of the electricity consumption in Brabant (a single power load output) measured at 16 main transformator substations, over the period of 1996 till 2001. For the inputs we received 23 input variables (like temperature, cloudiness degree, and socio-economical variables). The sampling rate was 4 times per hour, which gave a data set of 175.392 points (20%, the last year, has been kept as a validation set).

In a first phase we built a data structure (data aggregation, vizualization, data correction, data conversion to Matlab, normalization, rough input selection). In a next phase explorative static and dynamic linear models were constructed, by ordinary Least Squares (OLS) and AutoRegressive eXogeneous input models (ARX). ARX is a dynamical model that takes into account information from the past as inputs and is typically applied on stationary time invariant systems with white noise. A trend elimination and removal of periodicity was performed in

order to obtain stationary data, such that a linear model with root mean square error $R^2 = 93\%$ explication was reached. To compensate for the (nonlinear) temperature dependencies, the temperature variable needed to be non-linearly transformed. In Figure 3.5 we illustrate the use of KPLS based kernel regression on the estimate of the power load versus temperature. With 10-fold cross-validation we reached a best fit with $s = 4$ components and width $h^2 = 9$ for a RBF kernel.

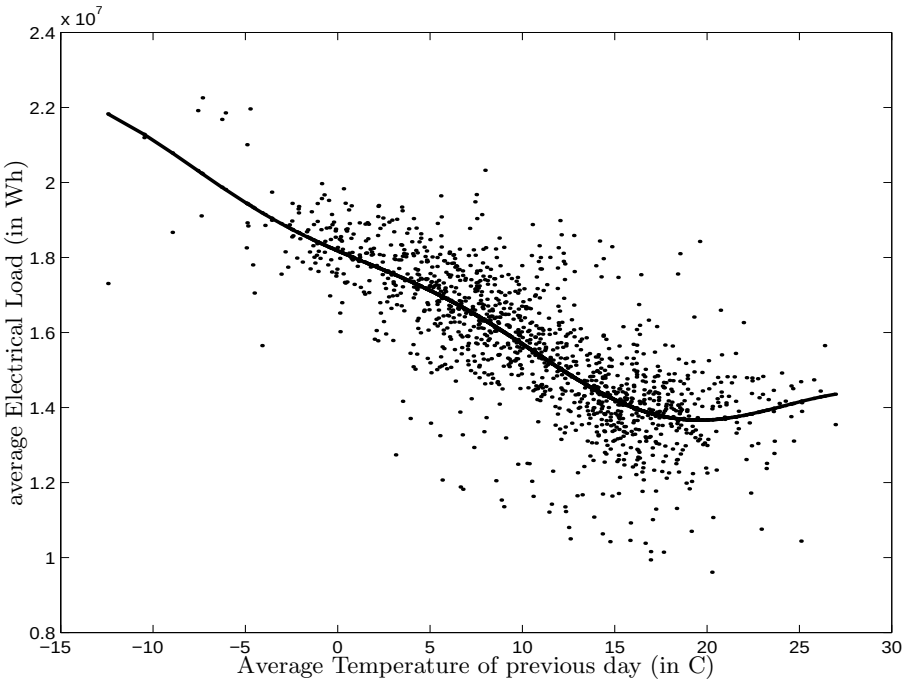


Figure 3.5: Illustration of nonlinear modelling with KPLS based kernel regression on a real-life data set in the energy sector. Shown is the average electrical load versus the temperature of the previous day. The model was used in subsequent analysis. The result is not different from the one obtained by LS-SVM.

3.6 An application in Subspace Identification

Next to delivering a useful subspace for nonlinear regression, the above introduced covariance methods yield also a useful subspace in the context of Subspace Identification models. In [71] we presented a method for the nonlinear identification of Hammerstein-Wiener systems, using Kernel Canonical Correlation Analysis. We will consider here mainly the KCCA aspects of the application.

3.6.1 Introduction

Subspace identification algorithms were originally developed by Van Overschee and De Moor [178, 179]. They are an important parametric paradigm in the field of system identification [111] and are widely known and appreciated for their quick and reliable estimation of *linear* models based on available input/output measurements. Much of the reliability of subspace identification algorithms is attributed to the fact that a model is obtained, solely by using numerically reliable matrix and vector manipulations such as projections and singular value decompositions. This is in contrast with the more classical predictor error methods which often involve minimization of a non-convex cost-function, with no guarantee that the obtained local minimum yields a good model.

Although subspace identification algorithms are fast and robust, a major drawback is that their use is mostly restricted to the class of linear systems. Some attempts to extend the use of subspace identification algorithms to nonlinear systems have been made in the past for general nonlinear systems [106, 186], or more restricted model structures such as bilinear models [58, 124, 187], Wiener models [189], and Hammerstein models [192, 8].

In particular, we consider the extension of the classical subspace intersection algorithm [126] to Hammerstein-Wiener systems in state-space form:

$$\begin{cases} x_{t+1} &= Ax_t + Bf(u_t), \\ g^{-1}(y_t) &= Cx_t + Du_t, \end{cases} \tag{3.59}$$

where $u_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^l$ are the input and output at time t and $x_t \in \mathbb{R}^n$ denotes the state. $f : \mathbb{R}^m \to \mathbb{R}^m$ and $g : \mathbb{R}^l \to \mathbb{R}^l$ are static nonlinear maps with g such that g^{-1} exists for all possible outputs of the system. When f and g would be identity maps, a classical *linear* state-space model would be obtained. When only f is present, one speaks of a Hammerstein system; when only g is present, one speaks of a Wiener system.

The proposed extension is obtained by replacing the linear CCA-step, used for the estimation of the state by a nonlinear kernel CCA (KCCA) approximator. In a second step, the system matrices A, B, C and D and the nonlinearities

f and g are obtained from the solution of a Least Squares - Support Vector Machines (LS-SVMs) regression problem, which was already originally devised and extensively described in an extension of the N4SID [72] identification algorithm towards Hammerstein systems.

To our knowledge, the available literature on the identification of Hammerstein-Wiener models is rather sparse. In [8], a scheme for the identification of SISO Hammerstein-Wiener systems is developed based on the idea of overparametrization [26]. However, in this scheme a very specific model structure is assumed, limiting its practical applicability. Based on [8], a more general blind approach for the identification of SISO systems was proposed in [9]. An identification method for Hammerstein-Wiener MIMO systems was proposed in [33, 32] but imposes certain restrictions on the inputs and is iterative in nature. Other contributions such as [57, 201] are limited to SISO systems and/or iterative in nature.

In contrast to these methods, a clear advantage of the proposed technique is that it does not rely on restrictive assumptions on the inputs such as white noise or periodicity, that it is non-iterative in nature, and that it can conveniently be applied to MIMO systems. Furthermore, other than the invertibility of g and a certain degree of smoothness, no specific restrictions are imposed on the nonlinear maps f and g.

As a general rule in this treatment, lowercase symbols will be used to denote column vectors. Uppercase symbols are used for matrices. Elements of matrices and vectors are selected using Matlab standards, e.g. $A(i,j)$ denotes the ij^{th} entry of a matrix A, and $A(:,i)$ symbolizes the i^{th} column of the same matrix. Estimates for a parameter x will be denoted by $\hat{x}$.

3.6.2 The subspace intersection algorithm

We will briefly review the basic ingredients of the subspace intersection algorithm for linear systems. The subspace algorithm considered in this paper was originally proposed in [46, 126] and is largely based on the idea that the state of a linear or nonlinear model can be considered as the minimal intersection between past and future measurement data [106].

The subspace intersection algorithm identifies models of the form

$$\left\{ \begin{array}{rcl} x_{t+1} & = & Ax_t + Bu_t, \\ y_t & = & Cx_t + Du_t, \end{array} \right. \tag{3.60}$$

for all $t = 0, \ldots, N-1$ and with $u_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^l$ the input and output at time t. $x_t \in \mathbb{R}^n$ denotes the state. Based on a finite set of training data $\{u_t, y_t\}, t = 0, \ldots, N-1$, intersection algorithms are concerned with finding an estimate for the model order n of the system (3.60), and estimates for the

system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$ and $D \in \mathbb{R}^{l\times m}$ up to a similarity transformation. Block Hankel matrices play an important role in these algorithms. The input block Hankel matrices are defined as

$$U_{0|2i-1} := \left[\begin{array}{ccccc} u_0 & u_1 & u_2 & \dots & u_{j-1} \\ u_1 & u_2 & u_3 & \dots & u_j \\ \vdots & \vdots & \vdots & & \vdots \\ u_{i-1} & u_i & u_{i+1} & \dots & u_{i+j-2} \\ \hline u_i & u_{i+1} & u_{i+2} & \dots & u_{i+j-1} \\ u_{i+1} & u_{i+2} & u_{i+3} & \dots & u_{i+j} \\ \vdots & \vdots & \vdots & & \vdots \\ u_{2i-1} & u_{2i} & u_{2i+1} & \dots & u_{2i+j-2} \end{array}\right] := \left[\begin{array}{c} U_p \\ \hline U_f \end{array}\right] \in \mathbb{R}^{(2i)m\times j},$$

with i and j user defined indices such that $2i + j - 1 = N$. The output block Hankel matrices $Y_p, Y_f \in \mathbb{R}^{il\times j}$ are defined in a similar way. The joint past and future W_p and W_f are defined as

$$W_p := \begin{bmatrix} U_p \\ Y_p \end{bmatrix}, \quad W_f := \begin{bmatrix} U_f \\ Y_f \end{bmatrix}.$$

Finally,

$$\begin{aligned} X_p &:= \begin{bmatrix} x_0 & x_1 & \dots & x_{j-1} \end{bmatrix}, \\ X_f &:= \begin{bmatrix} x_i & x_{i+1} & \dots & x_{i+j-1} \end{bmatrix}, \end{aligned}$$

are introduced as finite state sequences of length j. The main reasoning behind subspace intersection algorithms follows from the fact that under the assumptions that:

1. the input u_k is persistently exciting of order $2i$, i.e. the input block Hankel matrix $U_{0|2i-1}$ is of full rank,
2. The intersection of the row space of U_f (the future inputs) and the row space of X_p (the past states) is empty,

the following relation holds:

$$\mathrm{row}(\mathrm{X_f}) = \mathrm{row}(\mathrm{W_p}) \cap \mathrm{row}(\mathrm{W_f}).$$

Hence, the order of the system and a realization of the state can be obtained from the intersection of past and future. Mathematically, this step is typically performed using a CCA algorithm, and retaining the canonical variates corresponding to canonical angles equal to 1. Once the state is known, extraction of A, B, C and D is straightforward.

Without going into further theoretical details of the subspace intersection algorithm (interested readers are referred to [46, 126]), we summarize here a practical implementation that will be used towards the Hammerstein-Wiener model extension:

1. Perform canonical correlation analysis on W_p and W_f:

$$\begin{array}{rcl} W_p W_f^T V_f & = & W_p W_p^T V_p \Lambda, \\ W_f W_p^T V_p & = & W_f W_f^T V_f \Lambda, \end{array} \tag{3.61}$$

   with Λ a diagonal matrix containing the canonical correlations.

2. Determine the order n from the number of canonical correlations equal to one. Retain X_f as the n corresponding canonical variates in W_p.

$$X_f = V_p(:, 1:n)^T W_p. \tag{3.62}$$

3. Extract A, B, C and D from the overdetermined set of linear equations:

$$\begin{bmatrix} X_f(:, 2:j) \\ Y_{i|i}(:, 1:j-1) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_f(:, 1:j-1) \\ U_{i|i}(:, 1:j-1) \end{bmatrix}. \tag{3.63}$$

3.6.3 Hammerstein-Wiener subspace intersection

We will discuss how to extend the linear intersection algorithm towards a nonlinear setting using a variation on the theme of kernel CCA.

Introducing the static nonlinearities

Equation (3.60) is transformed into a Hammerstein-Wiener system by introducing two static nonlinearities $f : \mathbb{R}^m \to \mathbb{R}^m$ and $g : \mathbb{R}^l \to \mathbb{R}^l$. With this definition for the nonlinearities, and assuming that $g : \mathbb{R}^l \to \mathbb{R}^l$ is such that g^{-1} exists for all possible outputs of the system, equation (3.60) is rewritten as:

$$\begin{cases} x_{t+1} = Ax_t + Bf(u_t), \\ g^{-1}(y_t) = Cx_t + Df(u_t). \end{cases} \tag{3.64}$$

As mentioned in Section 3.6.2, a CCA algorithm could be used to extract the state x if $f(u)$ and $g^{-1}(y)$ were known. The state is then obtained from

$$\begin{array}{rcl} \mathcal{F}(W_p)\mathcal{F}(W_f)^T V_f & = & \mathcal{F}(W_p)\mathcal{F}(W_p)^T V_p \Lambda, \\ \mathcal{F}(W_f)\mathcal{F}(W_p)^T V_p & = & \mathcal{F}(W_f)\mathcal{F}(W_f)^T V_f \Lambda, \end{array}$$

where $\mathcal{F}(W_p)$ is defined as follows

$$\mathcal{F}(W_p) := \begin{bmatrix} f(u_0) & f(u_1) & \dots & f(u_{j-1}) \\ \vdots & \vdots & & \vdots \\ f(u_{i-1}) & f(u_i) & & f(u_{i+j-2}) \\ g^{-1}(y_0) & g^{-1}(y_1) & \dots & g^{-1}(y_{j-1}) \\ \vdots & \vdots & & \vdots \\ g^{-1}(y_{i-1}) & g^{-1}(y_i) & & g^{-1}(y_{i+j-2}) \end{bmatrix}$$

with an equivalent definition for $\mathcal{F}(W_f)$. However, because $f(u)$ and $g^{-1}(y)$ are unknown, another approach is required to extract the state. A well-suited technique to fulfill this task is kernel CCA, a nonlinear extension of CCA, which will be treated in the following subsection. Once the state is known, $f(u)$ and $g^{-1}(y)$ will be estimated in a second step using LS-SVM regression, according to [72].

Introducing the kernel

To extract a state of a *nonlinear* dynamical system, a nonlinear extension of CCA is employed, known as kernel CCA or KCCA [105, 6]. In kernel methods [165] the available data are mapped into a high-dimensional feature space of dimension n_H, where classical CCA is applied. The nonlinearity is condensed in the transformation, which is represented by feature maps $\varphi^u : \mathbb{R}^m \to \mathbb{R}^{n_H}$ and $\varphi^y : \mathbb{R}^l \to \mathbb{R}^{n_H}$. Using the mapped past data points $\varphi^u(u)$ and $\varphi^y(y)$, one constructs a feature matrix of the input block Hankel matrix (3.61)

$$\Phi_p := \Phi(W_p) := \begin{bmatrix} \varphi^u(u_0) & \varphi^u(u_1) & \dots & \varphi^u(u_{j-1}) \\ \vdots & \vdots & & \vdots \\ \varphi^u(u_{i-1}) & \varphi^u(u_i) & & \varphi^u(u_{i+j-2}) \\ \varphi^y(y_0) & \varphi^y(y_1) & \dots & \varphi^y(y_{j-1}) \\ \vdots & \vdots & & \vdots \\ \varphi^y(y_{i-1}) & \varphi^y(y_i) & & \varphi^y(y_{i+j-2}) \end{bmatrix} \in \mathbb{R}^{2i(m+l)n_H \times j}, \tag{3.65}$$

with a similar definition for $\Phi_f := \Phi(W_f)$.

In the kernel method context the feature maps are assumed to be associated with kernels $k^u : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R} : (u_s, u_t) \mapsto k^u(u_s, u_t)$ and $k^y : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R} : (y_s, y_t) \mapsto k^y(y_s, y_t)$. In the dual space one will now instead perform computations with (elements of) the $j \times j$ kernel gram matrices $K_p := \Phi_p^T \Phi_p$ and $K_f := \Phi_f^T \Phi_f$. With the feature matrix introduced in Eq. (3.65) we adopted a so-called *component-wise kernel*, originally suggested in [135, 134].

KCCA by the kernel trick

For reasons of clarity of presentation we adopt here a formal introduction into the KCCA algorithm as it was initially presented in [105, 6]. By mapping the elements of W_p and W_f the CCA problem in feature space becomes:

$$\begin{aligned} \Phi_p \Phi_f^T \; V_f &= \Phi_p \Phi_p^T \; V_p \Lambda, \\ \Phi_f \Phi_p^T \; V_p &= \Phi_f \Phi_f^T \; V_f \Lambda. \end{aligned} \tag{3.66}$$

Remark that the coefficient matrices V_p, V_f are elements of $\mathbb{R}^{2i(m+l)n_H \times 2i(m+l)n_H}$ where n_H can be potentially inifinite-dimensional, which is not practical.

However these matrices are restricted to the subspace spanned by the mapped data by redefining:

$$V_p^* := \Phi_p V_p,\ V_f^* := \Phi_f V_f, \tag{3.67}$$

and the first and second equation of (3.66) are left multiplied by Φ_p^T and Φ_f^T, respectively, we obtain:

$$\begin{aligned} K_p K_f\ V_f^* &= K_p K_p\ V_p^* \Lambda, \\ K_f K_p\ V_p^* &= K_f K_f\ V_f^* \Lambda. \end{aligned} \tag{3.68}$$

Assuming that K_p and K_f are invertible, which can be shown to be automatically satisfied if complex nonlinear kernels such as the RBF are involved, this can further be reduced to

$$\begin{aligned} K_f\ V_f^* &= K_p\ V_p^*\ \Lambda \\ K_p\ V_p^* &= K_f\ V_f^*\ \Lambda, \end{aligned} \tag{3.69}$$

which is the classical form of the KCCA algorithm as presented in [105, 6]. A disadvantage of this KCCA version is the fact that the used kernel derivations do not contain regularization leaving the possibility of a severe overfitting of the nonlinearities involved.

The KCCA version proposed in [165] is formulated using a support vector machine approach [183] with primal-dual optimization problems and an additional centering of the datapoints in feature space. Regularization is thereby incorporated within the primal formulation in a well-established manner leading to numerically better conditioned solutions. Without going into the details of this algorithm, we state here the final generalized eigenvalue problem:

$$\begin{aligned} K_f^c\ V_f^* &= \left(K_p^c + \frac{1}{\gamma} I_j\right)\ V_p^*\ \Lambda, \\ K_p^c\ V_p^* &= \left(K_f^c + \frac{1}{\gamma} I_j\right)\ V_f^*\ \Lambda, \end{aligned}$$

where $\mu_p = (1/j)\sum_{s=1}^{j} \Phi_p(:,s)$ and $\mu_f = (1/j)\sum_{s=1}^{j} \Phi_f(:,s)$ are the so-called mean centers of the mapped past and future, and with

$$\begin{aligned} K_p^c &= (\Phi_p - 1_j^T \otimes \mu_p)^T(\Phi_p - 1_j^T \otimes \mu_p), \\ K_f^c &= (\Phi_f - 1_j^T \otimes \mu_f)^T(\Phi_f - 1_j^T \otimes \mu_f), \end{aligned}$$

the centered kernels. The tuning parameter γ controls the amount of regularization. Comparison with the derived result without centering yields that $K_f^c = M_c K_f M_c$ with $M_c = (I_j - (1/j)1_j 1_j^T)$ [155].

Obtaining the state and state-space estimates

Thus by solving a generalized eigenvalue problem in the dual space, one can find the correlations and the nonlinear canonical variates, gathered resp. in the

KCCA estimates $\widehat{\Lambda}$ and $\widehat{V}_p^*$ and $\widehat{V}_f^*$. From the number of canonical correlations equal to one, we estimate the order n. The estimated state is obtained as the n corresponding linear combinations of the centered variates in Φ_p, which comes down to:

$$\widehat{X}_f = (\hat{V}_p^*(1:n,:))^T K_p^c.$$

In a second step one estimates A, B, C, D and the nonlinear function f and g. Basically, the linear system (3.63), formulated with features and with the estimated state inserted, is being rewritten such that equations of the form $f(\cdot) = w^T\varphi(\cdot)$ are formed. By applying then regularization on the w, a reliable estimate of the parameters can be obtained by LS-SVM primal-dual optimization. By solving regularized linear systems in dual space, the state space matrices are directly recovered and the nonlinear f and g are simply kernel expansions. As our scope was to demonstrate a useful application of the KCCA algorithm, we refer for further computational details to [71, 70]. With some examples we demonstrate the effectivity of the proposed approach.

3.6.4 Illustrative examples

A SISO system

Consider the following artificial linear Single Input Single Output (SISO) system which belongs to the class of Hammerstein-Wiener models:

$$y = g\left(\frac{B(z)}{A(z)}f(u)\right), \tag{3.70}$$

with A and B polynomials in the forward shift operator z where

$$\begin{aligned} B(z) &= z^6 + 0.8z^5 + 0.3z^4 + 0.4z^3, \\ A(z) &= (z - 0.98e^{\pm i})(z - 0.98e^{\pm 1.6i})(z - 0.97e^{\pm 0.4i}). \end{aligned} \tag{3.71}$$

The input- and output-nonlinearities are given by

$$\begin{aligned} &f : \mathbb{R} \to \mathbb{R} : f(u) = \mathrm{sinc}(u), \\ &g : \mathbb{R} \to \mathbb{R} : g(y) = \begin{cases} y/12, & y \leq 0, \\ \tanh(\mathrm{y}/4), & \mathrm{y} > 0. \end{cases} \end{aligned} \tag{3.72}$$

Two datasets were generated from this system with the inputs $u_k \sim \mathcal{N}(0, 2)$ white Gaussian noise sequences for $t = 0, \ldots, N-1$ with $N = 500$. Although the intersection algorithm is in principal only designed for deterministic systems, 5% of zero mean white Gaussian noise was added to the outputs in both datasets.

The first dataset was used to train the model, the second one was used to tune the model (hyper)parameters. Only the less critical number of block-rows in

the Hankel matrices was fixed beforehand at 10, a common choice in subspace algorithms. For k^u and k^y, both RBF kernels were chosen with $\sigma_u = 1$ and $\sigma_y = 0.5$ respectively. The KCCA regularization parameter was $\gamma = 1$ and LS-SVM regularization parameters $\gamma_u = 1$, $\gamma_y = 315$. The canonical correlation values dropped a bit too smooth, but a cutoff of order 6 was visually justified.

The obtained linear system using our KCCA-based algorithm (in combination with LS-SVM) is compared with the true system in Figure 3.6. Note that the first two resonances are nicely caught by the model, whereas the less energetic third resonance is not found. In itself not a bad result considering that (*i*) the used system has already been shown [72] to be a difficult one to estimate with classical techniques such as ARX, (*ii*) that the intersection algorithm is in essence a deterministic algorithm.

Further improvement is (for example) possible by rerunning a linear subspace algoritm such as the Past Outputs-Multivariable Output Error State SPace (PO-MOESP) [188] on the estimated inputs $\hat{f}(u)$ and outputs $\hat{g}^{-1}(y)$ of the linear system. That such a second step indeed improves the accuracy of the estimate of the linear model is shown in Figure 3.7.

The obtained nonlinear functions $\hat{f}$ and $\hat{g}$ evaluated on the validation inputs and outputs, are compared with the true functions f and g in Figure 3.8. As can be seen in the figure, the obtained estimates are quite reliable.

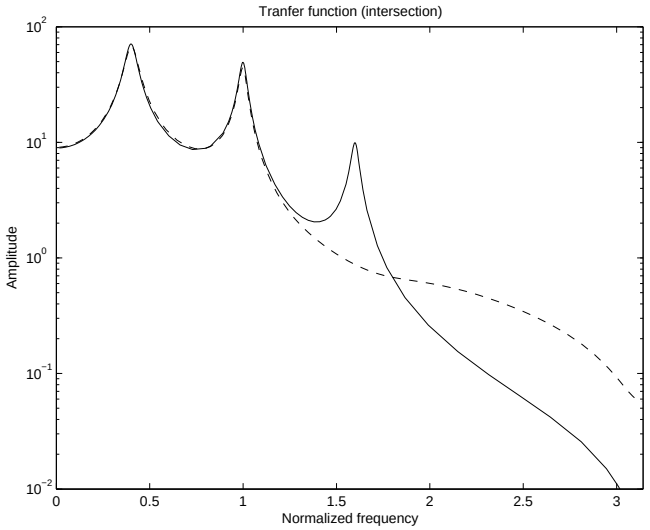


Figure 3.6: Estimated Transfer functions (dashed) for the SISO example described in Section (3.6.4) using our KCCA-based intersection-algorithm. The true Transfer function is displayed in solid. The main dynamics have been well-captured in the two first energy peaks, the third peak is to some extent hidden by the noise.

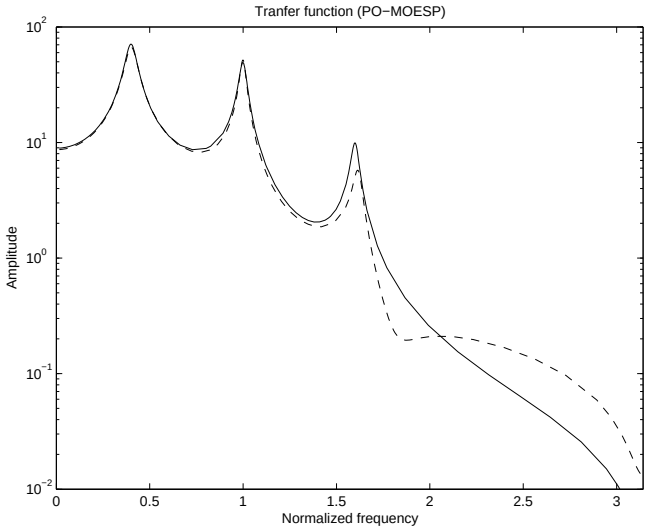


Figure 3.7: Estimated Transfer functions (dashed) for the SISO example described in Section 3.6.4 using a PO-MOESP after using the KCCA based intersection algorithm and estimation of the functions f and g. The true Transfer function is displayed in solid. As one can notice, also the dynamics of the third energy peak have been captured reliably, with a minimal amount of extra effort.

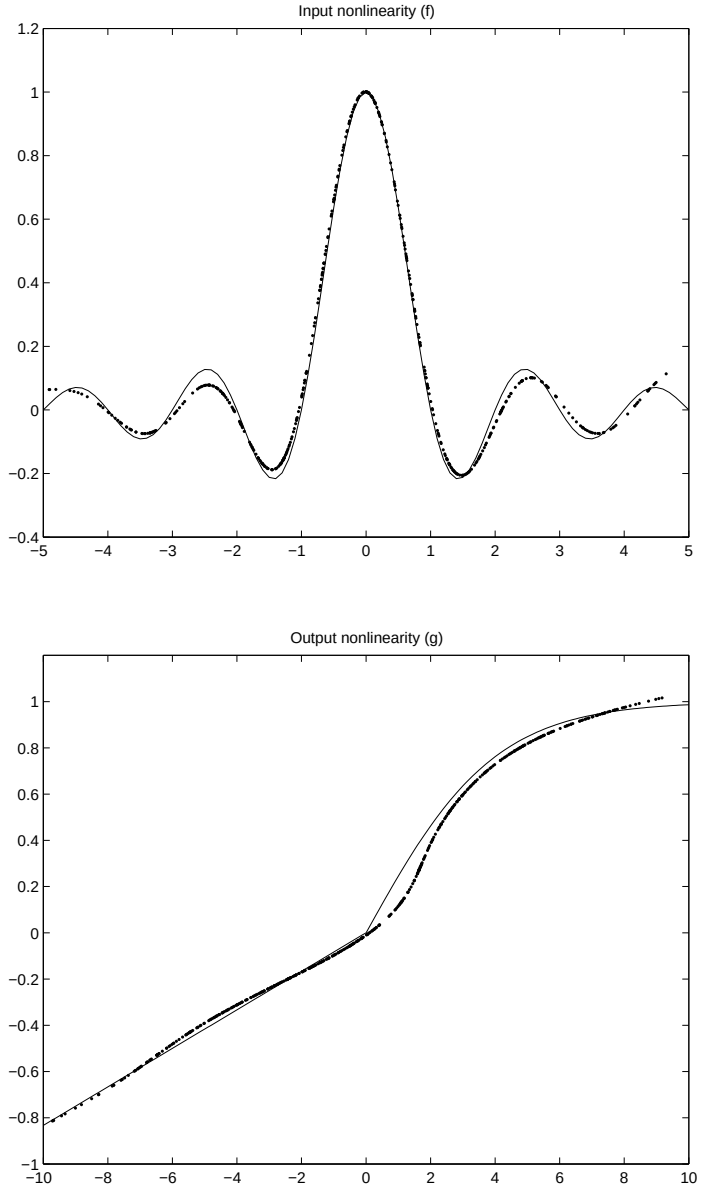


Figure 3.8: Estimated input nonlinearity f and output nonlinearity g evaluated on the validation inputs and outputs (dots) compared with the true nonlinearities (solid line) for the SISO example described in Section 3.6.4. The estimates are satisfactory.

A MIMO system

In a second example, the proposed identification method was applied to a 2×2 purely deterministic Multiple Input Multiple Output (MIMO) Hammerstein system with a static input-nonlinearity involving saturation and a saddle point. The system is given as:

$$y = \begin{bmatrix} \frac{b_1(z)}{a_1(z)} & \frac{b_2(z)}{a_1(z)} \\ \frac{b_1(z)}{a_2(z)} & \frac{b_2(z)}{a_2(z)} \end{bmatrix} f(u) + \begin{bmatrix} \frac{1}{a_1(z)} \\ \frac{1}{a_2(z)} \end{bmatrix} e \tag{3.73}$$

with elements specified as

$$\begin{aligned} a_1(z) &= (z - 0.98e^{\pm i})(z - 0.98e^{\pm 1.6i})(z - 0.97e^{\pm 0.4i}), \\ a_2(z) &= (z - 0.97e^{\pm 0.7i})(z - 0.98e^{\pm 1.4i})(z - 0.97e^{\pm 2.3i}), \\ b_1(z) &= z^6 + 0.8z^5 + 0.3z^4 + 0.4z^3, \\ b_2(z) &= z^6 + 0.9z^5 + 0.7z^4 + 0.2z^3, \\ f(u) &= \begin{bmatrix} -\arctan(u(1))\arctan(u(2)) \\ \arctan(u(1)) - \arctan(u(2)) \end{bmatrix}. \end{aligned} \tag{3.74}$$

A two-component zero mean white Gaussian input sequence u with length 500 and standard deviation 1 was generated and fed into the system (3.73). Based on u and the obtained output y, estimates for the linear system and f are obtained using the proposed Hammerstein-Wiener identification algorithm, whereby an RBF kernel was chosen for k^u and a *linear kernel* for k^y to effectively limit the Hammerstein-Wiener algorithm to Hammerstein systems.

As in the SISO example, the number of block-rows in the Hankel matrices was chosen equal to 10. The hyperparameters were again obtained by evaluation on a validation set and chosen as $\sigma_u = 1$, $\gamma = 0.1$ and $\gamma_u = \gamma_y = 1$. The order was easily found to be 12 from an inspection of the canonical correlations in the kernel CCA step (see Figure 3.9).

The results of a simulation on an independent test-set using the obtained model is shown in Figure 3.10 for the first component of the output and Figure 3.11 for the second component of the output. Also available in the figure are the results of a classical linear PO-MOESP subspace estimator which are clearly inferior than those obtained using the Hammerstein-Wiener approach.

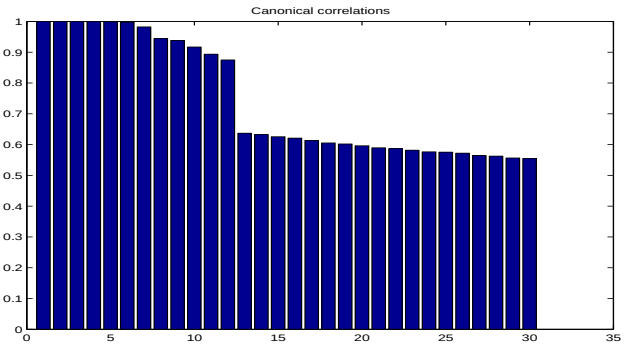


Figure 3.9: Canonical correlations obtained using kernel CCA on a twelfth order MIMO Hammerstein system described in Section 3.6.4. The order of the system is clearly seen to be equal to 12.

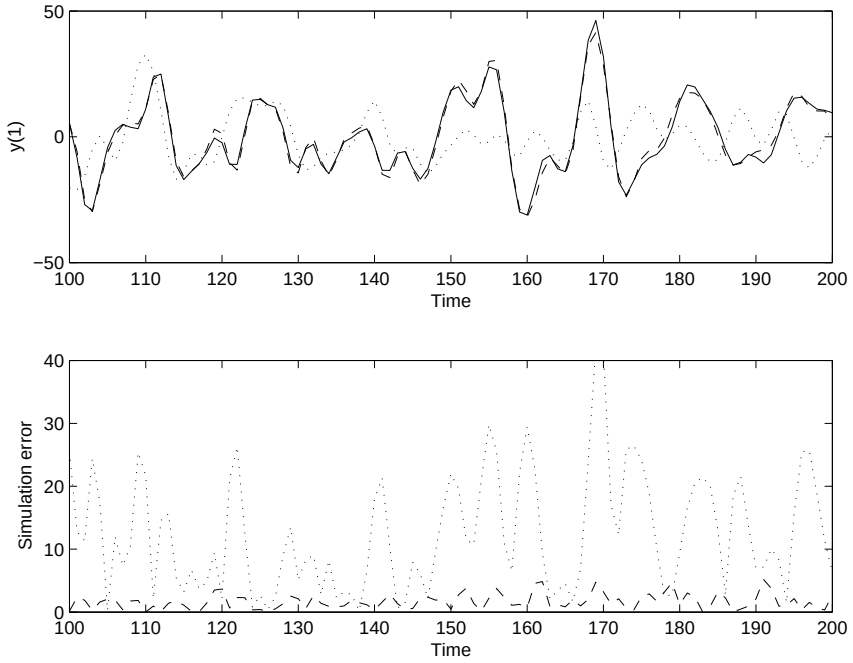


Figure 3.10: Simulation on an independent test-set of the first output of a twelfth order MIMO Hammerstein model described in Section 3.6.4 using a KCCA-based LS-SVM Hammerstein-Wiener estimator (dashed line) and a linear PO-MOESP subspace estimator (dotted line). The true output is depicted with a solid line. All simulations are initialized with $x_0 = 0$. The error between the estimated and the true output is also shown in the Figure.

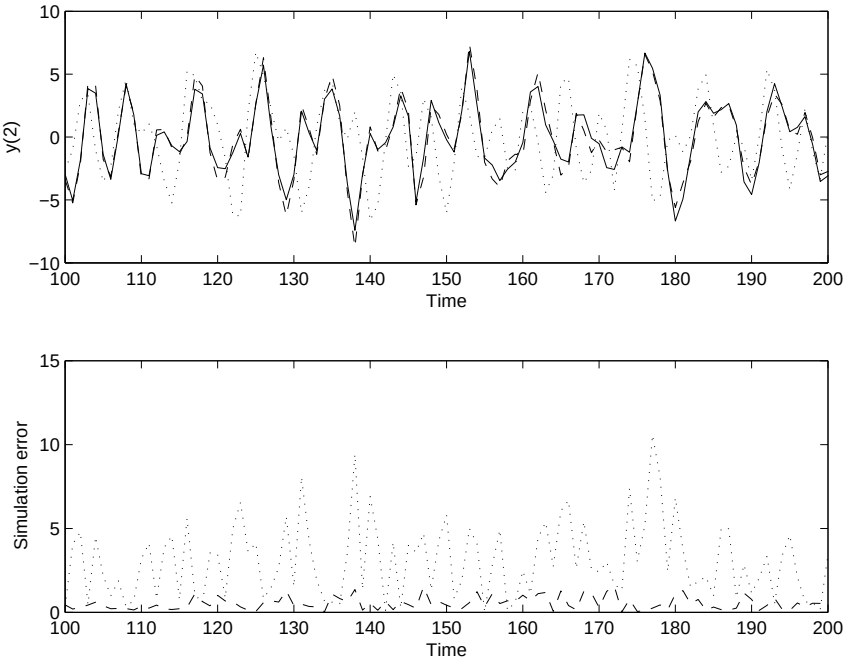


Figure 3.11: Simulation on an independent test-set of the second output of a twelfth order MIMO Hammerstein model described in Section 3.6.4 using a KCCA-based LS-SVM Hammerstein-Wiener estimator (dashed line) and a linear PO-MOESP subspace estimator (dotted line). The true output is depicted with a solid line. All simulations are initialized with $x_0 = 0$. The error between the estimated and the true output is also shown in the Figure.

3.7 Conclusions

In this chapter we mainly derived generalizations and extensions for existing methods. We presented three related methods -KPCA, KPLS and KCCA- in a unified view from the viewpoint of RKHS regression, where they deliver a suitable subspace in order to reduce the potentially high number of regression parameters in high-dimensional feature space. Starting from the least squares criterion we showed that also PLS can be motivated in a primal-dual LS-SVM optimization fashion, in addition to KPCA and KCCA [84]. We also transferred some additional variants of the PLS case to the kernel context and compared performances of the subspace regression models [83]. We finally employed successfully Kernel Canonical Correlation Analysis in the Subspace Identification of Hammerstein-Wiener systems [71], extending the classical CCA intersection algorithm to a nonlinear context.

Chapter 4

Reduced Set Methods

In this chapter we consider an extension of the least squares subspace regression in a Reproducing Kernel Hilbert Space (RKHS) that allows to perform kernel regression on very large data sets. Kernel based methods namely suffer from exceeding time and memory requirements when applied on large datasets, since the involved optimization problems typically scale polynomially in the number of data samples.

As a remedy, some kernel methods on the one hand only reduce the number of parameters (for fast training), or on the other hand only work on a reduced set (for fast evaluation). Departing from the Nyström based feature approximation, via the Fixed-Size LS-SVM model, we propose a general regression framework, based on restriction of the search space to a subspace and a particular choice of basis vectors in feature space [87].

In this general model both reduction aspects are unified and become explicit model choices. This allows to accommodate kernel Partial Least Squares and kernel Canonical Correlation analysis for regression with a sparse representation, which makes them applicable to large data sets, with little loss in accuracy [86]. The enhanced methods are illustrated and empirically compared on some benchmark data sets with excellent results [54].

4.1 Subset Based Least Squares Subspace Regression in RKHS

Kernel methods typically operate in a RKHS, of which we introduced the basics in Chapter 2. The high-dimensionality which could pose problems for proper parameter estimation is circumvented by the kernel trick, which brings the

dimensionality to the number of training instances n and at the same time allows excellent performance in classification and regression tasks. Yet, for large datasets this dimensionality in n means a serious bottleneck, since the corresponding training methods scale polynomially in n. Downsizing the system in dimensions to size $m \ll n$ is therefore needed. On the one hand, low-rank approximations (e.g. [194, 60]) offer reduction to smaller $m \times m$ matrices. On the other hand reduced set methods work with a thin tall $n \times m$ system (e.g. [109, 107, 160, 190, 143]). This work connects to the latter perspective.

We start from the Nyström approximation [195], delivering a direct approximation of features, which, applied in an ordinary least squares model, leads to kernel Principal Component Regression (KPCR) and fixed-size LS-SVMs [165]. Compared to KPCR, FS-LSSVM also only needs a small number of parameters, but additionally the regression is done in the primal space leading to a *sparse representation*, unlike estimation in the dual space as done in Gaussian Processes without having a sparse representation [194, 195]. Such a parsimonious model is highly desirable when dealing with large data sets since it implies a substantial reduction in training and evaluation time.

By formalizing the structure of the FS-LSSVM model, we then obtain a least squares regression framework in which (i) we explicitly restrict the regression coefficients to a subspace and (ii) we express the subspace in the basis formed by mapped training data points. These model constraints respectively allow to control the number of regression parameters and the number of kernels. Our general model formulation introduces kernels in a natural manner into the model, yields a complementary, unifying viewpoint on some other kernel methods and it comprises a class of models. We especially focus on the expression in a *subset* $m \ll n$ of features of the RKHS. This scheme of LS subspace regression delivers a linear system that consequently only needs polynomial training times in m.

Like KPCR was extended by FS-LSSVM with a sparse representation, we now accommodate here also kernel Partial Least Squares (KPLS) (together with some variants [83]) and kernel Canonical Correlation analysis (KCCA) with a sparse representation in a natural and efficient manner (different from [125], where sparsification is induced via a multi-step adaptation of the algorithm with extra computational burdens). Hereby we make their application possible for large scale data sets. In the subspace regression model, the extension consists of using alternative eigenspaces constructed by optimization of other (co)variance criteria. In some examples we finally show that these models perform well with little loss of accuracy and can effectively manage large data sets.

It is a difficulty that in general the elements of the feature matrix

$$\Phi = \begin{pmatrix} \varphi^T(\mathbf{x}_1) \\ \varphi^T(\mathbf{x}_2) \\ \vdots \\ \varphi^T(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} \sqrt{\lambda_1}\phi_1(\mathbf{x}_1) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_1) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_1) \\ \sqrt{\lambda_1}\phi_1(\mathbf{x}_2) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_2) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{\lambda_1}\phi_1(\mathbf{x}_n) & \sqrt{\lambda_2}\phi_2(\mathbf{x}_n) & \dots & \sqrt{\lambda_n}\phi_d(\mathbf{x}_n) \end{pmatrix} \tag{4.1}$$

are unknown because the explicit expression for φ is not available. The common approach to deal with this issue is then to make use of the so-called kernel trick [154]. The need for a direct expression is typically avoided by an ad hoc substitution such that scalar products are formed and consequently kernels come into play. Thus, the element $\varphi(\mathbf{x})$ may be not explicitly known, but the projection on any other $\varphi(\mathbf{x}')$ is simply $k(\mathbf{x}, \mathbf{x}')$, and then the unknown elements are eliminated.

Here we follow another approach in which we start from an approximate explicit expression for $\varphi(\mathbf{x})$, via the Nyström method. We will show how it leads to a least squares regression model, (Fixed-Size LS-SVM), with a downsized number of parameters and a sparse kernel expansion.

4.2 The Nyström approximation

The Nyström method dates back to the late 1920's [130]. It offers an approximate solution to integral equations [10, 139]. It became recently of interest in the kernel machine learning community via Gaussian processes [194, 195] and is recognized as implicitly present in the projection of features in the eigenspace produced by Kernel principal component analysis (KPCA), introduced in [155]. More specifically, the technique yields a simple approximation of the features for a predetermined subset of $m \ll n$ training points. These are then used on their turn as an interpolative, but quite accurate, formula for the features in the remaining training points.

4.2.1 Approximation of eigenfunctions

One can discretize the integral of (2.18) on a finite set of evaluation points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, with simple equal weighing, yielding a system of equations:

$$\frac{1}{n}\sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j)\phi_i(\mathbf{x}_j) = \lambda_i^{(n)}\phi_i^{(n)}(\mathbf{x}), \tag{4.2}$$

which can be structured as a matrix eigenvalue problem:

$$KU_n = U_n\Lambda_n, \tag{4.3}$$

where $K_{ij} = \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j)$ are the elements of the kernel Gram matrix, U_n a $n \times n$ matrix of eigenvectors of K and Λ_n a $n \times n$ diagonal matrix of nonnegative eigenvalues in non-increasing order. (We emphasize the dimensions of matrices by a subscript and for vectors it is indicated in the superscript, between brackets to avoid confusion with powers). Expression (4.2) delivers direct approximations of the eigenvalues and eigenfunctions for the $\{\mathbf{x}_j\}_{j=1}^n$ points:

$$\phi_i(\mathbf{x}_j) \approx \sqrt{n}\ \mathbf{u}_{ji}^{(n)}, \tag{4.4}$$

$$\lambda_i \approx \frac{1}{n}\lambda_i^{(n)}. \tag{4.5}$$

It was a key observation of Nyström to backsubstitute these approximations in (4.2) to obtain a good estimate of an eigenfunction evaluation in new points $\mathbf{x}'$:

$$\phi_i(\mathbf{x}') = \frac{\sqrt{n}}{\lambda_i^{(n)}} \sum_{j=1}^{n} k(\mathbf{x}', \mathbf{x}_j)\mathbf{u}_{ji}^{(n)} = \frac{\sqrt{n}}{\lambda_i^{(n)}} \left(\mathbf{k}^{(n)}(\mathbf{x}')\right)^T \mathbf{u}_i^{(n)}, \tag{4.6}$$

where $\mathbf{k}^{(n)}(\mathbf{x}) = \Phi\varphi(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})]^T$. As can be seen from (2.20),

$$\varphi(\mathbf{x}) = \left[\sqrt{\lambda_1}\ \phi_1(\mathbf{x}) \quad \sqrt{\lambda_2}\ \phi_2(\mathbf{x}) \quad \ldots \quad \sqrt{\lambda_d}\ \phi_d(\mathbf{x})\right]^T \tag{4.7}$$

the entries of features in a RKHS match the eigenfunctions, apart from a scaling factor. Thus an approximate eigenfunction yields an approximate feature.

4.2.2 Feature approximation based on the complete training data set

Maximally n approximate eigenfunctions ϕ_i can be computed, since we have only n given training data points, while the rest of the input space is unsampled. The Nyström approximation allows to obtain an explicit expression for the entries of the approximated feature vector:

$$\varphi_i(\mathbf{x}) = \sqrt{\lambda_i}\ \phi_i(\mathbf{x}) \tag{4.8}$$

$$= \left(\lambda_i^{(n)}\right)^{-1/2} \sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j)\mathbf{u}_{ji}^{(n)} \tag{4.9}$$

$$= \left(\lambda_i^{(n)}\right)^{-1/2} \left(\mathbf{k}^{(n)}(\mathbf{x})\right)^T \mathbf{u}_i^{(n)}. \tag{4.10}$$

The approximation to a feature becomes a $n \times 1$ vector:

$$\varphi(\mathbf{x}) \approx \Lambda_n^{-1/2} U_n^T \mathbf{k}^{(n)}(\mathbf{x}). \tag{4.11}$$

At the same time we obtain also a $n \times n$ matrix approximation Φ_n of the $n \times d$ *regressor matrix* (4.1) :

$$\Phi = \begin{pmatrix} \varphi^T(\mathbf{x}_1) \\ \varphi^T(\mathbf{x}_2) \\ \vdots \\ \varphi^T(\mathbf{x}_n) \end{pmatrix} \approx K U_n \Lambda_n^{-1/2} =: \Phi_n. \tag{4.12}$$

If we use this matrix in a least squares setting, we obtain a linear model in feature space. More specifically we are then performing PCR, which has the advantage of reducing the number of parameters if enough components are left out, but a sparse kernel expansion is not obtained. In next section we show that this extra requirement can be added.

4.2.3 Feature approximation based on a training data subset

In order to introduce parsimony in the number of kernels, we choose a subset of $m \ll n$ data points and then a likewise m-approximation can be made:

$$\lambda_i \approx \frac{1}{m} \lambda_i^{(m)}, \tag{4.13}$$

$$\phi_i(\mathbf{x}) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}} \sum_{j=1}^{m} k(\mathbf{x}, \mathbf{x}_j)\ u_{ji} = \frac{\sqrt{m}}{\lambda_i^{(m)}} \left(\mathbf{k}^{(m)}(\mathbf{x}) \right)^T\ \mathbf{u}_i. \tag{4.14}$$

Again one can obtain an explicit expression for the entries of the approximated feature vector:

$$\varphi_i(\mathbf{x}) = \sqrt{\lambda_i}\ \phi_i(\mathbf{x}) \tag{4.15}$$

$$\approx \left(\lambda_i^{(m)} \right)^{-1/2} \sum_{j=1}^{m} k(\mathbf{x}_j, \mathbf{x})\ u_{ji}. \tag{4.16}$$

The approximation to the feature vector now becomes a $m \times 1$ vector:

$$\varphi(\mathbf{x}) \approx \Lambda_m^{-1/2} U_m^T \mathbf{k}^{(m)}(\mathbf{x}). \tag{4.17}$$

It yields the following approximation of the feature matrix

$$\Phi_n = K_{nm} U_m \Lambda_m^{-1/2}. \tag{4.18}$$

If this approximation is directly used in a LS model we obtain

$$f(\mathbf{x}) = \left(\mathbf{w}^{(m)} \right)^T\ \varphi(\mathbf{x}) + b \tag{4.19}$$

$$= \sum_{j=1}^{m} w_j \left(\sum_{i=1}^{m} \left(\lambda_j^{(m)}\right)^{-1/2} u_{ij}^{(m)} k(\mathbf{x}_i, \mathbf{x}) \right) + b \tag{4.20}$$

$$:= \sum_{i=1}^{m} \beta_i \; k(\mathbf{x}_i, \mathbf{x}) + b \tag{4.21}$$

where $\beta_i = \sum_{j=1}^{m} w_j \left(\lambda_j^{(m)}\right)^{-1/2} u_{ij}^{(m)}$. This model, in conjunction with an entropy-based selection procedure has been named Fixed-Size LS-SVM and proposed in [165]. It delivers a *sparse kernel expansion* and at the same time the number of parameters of $\mathbf{w}$ is kept equal to m.

Eigenvector/value approximations

Next to the above two models that resulted from the Nyström approximation, yet another model can be identified. For this, consider the eigenvalues/vectors of the m-points approximation and how they relate to the n-points approximation [195]:

$$\lambda_i^{(n)} = \frac{n}{m} \lambda_i^{(m)}, \tag{4.22}$$

$$\mathbf{u}_i^{(n)} = \sqrt{\frac{n}{m}} \frac{1}{\lambda_i^{(m)}} K_{nm} \mathbf{u}_i^{(m)} \tag{4.23}$$

where $i = 1, \ldots, m$. And once again one can obtain an explicit expression for the m entries of the approximate feature vector:

$$\varphi_i(\mathbf{x}) = \sqrt{\lambda_i} \; \phi_i(\mathbf{x}) \tag{4.24}$$

$$\approx \left(\lambda_i^{(n)}\right)^{-1/2} \sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j) u_{ji}^{(n)} \tag{4.25}$$

$$= (\lambda_i^{(m)})^{-3/2} \sum_{j=1}^{n} k(\mathbf{x}, \mathbf{x}_j) \sum_{l=1}^{m} k(\mathbf{x}_j, \mathbf{x}_l) u_{li}^{(m)} \tag{4.26}$$

$$= (\lambda_i^{(m)})^{-3/2} \left(\mathbf{k}^{(n)}(\mathbf{x})\right)^T K_{nm} \mathbf{u}_i^{(m)}. \tag{4.27}$$

This approximation leads to a $m \times 1$ feature vector

$$\varphi(\mathbf{x}) \approx \Lambda_m^{-3/2} \; U_m^T \; K_{nm}^T \; \mathbf{k}^{(n)}(\mathbf{x}) \tag{4.28}$$

and feature matrix

$$\Phi = \begin{pmatrix} \varphi^T(\mathbf{x}_1) \\ \varphi^T(\mathbf{x}_2) \\ \vdots \\ \varphi^T(\mathbf{x}_n) \end{pmatrix} \approx K K_{nm} U_m \Lambda_m^{-3/2} = K \tilde{U}_{nm} \Lambda_m^{-1/2}, \tag{4.29}$$

where

$$\tilde{U}_{nm} = \begin{bmatrix} U_m \\ K_{(n-m)m} U_m \Lambda_m^{-1} \end{bmatrix} \tag{4.30}$$

is the approximation to the first m columns of the eigenspace of the kernel matrix K, but is not a true left eigensubspace, since its columns are not orthogonal to each other.

If this approximation is directly used in the least squares model we obtain the kernel expansion:

$$\begin{aligned} f(\mathbf{x}) &= \left(\mathbf{w}^{(m)}\right)^T \varphi(\mathbf{x}) + b && (4.31)\\ &= \sum_{j=1}^{m} w_j \left(\left(\lambda_j^{(m)}\right)^{-3/2} \sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i) \sum_{l=1}^{m} k(\mathbf{x}_i, \mathbf{x}_l) u_{lj}^{(m)} \right) + b && (4.32)\\ &= \sum_{j=1}^{m} w_j \left(\left(\lambda_j^{(m)}\right)^{-3/2} \sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i) K_{nm} \mathbf{u}_j^{(m)} \right) + b && (4.33)\\ &:= \sum_{i=1}^{n} \beta_i \; k(\mathbf{x}_i, \mathbf{x}) + b && (4.34) \end{aligned}$$

where $\beta_i = \sum_{j=1}^{m} w_j \left(\lambda_j^{(m)}\right)^{-1/2} \tilde{\mathbf{u}}_j^{(n)} = \sum_{j=1}^{m} w_j \left(\lambda_j^{(m)}\right)^{-3/2} K_{nm} \mathbf{u}_j^{(m)}$. We don't obtain here a sparse kernel expansion any more, but we do keep the sparseness in the number of regression parameters of $\mathbf{w}$.

With the above three models we are now in a position to generalize and identify in the following section a least squares regression model with specific restrictions. The properties of sparse kernel expansion and reduced number of parameters will become explicit model choices and these three models will be just special cases.

4.3 The restricted subspace regression model

The general goal is to obtain a parsimonious model (in both parameters and kernels) of the underlying function between a set of independent variables and an dependent variable, based on pointwise information: the data input/output instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \in \mathbb{R}^p \times \mathbb{R}$. Specifically we will firstly make explicit how to obtain reduction of the model parameters and secondly how to obtain a reduction of the kernel expansion. In this section we discuss the regression model, confine it to a subspace and restrain the choice of basis vectors.

4.3.1 Least Squares regression

Consider the standard simple linear regression model in feature space

$$\mathbf{y} = \Phi \mathbf{w}^{(d)} + \mathbf{e}, \tag{4.35}$$

where $\mathbf{y}$ represents a $n \times 1$ vector of observations of the dependent variable, Φ is a $n \times d$ matrix of regressors $\{\varphi(\mathbf{x}_i)\}_{i=1}^n$, $\mathbf{w}^{(d)}$ is the unknown $d \times 1$ vector of regression coefficients and $\mathbf{e}$ is a $n \times 1$ vector of errors with zero-mean Gaussian i.i.d. values of equal variance σ^2 (unknown). We will assume all mapped data variables have been mean-centered.

To estimate the unknown model parameters, the elements of $\mathbf{w}^{(d)}$, we choose to minimize the squared errors $\mathbf{e}$ by Ordinary Least Squares (OLS) regression

$$\min_{\mathbf{w}^{(d)} \in \mathbb{R}^d} \|\mathbf{y} - \Phi \mathbf{w}^{(d)}\|_2^2, \tag{4.36}$$

with the least squares estimate of the coefficient vector

$$\hat{\mathbf{w}}_{\mathrm{OLS}}^{(d)} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}. \tag{4.37}$$

Additionally, one can also choose to add a regularization parameter (with regularization constant γ) to the optimization problem, which is the well-known case of Ridge Regression (RR) [88]:

$$\min_{\mathbf{w}^{(d)} \in \mathbb{R}^d} \|\mathbf{y} - \Phi \mathbf{w}^{(d)}\|_2^2 + \frac{1}{\gamma} \|\mathbf{w}^{(d)}\|_2^2, \tag{4.38}$$

which introduces a bias, but may have the effect of variance reduction of the regression coefficients estimate

$$\hat{\mathbf{w}}_{\mathrm{RR}}^{(d)} = (\Phi^T \Phi + \frac{1}{\gamma} I_d)^{-1} \Phi^T \mathbf{y}. \tag{4.39}$$

To further enhance the model a bias term b can be added:

$$\min_{\mathbf{w}^{(d)} \in \mathbb{R}^d} \|\mathbf{y} - (\Phi \mathbf{w}^{(d)} + b)\|_2^2 + \frac{1}{\gamma} \|\mathbf{w}^{(d)}\|_2^2. \tag{4.40}$$

The bias term is often useful since it allows the model to account for an 'intercept' and may therefore improve the approximation. And then we obtain a system of linear equations

$$\begin{bmatrix} 1 & (\Phi \mathbf{1})^T \\ \Phi \mathbf{1} & \Phi^T \Phi + \frac{1}{\gamma} I_d \end{bmatrix} \begin{bmatrix} b \\ \mathbf{w}^{(d)} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T \mathbf{y} \\ \Phi^T \mathbf{y} \end{bmatrix}. \tag{4.41}$$

4.3.2 Restriction to a subspace

A first difficulty in this feature space setup, is that d may be very large compared to the number of data points n, even $d = +\infty$ potentially. This implies that also an infinite number of regression coefficients will be necessary. But this infinite number of degrees of freedom is not practical, nor sensible, since one can approximate any function with an infinite number of parameters.

So in order to obtain a parsimonious model in the parameters one can reduce the search for the estimate to a subspace of $\mathbb{R}^{d\times 1}$, with finite dimension $s \ll d$. We can gather linearly independent vectors $\{\mathbf{v}_i\}_{i=1}^{s}$ in the columns of a $d \times s$ transformation matrix V, such that they span the subspace. This column space, denoted by range$(V) = \{\mathbf{w}^{(d)} | \mathbf{w}^{(d)} = V\boldsymbol{\alpha}^{(s)}$ for any $\boldsymbol{\alpha}^{(s)}\}$ forms then a confined search space for the regression vector estimate and our OLS optimization is adapted as

$$\min_{\mathbf{w}^{(d)} \in \text{ range(V)}} \|\mathbf{y} - \Phi\mathbf{w}^{(d)}\|_2^2, \tag{4.42}$$

where the restricted OLS estimate is determined by s regression coefficients:

$$\hat{\mathbf{w}}_{\text{OLS}}^{(d)} = V(V^T\Phi^T\Phi V)^{-1}V^T\Phi^T\mathbf{y} := V\hat{\boldsymbol{\alpha}}_{\text{OLS}}^{(s)}. \tag{4.43}$$

Remark that this model is treated in feature space, were $\hat{\boldsymbol{\alpha}}_{\text{OLS}}^{(s)}$ is typically called a *dual* variable, which can be associated to a Lagrange multiplier. The dual formulation is here equivalent with a LS problem in transformed regressors. If we express the $\varphi(\mathbf{x}_i)$ in the new coordinates $\mathbf{z}(\mathbf{x}_i) = V^T\varphi(\mathbf{x}_i)^T$, so that in matrix notation $Z = \Phi V$, we obtain Z as a $n \times s$ matrix of transformed regressors. The (i,k)th entry of Z is the value of the projection on the kth feature vector $\mathbf{v}_k$. The resulting estimate from the (unrestricted) regression in Z would be a directly obtained solution $\hat{\mathbf{w}}_{\text{OLS}}^{(s)}$, and is considered as a *primal* variable. In this latter sense the Fixed-Size LS-SVM is originally defined as a model in primal space rather than in dual space.

We also can modify the RR model with a penalisation of the magnitude of the regression parameter vector,

$$\min_{\mathbf{w}^{(d)} \in \text{ range(V)}} \|\mathbf{y} - \Phi\mathbf{w}^{(d)}\|_2^2 + \frac{1}{\gamma}\|\mathbf{w}^{(d)}\|_2^2, \tag{4.44}$$

where the restricted RR estimate is determined by s regression coefficients

$$\hat{\mathbf{w}}_{\text{RR}}^{(d)} = V(V^T\Phi^T\Phi V + \frac{1}{\gamma}V^TV)^{-1}V^T\Phi^T\mathbf{y} := V\hat{\boldsymbol{\alpha}}_{\text{RR}}^{(s)}. \tag{4.45}$$

This RR model is a LS problem in transformed regression coefficients, and *not* in the regressors, because for that, the term $\frac{1}{\gamma}V^TV$ would be $\frac{1}{\gamma}I$, which is plain RR on some set of transformed features. The resulting estimate from (unrestricted)

RR on some set of transformed features would then be $\hat{\mathbf{w}}_{\mathrm{RR}}^{(s)}$, and be considered as a *primal* variable. If $s = d$ then a mere basis change is performed, but when $s < d$, then the solution is confined to a proper subspace. The particular choice of the basis vectors will of course affect the quality of the model.

Including also a bias term b into the model (4.44) yields the general $(s{+}1){\times}(s{+}1)$ system

$$\begin{bmatrix} 1 & (\Phi V\mathbf{1})^T \\ \Phi V\mathbf{1} & V^T\Phi^T\Phi V + \frac{1}{\gamma}V^TV \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha}^{(s)} \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T\mathbf{y} \\ V^T\Phi^T\mathbf{y} \end{bmatrix}. \tag{4.46}$$

4.3.3 Choice of basis

A second difficulty is that in general the elements of matrix Φ are unknown because the explicit expression for φ is not available. An elegant and optimal approach is to make use of the so-called kernel trick [154]. The need for a direct expression is typically avoided by formal manipulation towards scalar products so that kernels come into play. The element $\varphi(\mathbf{x})$ may be not explicitly known, but the projection on any other $\varphi(\mathbf{x}')$ is simply $k(\mathbf{x}, \mathbf{x}')$.

In order to insert kernels in our setup, we may choose for the subspace any linear combination of $\varphi(\mathbf{x}_i)$. If we select a (sub)set of $m \leq n$ features, this can be achieved by introducing a subspace decomposition $V = \Phi_m^T A$ with Φ_m a feature matrix in the m input vectors. The $m \times s$ matrix A holds in fact the projections (scalar products or loadings) of the s basis vectors $\mathbf{v}_k$ onto the m (score) vectors $\varphi(\mathbf{x}_l)$. This expression of V in the column basis of Φ_m^T allows to write $Z = \Phi V = \Phi\Phi_m^T A = K_{nm}A$ and the OLS model equation becomes

$$\mathbf{y} = Z\boldsymbol{\alpha}^{(s)} + b\mathbf{1} + \mathbf{e} = (K_{nm}A)\boldsymbol{\alpha}^{(s)} + b\mathbf{1} + \mathbf{e}. \tag{4.47}$$

Remark that the use of kernels requires in fact to represent the subspace spanning vectors in the basis of the mapped datapoints. As such, the determination of V is replaced by a determination of the $m \times s$ matrix A on the basis of the data. In next subsection we will address the subspaces and loading matrices.

Some more general remarks:

- Once computed the $s \times 1$ vector $\boldsymbol{\alpha}^{(s)}$ of regression coefficients, we can obtain an estimate of $\mathbf{y}$, and we can indeed express the linear model as an expansion of kernels:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \left(\sum_{j=1}^{s} a_{ij}\alpha_j^{(s)} \right) k(\mathbf{x}_i, \mathbf{x}) + b := \sum_{i=1}^{m} \beta_i \; k(\mathbf{x}_i, \mathbf{x}) + b. \tag{4.48}$$

- If we wish to evaluate the function after training in other (test) points, then for a given test set $\{(\mathbf{x}_j, y_j)\}_{j=n+1}^{t}$ we have a $t \times d$ feature matrix Φ_t and a corresponding $t \times n$ kernel matrix $K_t = \Phi_t \Phi_m^T$ so that the prediction in new points is expressed as

$$\hat{\mathbf{y}}_t = (K_{tm} A)\boldsymbol{\alpha}^{(s)} + b\mathbf{1} \tag{4.49}$$

- Since we assumed that the data are centered, we need to adjust the expression $\varphi(\mathbf{x})$ with $\varphi(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n}\varphi(\mathbf{x}_i)$ everywhere. For the kernel matrices this has the consequence of replacing K by [155]

$$K := (I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T)K(I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T), \tag{4.50}$$

  where I_n is the identity matrix and $\mathbf{1}_n$ a $n \times 1$ vector of ones.

4.3.4 A complementary view

In table 4.1 it is summarized how the three LS models of Section 4.2 appear as special cases of the sparse restricted least squares kernel regression framework. But also on some other common kernel methods the concept of restricted regression offers a useful complementary view. As such, we do not present with our framework one kernel version of one specific algorithm, but in fact a whole general class of kernel models. We sum up a few important example subcases that can be encountered in the literature.

method	*subspace*		*sparsity*	
	subset	loading	kernels	parameters
KPCR	Φ_n^T	$U_n\Lambda_n^{-1/2}$	no	yes
FS-LSSVM	Φ_m^T	$U_m\Lambda_m^{-1/2}$	yes	yes
low-rank	Φ_m^T	$K_{mm}^{-1}K_{nm}^T$	yes	no

Table 4.1: Interpretation of different LS models with Nyström based features, as special cases in the sparse restricted least squares kernel regression framework.

- If we set $V = \Phi^T$ in the restricted OLS model with radial basis kernels, then we relate immediately to RBF-networks [138].
- If we confine the regression coefficients to a subspace $V = \Phi^T$, then the restricted RR model yields

$$\hat{\mathbf{w}}_{\text{RR}}^{(n)} = \Phi^T \left(K + \frac{1}{\gamma}I\right)^{-1}\mathbf{y}, \tag{4.51}$$

where we made use of the fact that K is invertible and symmetric. In this estimate we may readily recognize the kernel RR model solution, originally proposed in [150], and which shows up also in Regularization Networks [68, 56] and Gaussian Processes [129] context.

- If the restricted RR model is enhanced with an additional bias term then it is related to the Least Squares Support Vector Machine (LS-SVM) [165], which also emphasizes primal-dual formulations of the problem.

- Furthermore, to induce kernel sparseness, we could choose a reduced set of basis vectors $V = \Phi_m^T$ and then we recover the solutions [160, 143]

$$\hat{\mathbf{w}}_{\mathrm{RR}}^{(m)} = \Phi_m^T \left(K_{nm}^T K_{nm} + \frac{1}{\gamma} K_{mm} \right) K_{nm}^T \mathbf{y} \tag{4.52}$$

- When we also include the bias term, one can recover the following model [22]

$$\begin{bmatrix} 1 & \mathbf{1}_n^T K_{nm} \\ K_{mn}\mathbf{1}_n & K_{mn}K_{nm} + \frac{1}{\gamma} K_{mm} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha}^{(m)} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_n^T \mathbf{y} \\ K_{mn}\mathbf{y} \end{bmatrix}. \tag{4.53}$$

- Another proposal for a subspace may turn out to be close to the Nyström approximation. Suppose we plan to restrict to a subset of m basis vectors, then it seems a reasonable idea to take those vectors in this subspace that are closest to the other $n - m$ training vectors. If we understand 'closest' -for example- in the least squares sense, this suggests that we consider the so called 'projection matrix' of the column space of Φ_m^T,

$$P_m = \Phi_m^T (\Phi_m \Phi_m^T)^{-1} \Phi_m, \tag{4.54}$$

and take as the subspace the projected data points:

$$V = P_m \Phi^T = \Phi_m^T (\Phi_m \Phi_m^T)^{-1} \Phi_m \Phi^T. \tag{4.55}$$

From the subspace regression point of view, we thus employ a loading matrix $A = K_{mm}^{-1} K_{mn}$ in the model

$$\mathbf{y} = (K_{nm} A)\mathbf{w}^{(n)} + \mathbf{e} \tag{4.56}$$

and in this case we arrive at an alternative motivation for the well-known Nyström low-rank approximation for K:

$$\tilde{K} = K_{nm} K_{mm}^{-1} K_{mn}. \tag{4.57}$$

Clearly some commonly applied kernel based (least squares) methods can be interpreted as implicitly restricting their regression coefficients to a particular subspace. This shows that the LS subspace regression view is a quite versatile

framework which links and complements smoothly to various other methods in the kernel methods field.

Specifically, we have seen in previous sections how KPCR and FS-LSSVM (with its favorable sparse kernel expansion) were related through the approximation of features. The framework of this section gives a more formal general view on those models where they appear to involve the choice of a particular subspace and basis.

4.3.5 Eigenvectors for the subspace

It is clear that a myriad of models can be devised, corresponding each time to another choice of subspace, that we can then conveniently extend with a sparse representation. In that respect we connect to the previous chapter that studied methods that yield meaningful eigenvectors based on the data, constructed unsupervised or supervised. The proposed framework developed in this chapter directly allows to accommodate these with a *sparse* kernel representation in a natural manner.

Starting for example from KPCA, we can take a subset of m training data points. We have then that the subspace $V = \Phi_m^T A$ is the $m \times s$ eigenspace of $\Phi_m^T \Phi_m$ and that the loading matrix $A = U_{ms} \Lambda_s^{-1/2}$, the $n \times s$ eigenspace of $K_{mm} = \Phi_m \Phi_m^T$, with an eigenvalue scaling (where we assume eigenvectors $\{\mathbf{a}_i\}_{i=1}^m$ ordered corresponding to nondecreasing values of the eigenvalues λ_i). Taking all components (thus $s = m$) corresponds to the Fixed Size LS-SVM model.

For KPLS (and its variants) and KCCA we just take the same approach. They deliver us meaningful $m \times m$ subspaces spanned by the column space of the loading matrix A. For an overview of the eigenvalue criterion for the methods (K)PCA, (K)CCA and some (K)PLS variants, we refer to Table 3.1 and Table 3.2. To evaluate and compare the resulting models we devised a set of experiments.

4.4 Experiments

We perform some experiments on the LS subspace models on a reduced set of size m, with each time a different eigenspace, as outlined in the previous section. We assess whether there are severe differences in performance for prediction on an independent test set. We do these tests of an artificial and two real-world data sets. These data sets are rather of moderate size, instead of large, but it allows to compare with a standard regressor solved on the full set, as an absolute reference. With a large scale data set example we demonstrate that the sparse

kernel framework indeed can manage where other methods fail.

4.4.1 Artificial data example

We applied the LS subspace model with the various subspace choices on a simple sinc function. We considered a domain dataset of 200 equally-spaced points in the interval $[-10, 10]$. The corresponding output values were centralized. We used the common Gaussian kernel (with width parameter h)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{h^2}\right). \tag{4.58}$$

In Figure 4.1 a typical picture of the first three components qualitatively show a good correlation with the target function. Non-equally-spaced sampling causes the components to be more irregular and oscillatory, while prediction will be less performant in undersampled regions and near boundaries.

In Figure 4.2 we show an approximation example on the same data, but with added Gaussian noise with standard deviation $\sigma = 0.2$. The other methods give similar component profiles and prediction results. This example shows that often with a very small subset very good approximations can be made, with performance close to the one of methods that make use of information on the full training set.

We compared for the methods for different subset sizes with each 20 trials on sinc data sets (noise added with standard deviation $\sigma = 0.2$ and subset sizes $m = 1, 2, \ldots, 50$). Parameter $h^2 \in \{e^{-4}, e^{-3.5}, \ldots, e^{10}\}$ was determined by 10-fold cross-validation (CV). From Figure 4.3 we see only one representer for all methods because they took virtually the same value of mean square error (MSE) on an independent test data set, and have comparable variance. This variance is indicated to show that for a small subset the performance is highly sensitive, while for a larger set this variance decreases. And the slight increase at the end indicates the start of the bias-variance tradeoff effects.

For reference we included the solutions of a state-of-the-art regression solver, the standard LS-SVM for regression, using the LS-SVMlab software from http://-www.esat.kuleuven.ac.be/sista/lssvmlab. The LS-SVM was trained on the full set and also on the subset only, which corresponds to best case and worst case performances respectively. From a subset size $m = 6$ on, the difference with the full set solution vanishes. Intuitively, we might say that the more redundancy in the data, the quicker a reduced set method will reach the optimal performance.

As for the KCCA parameters ν_1 and ν_2 we conclude that the regression result is fairly insensitive to ν_2, but that large values of ν_1 cause overfitting, we further just took unity values. The use of other kernels, like the polynomial or the sigmoidal kernel, did not produce such good results as the Gaussian kernel.

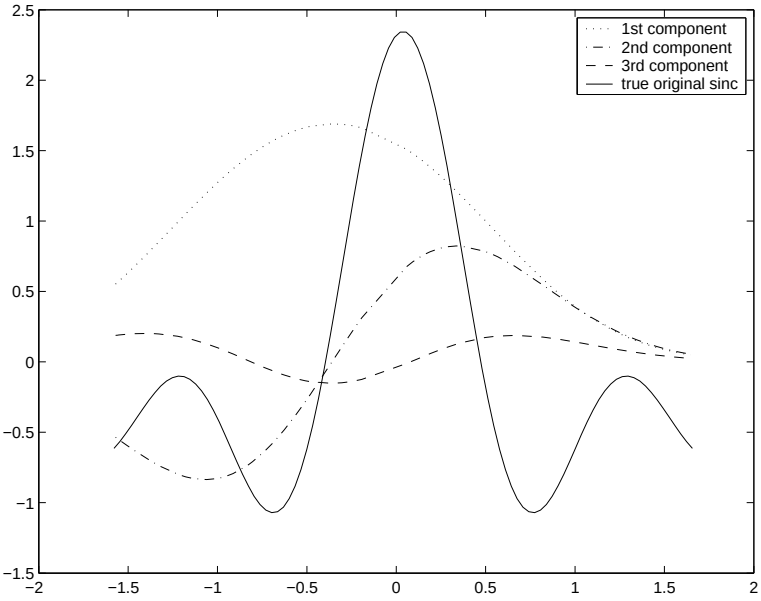


Figure 4.1: Visualisation of the features of the LS PLS-SVD subspace regression model on a sinc artificial data set sample, with added noise. The subset consists of 6 points. It are linear combinations of these clearly nonlinear features that will fit the original curve.

4.4.2 Real world data examples

The Boston Housing data set [77] consists of 506 cases having $p = 13$ input variables. The aim is to predict the housing prices. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 400$ and a test set of size $t = 106$.

We performed 10-fold CV to estimate the kernel width parameter. After CV the best model was evaluated on an independent test set. The randomization trials were repeated 20 times and the subset size over the values $(25 : 25 : 300)$ is shown in Figure 4.4. In the plot we show the MSE versus subset size. For the corresponding numeric overview of the MSE performances of the different methods, we refer to Table 4.2. The full set LS-SVM solution, achieved a best MSE= 0.1381 with regularization parameter $\gamma = 32.3517$ and kernel width $h^2 = 15.9694$, determined by CV.

On this data set the different methods yield more different results than on the sinc, and for clarity we show the worst LS subspace regression result and the best; all other methods take their values in this band. Yet, a Wilcoxon rank

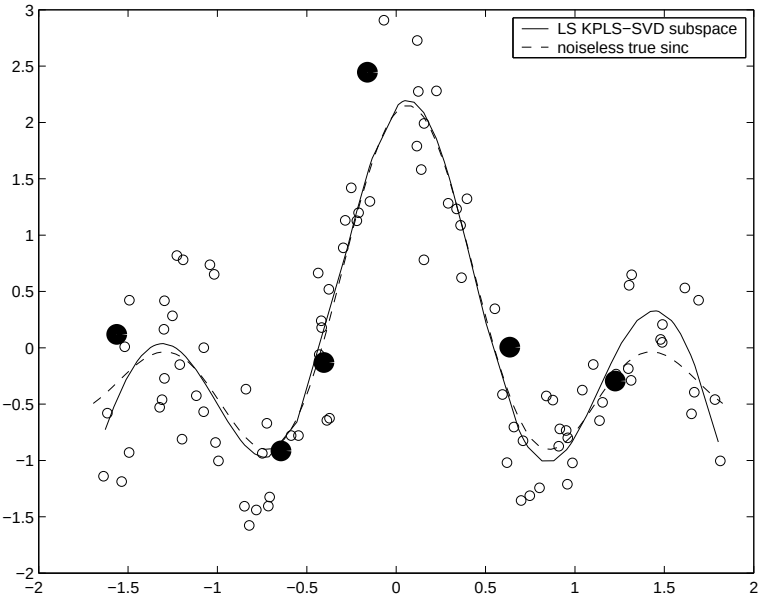


Figure 4.2: Visualization of the LS KPCA subspace regression model, leading to a sparse representation, on sinc artificial data set sample. The subset consists of 6 points, marked with a full dot on the figure.

sum test [140] at a significance level of $\alpha = 0.05$ shows no difference of the mean values between the methods relative to the standard deviations. Apart from this nondifference, we must remark that the mean of the KPCA based model is approximately equal to the mean of the KPLS1 model.

The Abalone data set is another benchmark from the same UCI repository [15], consisting of 4177 cases, having $p = 7$ input variables. The aim is to predict the age of abalone fish from physical measurements. We picked at random a training set of size $n = 3000$ and a test set of size $t = 1117$. The same tests were repeated. The full set LS-SVM solution achieved a best MSE= 0.4476 with regularization parameter $\gamma = 9.768$ and kernel width $h^2 = 12.276$, both determined by CV.

For the numeric overview of the MSE performances of the different methods, we refer to Table 4.3. As can be seen from Figure 4.5, on this data set the methods start to yield more different performance from around $m = 75$ on. Again we show a band of the worst (KPLSx), with in between the other methods (KPLS-SVD, KPLSy, KCCA, KCCA1), till the best (KPLS-U, KPLS-WA, KPLS1, KPCR). Yet, relative to the standard deviation, we may conclude that the difference is not too large in fact either. Again we note that KPLS1 coincides practically with the kernel principal component regression (KPCR) solution.

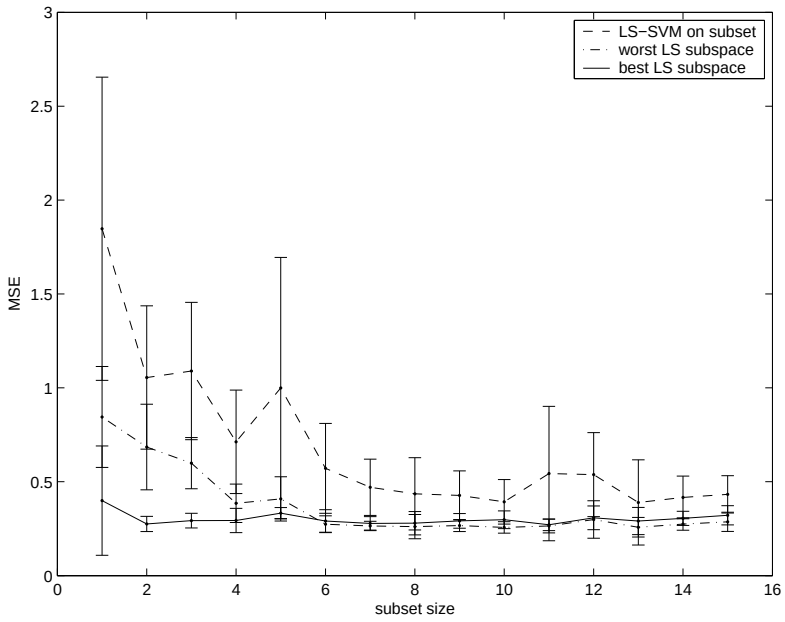


Figure 4.3: Plot of mean square error on independent test set versus subset size for the sinc artificial data set. It suffices to choose a subset of 6 data points to approximate the original function to a proper accuracy.

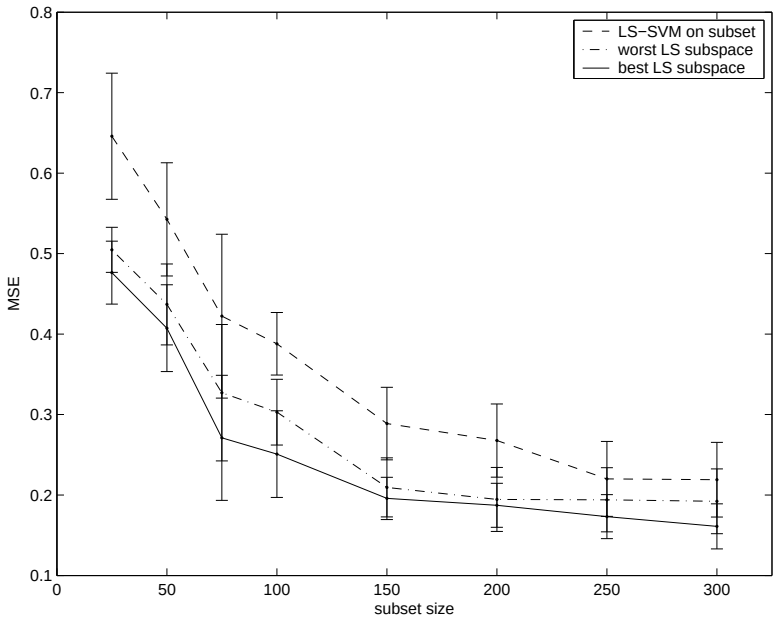


Figure 4.4: Plot of mean square error on independent test set versus subset size for the Boston housing data set. In between the best and worst bounds, the other methods take their mean performance values on the randomization sets.

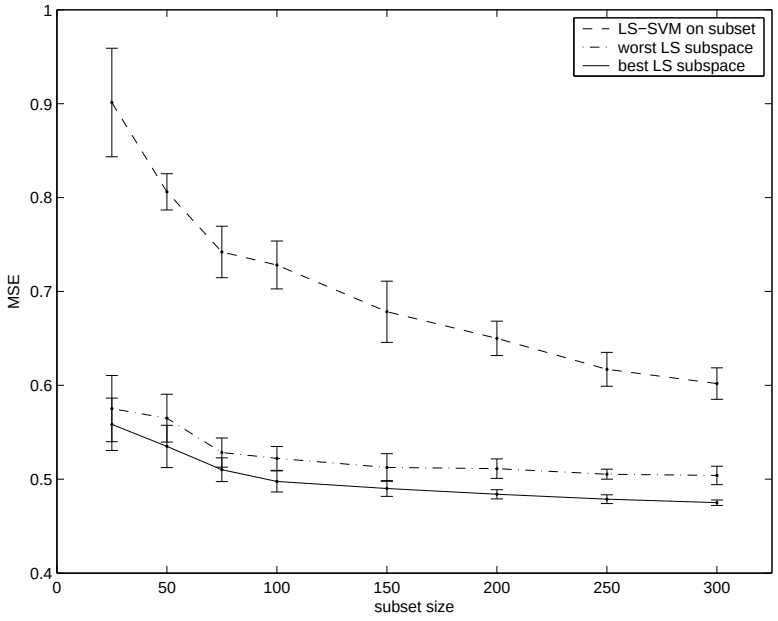


Figure 4.5: Plot of mean square error on independent test set versus subset size for the Abalone data set. In between the best and worst bounds, the other methods take their mean performance values on the randomization sets.

subset size	25	50	75	100	150	200	250	300
KPCR	0.476	0.407	0.271	0.251	0.196	0.187	0.173	0.161
	(0.039)	(0.054)	(0.078)	(0.054)	(0.026)	(0.027)	(0.027)	(0.028)
KPLS-SVD	0.505	0.437	0.327	0.303	0.209	0.195	0.194	0.192
	(0.028)	(0.050)	(0.085)	(0.041)	(0.037)	(0.040)	(0.040)	(0.040)
KPLSx	0.515	0.433	0.308	0.297	0.211	0.196	0.194	0.181
	(0.043)	(0.053)	(0.062)	(0.042)	(0.034)	(0.038)	(0.038)	(0.038)
KPLSy	0.504	0.418	0.329	0.297	0.211	0.201	0.192	0.192
	(0.045)	(0.046)	(0.094)	(0.046)	(0.034)	(0.036)	(0.038)	(0.036)
KPLS-WA	0.478	0.410	0.272	0.255	0.196	0.199	0.197	0.183
	(0.039)	(0.055)	(0.077)	(0.055)	(0.026)	(0.029)	(0.029)	(0.030)
KPLS-U	0.478	0.409	0.273	0.255	0.195	0.194	0.188	0.180
	(0.039)	(0.053)	(0.078)	(0.055)	(0.025)	(0.027)	(0.028)	(0.028)
KPLS1	0.476	0.407	0.271	0.252	0.195	0.187	0.173	0.161
	(0.039)	(0.054)	(0.078)	(0.054)	(0.026)	(0.026)	(0.028)	(0.027)
KCCA	0.503	0.420	0.316	0.291	0.203	0.198	0.194	0.192
	(0.031)	(0.054)	(0.091)	(0.043)	(0.034)	(0.035)	(0.037)	(0.035)
KCCA1	0.526	0.435	0.334	0.298	0.207	0.195	0.194	0.194
	(0.050)	(0.063)	(0.092)	(0.049)	(0.038)	(0.039)	(0.042)	(0.040)
LS-SVM	0.646	0.543	0.422	0.388	0.289	0.268	0.220	0.219
(subset)	(0.078)	(0.070)	(0.102)	(0.039)	(0.045)	(0.045)	(0.047)	(0.046)

Table 4.2: Independent test set performance in terms of mean square error (standard deviation in brackets) versus subset size for different subset based least squares subspace regression models on the Boston housing data set.

4.4.3 Large scale example

In general, from the computational side our approach achieves overall a much smaller $O(nm)$ memory cost, compared to the typical $O(n^2)$ and a computational complexity of $O(nm^3)$ compared to the typical $O(n^2)$.

The ADULT UCI data set [15] consists of 45222 cases having 14 input variables. The aim is to classify if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status, etc. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 33000$ and a test set of size $t = 12222$. We used the common Gaussian kernel with width parameter $h^2 = 29.97$ derived from preliminary experiments. We performed 10-fold cross-validation to estimate the optimal number of components. After cross-validation the best model was evaluated on the independent test set. The randomization trials were repeated 20 times, and this for subset sizes over different values in the range $[25, 1000]$. Training and evaluation time per model is typically of the order of minutes (12s to 5m) for a modest number of components of order 100 (on a Pentium 2GHz pc).

In Figure 4.6 we show the averaged misclassification rate (in percent) versus subset size m on the test set. We show the two best results, namely FS-LSSVM

subset size	25	50	75	100	150	200	250	300
KPCR	0.558	0.535	0.510	0.498	0.490	0.484	0.479	0.475
	(0.028)	(0.022)	(0.013)	(0.011)	(0.008)	(0.005)	(0.005)	(0.003)
KPLS-SVD	0.575	0.565	0.528	0.522	0.513	0.511	0.505	0.504
	(0.035)	(0.025)	(0.016)	(0.013)	(0.015)	(0.010)	(0.005)	(0.010)
KPLSx	0.582	0.565	0.528	0.522	0.517	0.506	0.503	0.502
	(0.034)	(0.028)	(0.016)	(0.012)	(0.014)	(0.013)	(0.007)	(0.007)
KPLSy	0.579	0.562	0.525	0.521	0.511	0.499	0.498	0.497
	(0.022)	(0.027)	(0.013)	(0.013)	(0.020)	(0.010)	(0.014)	(0.008)
KPLS-WA	0.570	0.544	0.515	0.504	0.494	0.490	0.483	0.481
	(0.029)	(0.026)	(0.013)	(0.014)	(0.009)	(0.005)	(0.005)	(0.005)
KPLS-U	0.577	0.545	0.522	0.508	0.496	0.490	0.484	0.479
	(0.029)	(0.023)	(0.016)	(0.017)	(0.009)	(0.008)	(0.005)	(0.007)
KPLS1	0.558	0.535	0.510	0.498	0.490	0.484	0.479	0.475
	(0.028)	(0.022)	(0.013)	(0.011)	(0.008)	(0.005)	(0.005)	(0.003)
KCCA	0.578	0.567	0.525	0.522	0.511	0.499	0.499	0.498
	(0.022)	(0.029)	(0.015)	(0.015)	(0.022)	(0.013)	(0.011)	(0.013)
KCCA1	0.578	0.567	0.526	0.524	0.510	0.498	0.498	0.498
	(0.022)	(0.022)	(0.022)	(0.014)	(0.018)	(0.011)	(0.017)	(0.019)
LS-SVM	0.901	0.806	0.742	0.728	0.678	0.650	0.617	0.602
(subset)	(0.058)	(0.019)	(0.027)	(0.025)	(0.033)	(0.018)	(0.018)	(0.017)

Table 4.3: Independent test set performance in terms of mean square error (standard deviation in brackets) versus subset size for different subset based least squares subspace regression models on the Abalone data set.

and sparse KPLS1. The LS-SVM was trained and cross-validated each time only on the subset data of size m, because it is computationally not possible to take into account the information of the entire training set. Qualitatively both the FS-LSSVM and sparse KPLS1 outperform LS-SVM (on subset) with a considerable amount of at least 4%. A Wilcoxon rank sum test [140] at a significance level of $\alpha = 0.01$ shows no difference of the mean values between the two sparse methods relative to the standard deviations. Other tests on smaller benchmark datasets all confirm that KPLS equals FS-LSSVM in performance.

It is remarkable that already with a subset size of 75 support vectors a favorable classification is obtained. The lowest rate was reached at a subset size of $m = 400$, after which we found no further improvements. Compared with results from the literature, a correct test set classification rate of $84.7(\pm 0.3)\%$ ranks among the best of results delivered by other current state-of-the-art classifiers (like instance based learners, decision trees, standard SVM with different kernels, linear/quadratic discriminant analysis, standard naive Bayes classifier) [165]. The use of other kernels, like the polynomial or the sigmoidal kernel, did not produce such good results as the Gaussian kernel.

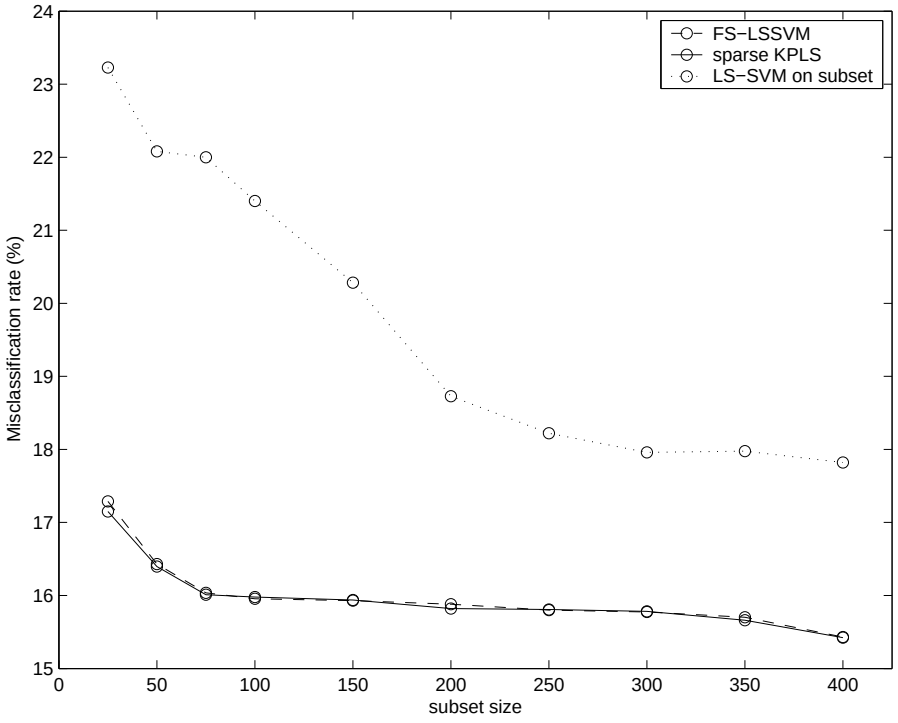


Figure 4.6: Averaged misclassification rate (in %) on the independent test set versus subset size m for FS-LSSVM, sparse KPLS and LS-SVM on the subset, on the Adult large scale data set of 45222 instances. Note that few basis vectors are necessary to reach favorable performance at a low computational cost.

4.5 A comparative benchmark study

Considered in the background of primal-dual optimization (Chapter 3), the most important benefit of the approximation of the feature vector is that instead of resorting to the dual problem, one can now remain in *primal space* and solve the task in the standard fashion whilst taking advantage of the nonlinearity embedded in the features. Next to this, there is the advantage of sparsity control in the number of model parameters and kernel terms. Such parsimonious models are a necessity when dealing with large data sets.

In this section, the emphasis is on this *primal space* based application of the proposed model in this chapter. The regression techniques (Ordinary Least Squares (OLS) and Ridge Regression (RR)) combined with subset based Kernel Principal Component Analysis (KPCA) and Kernel Partial Least Squares (PLS) are compared together with the *dual* version of LS-SVM on the 'Silver Box' dataset [158]. With our class of sparse kernel machines, we are effectively able to handle a large data set, with excellent accuracy [54].

4.5.1 The 'Silver Box' Identification Problem

The different variants of LS-SVM to be applied in this case, as described in previous sections, are:

- LS-SVM in dual space (LS-SVM). For this method, a subsample of size 1000 is used for training, as using the full training sample is prohibitive.
- Fixed Size LS-SVM in primal space with OLS (FS-OLS), RR (FS-RR) and PLS (FS-PLS). For these methods, different numbers of support vectors are selected to build the nonlinear mapping approximation. All subsamples are selected by maximization of the quadratic entropy criterion (for the motivation of this selection criterium we refer to Section 5.6.2).

The general *model structure* is a Nonlinear AutoRegression with eXtra input (NARX) specification of the form

$$y_t = f(y_{t-1}, \ldots, y_{t-p}, u_{t-1}, \ldots, u_{t-p}) + e_t, \tag{4.59}$$

where f is a nonlinear function of its arguments, e_i is the error, y_i are instances of the output and u_i are instances of the input. Exploratory analysis for estimating the order p is based on the training-validation framework.

Data description. According to the definition of the associated data generating formula, the nonlinear identification problem, known as the 'Silver Box' [158], is an example of a nonlinear dynamic system with a dominant linear behavior. The data contains samples for input u_i and output y_i, with $i = 1, \ldots, n$, with

$n = 131,072$ datapoints. An initial plot of the output (the "arrow") is given in Figure 4.7.

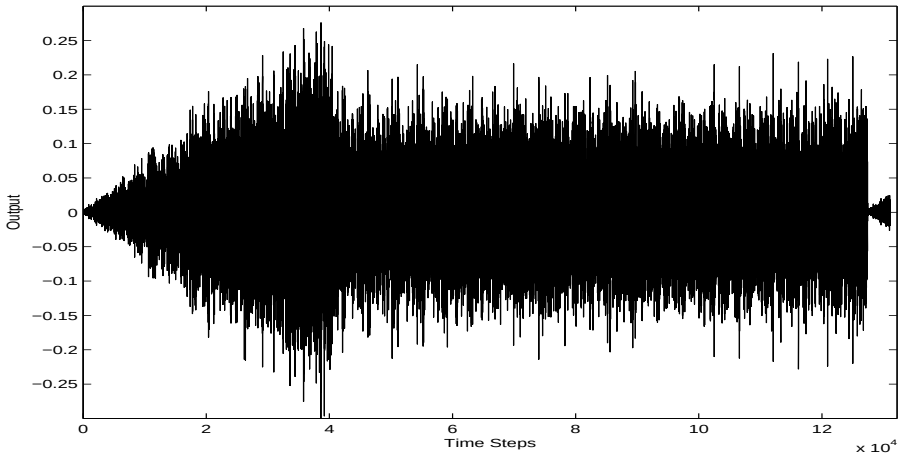


Figure 4.7: Available output data series for the Silver Box identification problem.

Training Procedure. The working strategy for using the data in terms of training, validation and testing goes as follows:

- Training Sample: The first half of the "body of the arrow", namely datapoints 40,001 to 85,000. Models are estimated using this part of the data. The mean squared error (MSE) for a one-step-ahead prediction can be computed directly using this training sample.
- Validation Sample: The second half of the "body of the arrow", datapoints 85,001 to the end. Having estimated the model parameters using the training sample, now the model is validated using new datapoints. A MSE on the validation set is computed on a one-step-ahead basis. Model selection is based on the validation MSE.
- Test Sample: The "head of the arrow", datapoints 1 to 40,000. After the model selection (using the validation MSE), the prediction for the test set is done. In this case, an iterative prediction is computed for the entire test set (each time using past predictions as inputs, using the estimated model in simulation mode). The MSE on the test set is computed.

Hyperparameter Definitions. The LS-SVM formulation requires to use a kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. In our experiments, as the nonlinear system is known

to have a dominant linear behavior, we implemented not only the RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||_2^2/\sigma^2)$, but also the polynomial kernel $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d$. Parameters σ, d, c and the regularization parameter γ are also tuned based on the training-validation scheme. From the machine-learning point of view, this procedure is not optimal as we are using only one training-validation setup; normally multiple crossvalidation is performed to achieve optimal results. In this sense, the results of this experiment can even be further improved.

4.5.2 Experimental Results

In this section we show the main results for the iterative prediction obtained with each method, and also some intermediate results related to specific definitions of the model [54].

Estimation and Model Selection

Using the definition of training and validation data described above, we start checking different lag orders and general parameters. Each time the model is estimated using the training set and then evaluated in the validation set, always on a one-step-ahead basis. We select the best model based on the lowest MSE on the validation set (MSE_{val}).

An initial analysis using a linear ARX model with increasing lags of inputs and outputs, using the same training/validation scheme, shows that the MSE for the validation set can easily reach levels of 1×10^{-7}. Figure 4.8 shows the MSE_{val} obtained when the number of lags varies from 5 to 40. Since a larger number of corresponds to more degrees of freedom, one must be cautious in selecting this number. A visual inspection of the figure shows that an order of 10 is a maximum, and that any further introduction of lags will not render better models.

For the NARX models, Table 4.4 shows the best results achieved for each one of the different techniques. It is important to remember that all the techniques based on the fixed-size primal space version make use of all the training/validation set; whereas the LS-SVM in dual space is limited to a subsample of 1000 points for training and validation. All MSE figures are expressed in the original units of the data.

It is clear that for all cases the polynomial kernel outperforms the RBF one, up to 2 orders of magnitude. Although the RBF kernel is widely used, the dominant linear behavior of the data is better captured here by the polynomial specification. Additionally, the performance of the FS-RR and FS-PLS models with polynomial kernel is much better than the obtained with the FS-OLS in the training/validation scheme. This means that there exists helpful information in

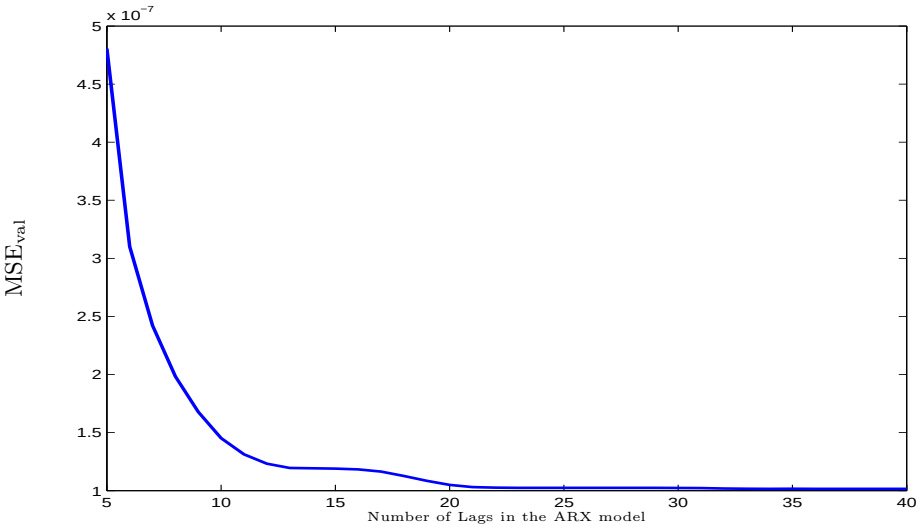


Figure 4.8: The error in the validation set using a linear ARX model with increasing number of lags.

Method	γ	Kernel	p	$\text{MSE}_{\text{train}}$	MSE_{val}
LS-SVM	10	Poly	5	2.63×10^{-9}	4.55×10^{-9}
	10	RBF	5	5.78×10^{-8}	7.66×10^{-8}
FS-OLS	-	Poly	7	1.33×10^{-7}	1.23×10^{-7}
	-	RBF	7	3.57×10^{-7}	1.72×10^{-6}
FS-RR	1000	Poly	10	5.28×10^{-8}	5.05×10^{-8}
	1000	RBF	10	3.61×10^{-7}	2.93×10^{-7}
FS-PLS	1000	Poly	10	5.34×10^{-8}	5.09×10^{-8}
	1000	RBF	10	1.16×10^{-6}	1.00×10^{-6}

Table 4.4: Best models, based on the MSE_{val}. For all cases shown, $\sigma = 5.19p^{-1/2}$ with p=number of lags (for RBF kernel); $d = 3, c = 11$ (for Polynomial kernel).

the last components of the eigendecomposition of the $m-$sized kernel matrix K, which are just dropped out in the case of FS-OLS but not in FS-RR or FS-PLS.

The effect of selecting different numbers m of initial support vectors on the validation performance is reported in Table 4.5, for the FS-OLS version with polynomial kernel, where it is clear that the performance is improving marginally

for $m > 500$. Therefore, taking into account practical considerations, we chose to keep $m = 500$ for the whole modeling exercise. The position of the selected 500 support vectors can be visualized in terms of the corresponding position of the output data $\mathbf{y}$. Figure 4.9 shows the dependent variable $\mathbf{y}$ in the training set, and the dark bars at the bottom represent the position of the selected support vectors. The quite uniform distribution shows that $\mathbf{y}$ does not have critical transition regions or critical zones. Finally, the effect of the inclusion of different lags was tested for the NARX models, using lags from 2 to 10. Figure 4.10 shows the evolution of the MSE in the validation set for FS-RR (full line), FS-OLS (dash-dot) and FS-PLS (dashed).

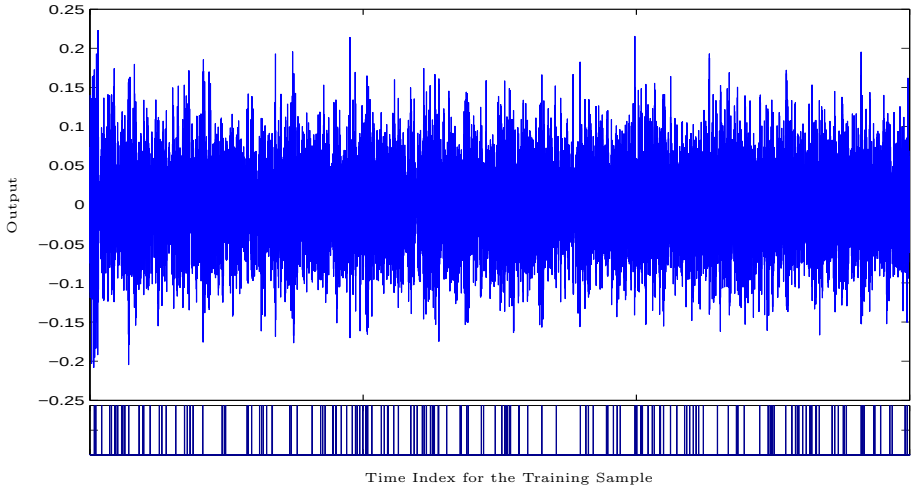


Figure 4.9: (Top) The output training sample; (Bottom) The position, as time index, of the 500 selected support vectors is represented by dark bars.

Final Results on Test Data Set

After selecting the order of the models and the parameters involved, each one of the estimated models is used to build an iterative prediction (simulation mode, using only past predictions and input information) for the first 40,000 datapoints (the "head of the arrow"). As this is a completely unseen dataset, from the point of view of the modeling strategy, we can expect two types of error sources: the first one, due to the iterative nature of the simulations, so past errors can propagate to the next predictions; and the second one, due to the fact that there are datapoints that are located beyond the range on which the models were trained, namely the wider zone of the "head of the arrow".

Number of Support Vectors m	$\text{MSE}_{\text{train}}$	MSE_{val}
100	7.91×10^{-6}	6.12×10^{-6}
200	2.11×10^{-7}	1.94×10^{-7}
300	1.57×10^{-7}	1.46×10^{-7}
400	1.42×10^{-7}	1.34×10^{-7}
500	1.33×10^{-7}	1.23×10^{-7}
1000	1.27×10^{-7}	1.19×10^{-7}
1500	1.23×10^{-7}	1.15×10^{-7}

Table 4.5: Effect of m on the performance of the FS-OLS estimator.

The iterated prediction series is compared with the true values, and then we compute this MSE on the test set (MSE_{test}).

Table 4.6 shows the results obtained with the iterative prediction, for all models, including the linear ARX model for comparison. The result for the linear model shows its lack of generalization ability for this example. The NARX models show a satisfactory performance, where FS-PLS and FS-RR obtain quite the same level of performance. As these two techniques do not drop off a priori any component of the $\boldsymbol{\varphi}$ nonlinear mapping, it can be expected to find that the performance of LS-OLS is lower than the other 2 techniques. Finally, LS-SVM with a direct subsampling for the computation of the model in dual space, obtains a MSE level in the test set within the same order of magnitude (10^{-7})), but almost 4 times the one obtained by FS-PLS or FS-RR.

Figure 4.11 shows the residuals of the iterative prediction (simulation mode), where it can be seen that the error remains within a stable zone, with the exception of very few peaks close to the wider zone of the 'head of the arrow'. In any case, the larger peak represents a 5% absolute error with respect to the level of the output series in that point.

Technique	Lags	MSE_{test}
Linear	30	0.0718
LS-SVM	5	3.90×10^{-7}
FS-OLS	7	3.67×10^{-7}
FS-RR	10	1.08×10^{-7}
FS-PLS	10	1.01×10^{-7}

Table 4.6: MSE with the final iterative prediction (simulation mode) on the test data.

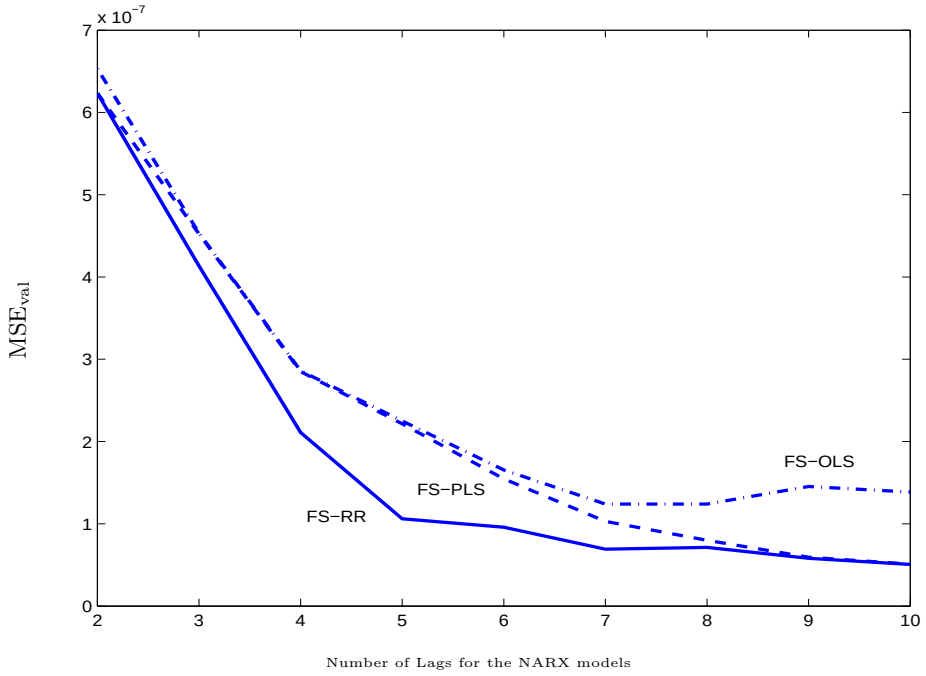


Figure 4.10: MSE on the validation set obtained for FS-RR (full line), FS-PLS (dashed) and FS-OLS (dash-dot) using different number of lags.

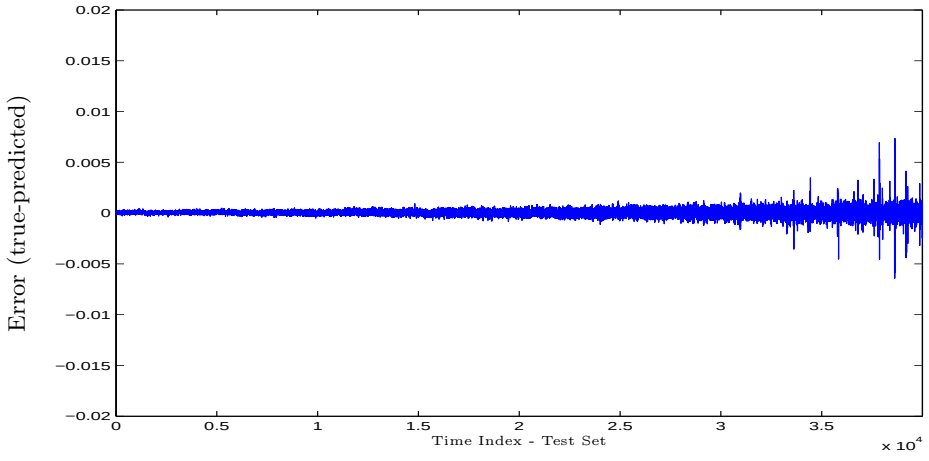


Figure 4.11: The errors of the iterative prediction (simulation mode) in the test set. Only few peaks with larger errors are visible.

4.6 Conclusions

In large scale applications a model must be able to process a large number of datapoints. Most kernel methods are simply not adequate for this task. In this chapter we started from the Nyström feature approximation which bears in a least squares regression context the KPCR and the Fixed-Size LS-SVM model. The latter operates in primal space and has two important advantages: a small number of regression coefficients (which allows a fast training) and a sparse kernel expansion (which allows fast evaluation).

Extending FS-LSSVM with supervised (i.e. including information of the dependent variable) counterparts, we presented a regression framework in which these two advantages are distinct model choices, through (i) a restriction of the search space to a subspace and (ii) a particular choice of basis vectors in feature space. We showed that our model is not just a kernelization of a single model, but comprises a whole general class of least squares kernel models at once.

We focused then on restricted models using only a small subset of basis vectors and chose eigenspaces, derived from covariance optimization criteria, that have yielded feasible models in a non-reduced setting before, like KPLS and KCCA, plus some common variants. As such, we obtain sparse KPLS and sparse KCCA models, enhanced to deal with large data sets.

From experiments we may conclude that the feasibility carries over well to the reduced setting, with a limited loss of accuracy. Taking the Nyström or KPCA based subset model as reference, some methods, like KPLS1, KPLS-U, KPLS-WA show on some test cases to perform equally well, while others perform close to it. On a large scale example we show that a sparse restricted regression model is effectively capable of dealing with a large dataset.

The application of the LS-SVM methodology to a large scale nonlinear identification problem implies the challenge of working with a large number of datapoints. In this case, the application of the LS-SVM methodology directly requires a subsampling of the data and it is not possible to take into account the information contained in the entire training set. On the contrary, fixed size variants of the LS-SVM developed to work in primal space, do rely on approximations of the nonlinear mapping φ. But these techniques have the advantage that traditional tools available for regression can be applied successfully. In this chapter, we have applied LS-SVM in dual space and 3 variants in primal space (fixed size - ordinary least squares, FS-OLS; fized size ridge regression, FS-RR; and fized size partial least squares, FS-PLS) to the identification of a nonlinear dynamical system using the *Silver Box* data.

The results show that the methods that rely on regularization and active construction of a regression in primal space (FS-RR and FS-PLS) obtain the best performance in the iterative prediction exercise. FS-OLS obtains a lower

performance, but still better than the one obtained by the LS-SVM method with a direct sampling. The best performance yields an MSE on the test set of 1.0×10^{-7}, which is better than other methods [158].

The above results were obtained with models under a suboptimal strategy for model validation and selection. Usually crossvalidation in multiple training/validation sets are done, and in this case we only used one dataset for training and one for validation. Although this is a practical decision, mainly related to the number of datapoints available, results could even be improved with a more optimal strategy.

Chapter 5

Introducing Sparsity

In this chapter we deal with the question how to choose the support vectors from the training set data. In comparison to the standard Support Vector Machines (SVM), LS-SVM only requires solving a linear system, but it lacks sparseness in the number of solution terms because it has not a built-in simultaneous resolution for this problem. Therefore LS-SVM has to be combined with an apart criterion that introduces sparsity. We consider here the pruning approach.

Standard ways of pruning the LS-SVM consist of recursively solving the approximation problem and subsequently omitting data that have a small error in the previous pass and are based on support values. We suggest a slightly adapted variant that improves the performance significantly. We assess the relative regression performance of these pruning schemes in a comparison with two (for pruning adapted) subset selection schemes, -one based on the QR decomposition (supervised), one that searches the most representative feature vector span (unsupervised)-, random omission and backward selection on independent test sets in some benchmark experiments. We made an empirical contribution by comparing some existing techniques [85].

As an extension of pruning we further consider the LS-SVM in an *online setting*, where now also the 'growing' is included. We create a framework by bringing together several components of up/downdating and show how these induce different selection criteria from the related error measures in both data and feature space. By combining the components and a selection mechanism one can construct iterative, online LS-SVM algorithms that can achieve an efficient representation based on a selected fraction of support vectors. Different strategies are compared on some benchmark datasets.

5.1 Pruning

In kernel based classification and function approximation the sparseness (i.e. limited number of kernel terms) of the approximator is an important issue, since it allows faster evaluation of new data points. The remaining points are often called support vectors. In Vapnik's SVM [184] the sparseness is built-in due to the ϵ-insensitive loss function that outrules errors of points inside a 'tube' around the approximated function. This results in a quadratic programming problem. In LS-SVM [167] a quadratic loss function is used instead, and the optimization problem reduces to solving a linear set of equations. But at the same time the sparseness is lost and must be imposed.

A simple approach to introduce sparseness is based on the sorted support value spectrum (the solution of the set of equations) [163]. From the LS-SVM solution equations follows a reasonable choice for pruning away points with a low error contribution in the dual optimization objective. Another recent publication [39] sophisticates the pruning mechanism by weighting the support values. The datapoint with the smallest error introduced after its omission is selected then. This pruning method is claimed to outperform the standard scheme of [163], but the extent of the comparison was limited to one example where noise was filtered out. We additionally suggest here a different *improved* selection of the pruning point based on their derived criterion. The previous chapter described alternative methods for achieving sparse LS-SVMs via Fixed-Size approaches [165]. This has been successfully applied to a wide class of problems for subspace regression in feature space [87]. For a general overview of pruning we refer to [142].

Pruning is closely related to subset selection, choosing relevant datapoints or variables in order to build a sparse model. Pruning assumes that the model on the full set is iteratively downsized, in a backward manner, while subset selection usually proceeds in a forward manner. Many subset selection schemes can be distinguished that organise their search in a greedy fashion [160]. In particular we focus on two such schemes. A supervised method is based on the QR decomposition [121, 27], that we can employ as omitting points of which the orthogonalized components have least correlation with the output. Furthermore, an unsupervised approach is based on a best fitting span [13], that we can employ as omitting points that have least similarity to that span.

In the experiments on pruning we aim at a comparison of regression performance between the two pruning procedures and the two subset selection procedures by performing a set of experiments with (i) evaluation on the independent test set (as the only true comparison basis) and (ii) including random and backward pruning to have an objective measure (serving as a lower bound for performance).

5.2 Pruning methods

5.2.1 Minimal coefficients

A simple way of imposing sparseness on the LS-SVM is by pruning those terms of the kernel expansion that have the smallest absolute value [163]. The motivation comes from the fact that the LS-SVM support values are proportional to the errors at the datapoints, namely $\alpha_i = \gamma e_i$. To omit the points that contribute least to the training error is a direct and cheap way to impose sparseness (denoted with 'sv' in our experiments).

5.2.2 Weighed support values

Recently a sophistication of the above pruning scheme (based on minimal coefficients) has been reported [39]. De Kruif *et al.* propose to omit the sample that introduces the smallest additional error *after* it is omitted from the data set for the next iteration.

Their motivation goes as follows. The estimation of the output at iteration m of training sample j can be calculated using just the expression of the LS-SVM kernel expansion:

$$\hat{y}_m(\mathbf{x}_j) = \sum_{i=1,k\neq j}^{n} \alpha_i^m k(\mathbf{x}_i, \mathbf{x}_j) + \alpha_j^m k(\mathbf{x}_j, \mathbf{x}_j) + b^m, \tag{5.1}$$

where the superscript on the coefficients denotes the iteration step m. When sample j is removed from the training set, the output of its input $\mathbf{x}_j$ in iteration $m+1$ is given as

$$\hat{y}_{m+1}(\mathbf{x}_j) = \sum_{i=1,k\neq j}^{n} \alpha_i^{m+1} k(\mathbf{x}_i, \mathbf{x}_j) + b^{m+1}, \tag{5.2}$$

Subtracting these outputs gives then the introduced error at sample j when sample j is omitted, as follows:

$$d(\mathbf{x}_j) = \sum_{i=1,k\neq j}^{n} \big((\alpha_i^m - \alpha_i^{m+1})k(\mathbf{x}_i, \mathbf{x}_j)\big) + \alpha_j^m k(\mathbf{x}_j, \mathbf{x}_j) + (b^m - b^{m+1}). \tag{5.3}$$

This equation shows that the introduced error does not just depend on the coefficient α_j^m, but more factors determine the introduced error. Hence, here is a valid argument for improving on the pruning criterion of [163].

De Kruif *et al.* continue then with the reasoning that the omission of a datapoint is equal to setting its Lagrangian multiplier to zero. And, on its turn, the

multipliers depend on the value of the regularization parameter γ, because $\gamma = 0$ minimizes the coefficients to zero. This leads to the key conclusion that the coefficient of a selective training point can be made zero by applying regularization with $\gamma = 0$ *only on that point.*

Case 1. So, assuming that $\gamma = \infty$ (thereby excluding the global regularization), applying only regularization for element j with an additional parameter λ gives the following set of linear equations:

$$\left[\begin{array}{c|ccc} 0 & \mathbf{1}^T & 1 & \mathbf{1}^T \\ \hline \mathbf{1} & K_{1:j-1,1:j-1} & K_{1:j-1,j} & K_{1:j-1,j+1:n} \\ 1 & K_{j,1:j-1} & K_{j,j}+\lambda & K_{j,j+1:n} \\ \mathbf{1} & K_{j+1:n,1:j-1} & K_{j+1:n,j} & K_{j+1:n,j+1:n} \end{array}\right] \left[\begin{array}{c} \mathrm{b} \\ \hline \boldsymbol{\alpha} \\ \alpha_j \\ \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{y}_{1:j-1} \\ y_j \\ \mathbf{y}_{j+1:n} \end{array}\right]. \quad (5.4)$$

When $\lambda \leftarrow 0$ then the equation becomes the known system (3.9). When $\lambda \rightarrow \infty$, the value of α_j is forced to become zero. The corresponding systems $A\mathbf{x}_1 = \mathbf{c}$ and $A_2\mathbf{x}_2 = \mathbf{c}$ are for the intermediate values of λ related to each other by a rank one difference matrix uu^T, such that [73]

$$A_2 = A + uu^T \quad (5.5)$$

$$A_2^{-1} = A^{-1} - \frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u} \quad (5.6)$$

where the column vector $u = \left[0\ \mathbf{0}\ \sqrt{\lambda}\ \mathbf{0}\right]^T$. This yields for the difference in coefficients

$$\Delta\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2 \quad (5.7)$$

$$= \left(\frac{A^{-1}uu^TA^{-1}}{1 + u^TA^{-1}u}\right)\mathbf{c} \quad (5.8)$$

$$= \left(\frac{\lambda A^{-1}e_je_j^TA^{-1}}{1 + \lambda e_j^TA^{-1}e_j}\right)\mathbf{c} \quad (5.9)$$

where e_j is a column vector of size $n+1$ filled with zeros except element $j+1$ which is equal to one. By taking the limit of $\Delta\mathbf{x}$ for with $\lambda \rightarrow \infty$ we get

$$\lim_{\lambda\rightarrow\infty} \Delta\mathbf{x} = \frac{A^{-1}e_je_j^TA^{-1}}{e_j^TA^{-1}e_j}\mathbf{c}, \quad (5.10)$$

which is used to express the error in sample j when sample j is omitted:

$$d(x_j) = [A\Delta\mathbf{x}]_j \quad (5.11)$$

$$= \left[\frac{AA^{-1}e_je_j^TA^{-1}}{e_j^TA^{-1}e_j}\mathbf{c}\right]_j \quad (5.12)$$

$$= \left[\frac{e_je_j^T\boldsymbol{\alpha}}{e_j^TA^{-1}e_j}\right]_j \quad (5.13)$$

$$= \frac{\alpha_j}{A^{-1}_{jj}}. \tag{5.14}$$

Instead of omitting the sample with the smallest absolute value of α_j, the sample with the smallest absolute value of α_j divided by the jth diagonal element of the inverse of A should be thrown out to obtain the smallest introduced error. Hence the term weighed support values. (In our experiments we denoted this model with 'weighed sv $\gamma = \infty$'.)

Case 2. In the case of $\gamma \neq \infty$ (including the global regularization with a smoothing effect), a similar reasoning can be performed. Instead of the unregularized matrix A, the regularized matrix

$$A_\gamma = A + \gamma^{-1} \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & I_n \end{bmatrix} \tag{5.15}$$

is used, and they obtain the introduced error

$$d(x_j) = \left[\frac{A A_\gamma^{-1} \mathbf{e}_i \mathbf{e}_i^T A_\gamma^{-1}}{\mathbf{e}_i^T A_\gamma^{-1} \mathbf{e}_i} c \right]_j, \tag{5.16}$$

were $A = [0, \mathbf{1}^T; \mathbf{1}, K]$, $c = [0; \mathbf{y}]$, $A_\gamma = A + \gamma^{-1} I_{(n+1)}$, and $\mathbf{e}_i$ a column vector with value 1 on element $i+1$. So the alpha's need to be weighed, which resembles the formula of optimal brain surgeon in which the inverse of the Hessian of the error surface of the model parameters appears in the denominator [78]. Because the regularization parameter γ is not infinite anymore, the omission of a sample also introduces an error at other samples. (In our experiments we denoted this model with 'weighed sv $\gamma \neq \infty$'.)

We make two remarks on this approach:

- They claim to outperform the standard method [163] and an example is given where the *training error* is indeed systematically lower. First of all, it is of utmost importance to assess any algorithm on an independent test set. The better performance could for instance be a result of overfitting. Second, only an example in which no regularization was necessary, with formula (5.15), was given. Formula (5.16) was not tested on any example at all. Strong conclusions of superiority were drawn from one special example. And third, when the sample introduces also error at other samples, it may not be most optimal to only consider the error at the point itself for selection.

  On these points we have complemented their results by performing thorough experiments (section 5.3). And as an extension to their work we propose in the case $\gamma \neq \infty$ to omit the sample such that the *sum* of all errors, introduced in *every* point is smallest. A measure like the squared norm would be more appropiate:

$$\min_i \| \frac{A A_\gamma^{-1} \mathbf{e}_i \mathbf{e}_i^T A_\gamma^{-1}}{\mathbf{e}_i^T A_\gamma^{-1} \mathbf{e}_i} c \|_2^2. \tag{5.17}$$

Because omitting a point introduces error at all points, it makes sense to look at the global increase in error over all points and exclude the point that minimizes this measure, at no extra cost [85]. In the experiments in the next section we will find empirical evidence that indeed a better selection is obtained in this manner (indicated by 'weighed sv sum $\gamma \neq \infty$').

- Compared to [163], the extension of de Kruif *et al.* comes with a more expensive computation since (5.16) requires an extra $O(4n^2)$ operations. For (5.15) they proposed to use the kernel matrix K instead of A^{-1}, to employ a downdate scheme, but then additionally K needs to be inverted at every iteration. We will show later in section 5.6 that this can be avoided and downdating is possible at virtually no cost at all (without making use of approximations), which fits in a framework of downdating criteria (see Subsection 5.6.4).

5.2.3 Orthogonal error minimizing regressors

We show in this subsection that with a simplified linear model one can motivate a support vector selection that introduces again least error. Consider the basic primal equation of LS-SVM (in matrix form), without bias term:

$$\mathbf{y} = \Phi\mathbf{w} + \mathbf{e}. \tag{5.18}$$

The primal vector $\mathbf{w}$ still remains a linear combination of the dual vector $\boldsymbol{\alpha}$ with the kernel evaluations as coefficeints (which can be derived by principled primal-dual formulations, analogously to the solution condition (3.5), or motivated from the representer theorem [151], namely $\mathbf{w} = \Phi^T\boldsymbol{\alpha}$, such that in dual space one needs to solve

$$\mathbf{y} = K\boldsymbol{\alpha} + \mathbf{e}. \tag{5.19}$$

Substitution of the ridge regression solution $K^T\mathbf{y} = (K + \gamma^{-1}I_n)\hat{\boldsymbol{\alpha}}$ in the expression of the sum of squares errors on the training data

$$\mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - \hat{\boldsymbol{\alpha}}^T K^T K\hat{\boldsymbol{\alpha}} - \frac{2}{\gamma}\hat{\boldsymbol{\alpha}}^T\hat{\boldsymbol{\alpha}}, \tag{5.20}$$

reveals dependency on the choice of variables. The second term contains negative contributions to the total error that origins from the regressors.

But, because of interdependence it is not clear to attribute more or less influence to the particular regressors. To overcome this, one can decompose $K = SR$ such that an *orthogonal* (not orthonormal) matrix S spans the same column space as K and T is an upperdiagonal matrix. This can be readily done for instance by means of the QR-factorization, such that we obtain an equivalent model

$$\mathbf{y} = S\boldsymbol{\beta} + \mathbf{e} \tag{5.21}$$

with the transformed regressors $\boldsymbol{\beta} = T\boldsymbol{\alpha}$. This is called orthogonal least squares [27]. The parameter vector obtained by least squares minimization $\hat{\boldsymbol{\beta}} = (S^T S)^{-1} S^T \mathbf{y}$ yields now more transparant terms in the error

$$\mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - \sum_{i=1}^{n} \hat{\boldsymbol{\beta}}^2 \mathbf{s}_i^T\mathbf{s}_i - \frac{2}{\gamma}\hat{\boldsymbol{\beta}}^T\hat{\boldsymbol{\beta}} \tag{5.22}$$

$$= \mathbf{y}^T\mathbf{y} - \sum_{i=1}^{n} \hat{\boldsymbol{\beta}}^2 \frac{s_i^T\mathbf{y}}{(\mathbf{s}_i^T\mathbf{s}_i)(\mathbf{y}^T\mathbf{y})} - \frac{2}{\gamma}\hat{\boldsymbol{\beta}}^T\hat{\boldsymbol{\beta}}. \tag{5.23}$$

Due to the orthogonality of the $\mathbf{s}_i$, the net contribution of each regressor can appear as a separate error reducing term. Since each such term is in fact the squared cosine value of the angle between the vector $\mathbf{s}_i$ and $\mathbf{y}$, the regressors that are most coincident with the independent variable contribute most to the error reduction. This observation may motivate a procedure for the selection of regressors and was originally introduced for the purpose of subset selection [121]. So the impact of each support vector can be quantified, which allows to rank the points. Then for pruning the least error-influencing point will be omitted in the experiments (denoted 'dQR' for reference to this method).

5.2.4 Approximating representers

A recent subset selection procedure [13] in the kernel domain is unsupervised in nature and aims at finding (in a greedy fashion) a span of feature vectors that are most similar to the other remaining ones. Several additional computational improvements of this procedure were extensively discussed in [24].

The similarity is measured by the distance between a remaining feature vector $\varphi(\mathbf{x}_i)$ and an arbitrary linear combination of previously found feature vectors (assumed to be gathered in the set S):

$$\min_{\boldsymbol{\beta}} \|\varphi(\mathbf{x}_i) - \Phi_S^T\boldsymbol{\beta}\|^2 / \|\varphi(\mathbf{x}_i)\|^2. \tag{5.24}$$

Computing the derivative with respect to $\boldsymbol{\beta}$ and equating to zero yields $\boldsymbol{\beta} = K_{SS}^{-1}\ \mathbf{k}_{Si}$, where $K_{SS} = \Phi_S\Phi_S^T$ and $\mathbf{k}_{Si} = \Phi_S\ \varphi(\mathbf{x}_i)$. Substitution into (5.24) yields after some algebra the value

$$K_{ii} - \mathbf{k}_{Si}^T\ K_{SS}^{-1}\ \mathbf{k}_{Si}. \tag{5.25}$$

This criterion selects points by which all the remaining features can be best approximated. The datapoints are thereby again ranked in importance and for pruning the last one will be omitted here (denoted 'span' in the experiments).

5.3 Experiments with Pruning

In this section we describe the experimental comparison of the above 6 pruning schemes. The prediction performance on an *independent test set* will serve as a measure. For reference we include pruning of a randomly selected point (as an upper bound) and pruning of a backward selected point, i.e. the one that yields a model with least error after omitting and retraining the model on the remaining points (as a lower bound).

Backward selection can be expected to perform well, but is an overly expensive method. We applied the algorithms to an artificial sinc function estimation problem and several benchmarks from the UCI machine learning repository [15]. In every experiment we standardized the data to zero mean and unit variance, we used the common Gaussian kernel and determined the tuning parameters h^2 and γ with standard 10-fold cross-validation. We randomized (25 times) over selection of data points for the training set and test set (both having always a predetermined size-see below for the data sets description).

In every experiment we measured the mean square error on an independent test set, while every time one training sample has been left out according to the criterion of the algorithm. This sets out the averaged mean square error on an independent test set against training set reduction (in %). We can draw the following observations from the experiments.

- **Sinc**. For the mexican hat function we employed a training set of size 100 of uniformly spaced samples of a sinc with added Gaussian noise with standard deviation $\sigma = 0.2$, while the test set size was 50. From Figure 5.1 we can observe that leaving out the sample with smallest weighed support values ($\gamma \neq \infty$) is definitely performing worse than a random pruning. The other methods perform all well and better than 'sv'. The sv pruning based on the assumption $\gamma = \infty$ leads surprisingly to a good result even if on average only $\gamma = 356.5711$ so that the case is in fact not applicable.

- **Boston housing**. This data set [77] consists of 506 cases having 13 input variables. The aim is to predict the housing prices. We picked at random a training set of size 400 and a test set of size 106. The tuning parameters were $h^2 = 15.96$ and $\gamma = 32.35$. From the table we observe that pruning by the dQR method yields excellent results, even outperforming backward selection. Again the 'weighed sv $\gamma \neq \infty$' pruning performs worse than random pruning. The other methods perform all well and better than 'sv'.

- **Machine cpu**. This data set consists of 209 cases having 6 input variables. The aim is to predict the relative cpu performance. We picked at random a training set of size 200 and a test set of size 109. The tuning parameters were $h^2 = 15.03$ and $\gamma = 1.74$. From the table we observe the same

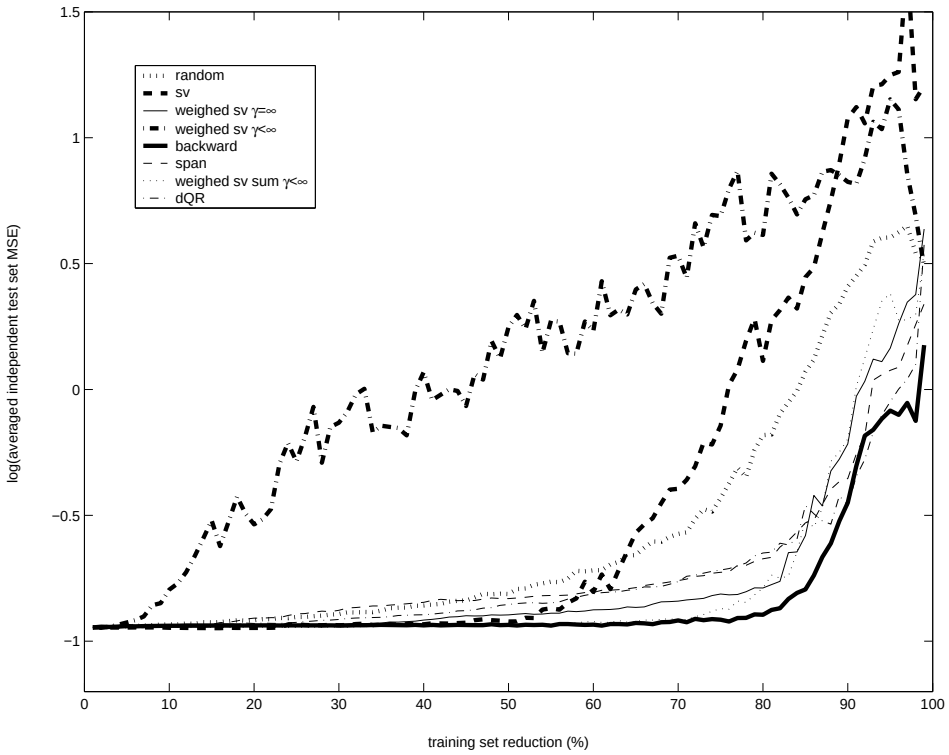


Figure 5.1: Averaged mean squared error on an independent test set versus training set size reduction (%) for a 1-dimensional sinc function, with noise level $\sigma^2 = 0.2$.

behaviour as for the Boston data set. The method 'weighed sv $\gamma = \infty$' yields efficient pruning although γ is small and the formula does not apply in that case (in theory).

- **Servo**. This data set consists of 167 cases having 4 input variables. The aim is to predict the rise time of a servo mechanism. We picked at random a training set of size 100 and a test set of size 67. The tuning parameters were $h^2 = 16.20$ and $\gamma = 91.07$. From the table we observe again that the method 'weighed sv $\gamma = \infty$' yields most efficient pruning, and that our suggested 'weighed sv sum $\gamma \neq \infty$' pruning is close to this.

- **Auto**. This data set consists of 392 cases having 7 input variables. The aim is to predict the fuel consumption. We picked at random a training set of size 300 and a test set of size 92. The tuning parameters were $h^2 = 56.57$ and $\gamma = 58.44$. From the table we observe the same line of behaviour as on the Servo data set.

Overall, we can observe that leaving out the sample with smallest weighed

support values for the case $\gamma \neq \infty$ is performing too often worse than a random pruning. The other methods perform all consistently better than random. Our suggested alternative of *summing* the errors in case $\gamma \neq \infty$ may safely be regarded as a very significant improvement of the pruning method of [39]. The unweighed support value pruning is achieving moderate to good results. The support value pruning based on the assumption $\gamma = \infty$ leads -surprisingly- to excellent results even though γ has small values in all experiments. Quite remarkable whereas this pruning criterion is in fact never applicable from the theory point of view of [39]. From the table we further observe that pruning by the dQR method yields also very good results and the unsupervised pruning method 'span' performs well.

All data set characteristics, selected tuning parameters and pruning results have been reported in the overview table for several pruning reduced training sets. The best results are typeset in bold face (excluding the performance of backward selection).

data set	dim	train	test	h^2	γ
Boston	13	400	106	15.96	32.35
Machine	6	200	109	15.03	1.74
Servo	4	100	67	16.20	91.07
Auto	7	300	92	56.57	58.44

Table 5.1: Overview of the experimental parameters used for the datasets

set	log(mean square error)	**training set reduction by pruning**						
	pruning method	90 %	80 %	60 %	50 %	40 %	20 %	10 %
Boston	random	-0.924	-1.175	-1.409	-1.413	-1.670	-1.849	-1.767
	sv	-1.137	-1.350	-1.801	-1.793	-1.873	-1.962	-1.973
	weighed sv $\gamma = \infty$	**-1.810**	**-2.068**	-2.038	-2.022	-2.049	-1.989	-1.985
	weighed sv $\gamma < \infty$	-0.101	-0.304	-1.274	-1.290	-1.406	-1.456	-1.946
	backward	-1.597	-1.890	-1.970	-1.955	-1.966	-1.982	-1.983
	span	-1.196	-1.759	-2.012	-1.996	-2.013	**-2.059**	-2.011
	weighed sv sum $\gamma < \infty$	-1.406	-2.032	-1.970	-1.986	-1.990	-1.980	-1.979
	dQR	-1.382	-1.811	**-2.071**	**-2.063**	**-2.059**	-2.025	**-2.013**
Machine	random	0.600	0.585	0.385	0.424	0.195	0.113	0.069
	sv	0.434	0.184	0.082	0.069	0.063	0.045	-0.041
	weighed sv $\gamma = \infty$	0.321	0.267	0.088	**0.064**	**0.039**	**0.028**	0.024
	weighed sv $\gamma < \infty$	0.696	0.533	0.552	0.504	0.058	0.046	0.038
	backward	0.154	0.035	-0.022	-0.047	-0.026	-0.055	-0.066
	span	0.621	**0.139**	0.092	0.075	0.066	0.051	0.052
	weighed sv sum $\gamma < \infty$	0.458	0.210	0.083	0.069	0.070	0.040	**-0.041**
	dQR	**0.299**	0.147	**0.078**	0.072	0.060	0.051	0.052
Servo	random	-0.468	-0.695	-0.953	-1.132	-1.261	-1.451	-1.476
	sv	0.227	-0.635	-0.442	-1.322	-1.431	-1.499	-1.514
	weighed sv $\gamma = \infty$	-0.343	**-1.341**	**-1.562**	**-1.559**	**-1.539**	-1.511	-1.511
	weighed sv $\gamma < \infty$	-0.042	-0.851	-1.441	-1.444	-1.345	-1.501	-1.510
	backward	-0.731	-1.282	-1.519	-1.485	-1.506	-1.506	-1.514
	span	-0.226	-0.467	-1.130	-1.290	-1.467	**-1.519**	-1.539
	weighed sv sum $\gamma < \infty$	**-0.777**	-1.332	-1.421	-1.529	-1.534	-1.510	**-1.512**
	dQR	-0.078	-1.207	-1.353	-1.417	-1.410	-1.482	-1.475
Auto	random	-1.834	-2.115	-2.197	-2.277	-2.233	-2.316	-2.340
	sv	-0.612	-2.125	-2.317	-2.311	-2.318	-2.339	-2.333
	weighed sv $\gamma = \infty$	**-2.206**	**-2.257**	**-2.316**	-2.303	**-2.348**	-2.347	-2.339
	weighed sv $\gamma < \infty$	-1.936	-2.321	-2.269	-2.262	-2.325	-2.307	-2.310
	backward	-2.169	-2.328	-2.365	-2.379	-2.327	-2.312	-2.310
	span	-1.975	-2.157	-2.265	**-2.327**	-2.332	-2.344	-2.334
	weighed sv sum $\gamma < \infty$	-2.149	-2.227	-2.304	-2.312	-2.339	-2.337	-2.336
	dQR	-2.086	-2.177	-2.289	-2.311	-2.316	-2.363	**-2.345**

Table 5.2: Numerical results for the comparison of pruning algorithms on different datasets. Shown are log(MSE) values versus reduction in training set size (in %). The best result was typeset each time in bold.

5.4 Selection criteria for online LS-SVM's

Many real-life machine learning problems appear mostly naturally as an online process rather than batch learning problems. Often this is because data is intrinsically collected continuously over time or underlying driving concepts may evolve in time. Furthermore, considerations like memory usage and time constraints itself may render the need for an online sparse solution.

In kernel based classification and function approximation the sparseness (i.e. limited relatively small number of kernel terms) of the approximator is an important issue, since it allows faster evaluation of new data points. These points that constitute the sparse set are usually called support vectors. In Vapnik's Support Vector Machine (SVM) [184] the sparseness is built-in due to the ϵ-insensitive loss function that outrules errors of points inside a 'tube' around the approximated function. This results in a quadratic programming problem, which requires large training times for the optimization of the model parameters. Yet, over the years, this problem has been surmounted largely through significant effort spent on the development of online SVM learning [21, 103, 96, 169].

In Least Squares Support Vector Machines (LS-SVM) [167] a quadratic loss function is used instead, and the optimization problem comes down to solving only a linear set of equations, but unfortunately the sparseness is lost. To effectively equip the LS-SVM with a sparse representation, one can consider a representation based on a reduced set of vectors, adaptively selected during online learning. It is a pathway that has not yet been explored to its full extent. Most of the recent focus to make LS-SVM deal with that problem has been merely on imposing sparseness on a batch solved system by means of pruning schemes [163, 39, 142] or via Fixed-Size approaches [165, 87, 22, 100] which work with reduced sets, or deal only with an aspect of online LS-SVM, like the recursive update [28, 110, 75].

In the following sections we treat LS-SVM in terms of a relatively small, fixed set of basis vectors, in an online setting where iterative updating (inclusion of a new point) is alternated with downdating (exclusion) according to a selection criterion on the points in the sparse set. In particular, we gather three updating approaches (extension, approximation and recursive) and three downdating approaches (truncation, approximation, recursive) in a structured scheme and show how the latter induce naturally selection criteria based on error measures in both data space and feature space. We make comparisons on stationary benchmark data sets of some up/downdating combinations for the various selection criteria, and conclude which measure to use for performant prediction with online LS-SVM.

Instead of the 'full' solution of eq. (3.13)

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T\varphi(\mathbf{x}) + b = \sum_{i=1}^{n} \alpha_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b, \qquad (5.26)$$

based on n kernel terms, a *sparse* approximation will consist of $m \ll n$ kernel basis functions:

$$f = \alpha_0 + \sum_{i=1}^{m} \alpha_i \phi_i. \qquad (5.27)$$

where for convenience of notation we denote the kernel basis functions (or features, representers) $\phi_i := k(\mathbf{x}_i, \cdot)$ and the bias term $\alpha_0 := b$.

5.5 Updating

Starting from a model with m representers, like in eq. (5.27), adding a basis vector yields an expanded model with one extra parameter:

$$f_+ = \beta_0 + \sum_{i=1}^{m} \beta_i \phi_i + \beta_{m+1}\phi_{m+1}. \qquad (5.28)$$

A first issue is how to determine the $(m+2)$ parameter entries in vector $\boldsymbol{\beta}$. We discuss a few possible approaches in the sequel and also mention some update quality measures.

5.5.1 Extension

The most simple form of updating might be extension: one adds a kernel function with a certain coefficient and the other coefficients are retained.

As a direct simple estimate for the extra coefficient β_{m+1}, one can take the optimality condition $\boldsymbol{\alpha}_i = \nu \mathbf{e}_i$ from eq. (3.7) as a guideline, and thus set $\beta_{m+1} = f(\mathbf{x}_{m+1}) - y_{m+1}$. We can then formally write for the new coefficients that

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\alpha} \\ f(\mathbf{x}_{m+1}) - y_{m+1} \end{bmatrix}. \qquad (5.29)$$

This approach requires negligible computational requirements (one point evaluation) and can also be alternatively interpreted in a gradient descent framework, where one seeks the solution f of the equation $y_i = L_i f$ in a least squares sense, with L_i a functional in the RKHS [51]. Dodd *et al.* employ a functional approximation approach to describe on-line learning with kernels.

5.5.2 Approximation

A more accurate solution for the coefficient β_{m+1} is to determine a least squares optimal value, by solving

$$\min_{\beta \in \mathbb{R}} \|\mathbf{t}\beta - \mathbf{z}\|^2, \tag{5.30}$$

where the column vectors correspond to

$$\begin{aligned} \mathbf{t} &= [A_+]_{:,m+1} \\ \mathbf{z} &= \begin{bmatrix} 0 & \mathbf{y}^T & y_{m+1} \end{bmatrix}^T - [A_+]_{:,0:m}\boldsymbol{\alpha} \end{aligned} \tag{5.31}$$

where we used Matlab notation; $\mathbf{t}$ is the $(m+1)$th column of A_+ and $\mathbf{z}$ is the matrix A_+ without the last column. This yields

$$\beta_{m+1} = \frac{\mathbf{t}^T\mathbf{z}}{\|\mathbf{t}\|^2}, \tag{5.32}$$

at the cost of $O(m^2)$ operations.

5.5.3 Recursive

The batch solution of the linear system (3.9), when using a subset of $(m+1)$ representers to compute $\boldsymbol{\beta}$, takes $O(m^3)$ operations for the matrix inversion. By making use of the previous solution of the m-term model, a recursive more efficient updating can save computational effort.

Let us denote the matrix of the left-hand side of the LS-SVM system

$$\left[\begin{array}{c|c} 0 & \mathbf{1}^T \\ \hline \mathbf{1} & K + \nu^{-1}I \end{array}\right] \left[\begin{array}{c} \mathrm{b} \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{y} \end{array}\right], \tag{5.33}$$

as the $(m+1) \times (m+1)$ matrix A. Then an extension with an extra kernel based function yields a $(m+2) \times (m+2)$ matrix

$$A_+ = \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & a \end{bmatrix}, \tag{5.34}$$

where $\mathbf{a}$ is a $m+1$-sized column vector. The same extension is applicable to the kernel matrix K, that becomes K_+, when another representer is included. A recursive update of the inverse of A_+ is given by the well-known block matrix formula [73]:

$$\begin{aligned} A_+^{-1} &= \begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & a \end{bmatrix}^{-1} \\ &= \begin{bmatrix} A^{-1} + A^{-1}\mathbf{a}s_A^{-1}\mathbf{a}^T A^{-1} & -A^{-1}\mathbf{a}s_A^{-1} \\ -s_A^{-1}\mathbf{a}^T A^{-1} & s_A^{-1} \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + s_A^{-1}\,\mathbf{q}\mathbf{q}^T \end{aligned} \tag{5.35}$$

where the scalar s_A is the *Schur complement* of A:

$$s_A := a - \mathbf{a}^T A^{-1} \mathbf{a} = 1/b \tag{5.36}$$

where $b := \left[A_+^{-1}\right]_{(m+2)(m+2)}$ and the rank-one update matrix is composed with the vector

$$\mathbf{q} := \begin{bmatrix} A^{-1}\mathbf{a} \\ -1 \end{bmatrix}. \tag{5.37}$$

If in formula (5.35) the matrix-vector multiplications are performed first, a gain in computation can be achieved such that only $O(m^2)$ operations are needed. So finally we obtain for the new coefficient vector

$$\boldsymbol{\beta} = A_+^{-1} \begin{bmatrix} 0 \\ \boldsymbol{y} \\ y_{m+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ 0 \end{bmatrix} + \; b \; (\boldsymbol{\alpha}^T \mathbf{a} - y_{m+1}) \; \mathbf{q}. \tag{5.38}$$

The result is completely equivalent with the batch LS-SVM solution and all information of the new sample is incorporated into the model.

Updating	$\boldsymbol{\beta}$
Extension	$\begin{bmatrix} \boldsymbol{\alpha} \\ f(\mathbf{x}_{m+1}) - y_{m+1} \end{bmatrix}$
Approximation	$\begin{bmatrix} \boldsymbol{\alpha} \\ \frac{\mathbf{t}^T \mathbf{z}}{\|\mathbf{t}\|^2} \end{bmatrix}$
Recursive	$\begin{bmatrix} \boldsymbol{\alpha} \\ 0 \end{bmatrix} + b \; (\boldsymbol{\alpha}^T \mathbf{a} - y_{m+1}) \; \mathbf{q}$

Table 5.3: Overview of the expressions of the new coefficient vector $\boldsymbol{\beta}$ for the three different updating schemes.

5.5.4 Update criteria

A new basis vector brings in novelty into the model, to a certain degree. Particularly, in case of a nonzero sparsity window, we need to assess the difference between the models and use it to decide whether to update or not. The computation typically takes $O(m^2)$ operations. (For a stochastical approach of novelty detection we refer to [157].)

- In *feature space* we can measure, with the RKHS norm (excluding the bias), for instance the length of the difference vector between the newly updated model f_+ and the previous model f:

$$\begin{aligned} \|\boldsymbol{\Delta}_+\|^2_{\mathcal{H}_k} &:= \|f_+ - f\|^2_{\mathcal{H}_k} \\ &= \|f_+\|^2_{\mathcal{H}_k} + \|f\|^2_{\mathcal{H}_k} - 2\langle f_+, f\rangle_{\mathcal{H}_k} \\ &= \boldsymbol{\beta}^T K_+ \boldsymbol{\beta} + \boldsymbol{\alpha}^T K \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T \left[\begin{array}{cc} K & \mathbf{k} \end{array}\right] \boldsymbol{\beta}. \end{aligned} \tag{5.39}$$

  Comparison with a threshold can serve as a *global* indication of the effect of a new input on the model. The optimal solution minimizing this norm with respect to $\boldsymbol{\beta}$ yields

$$\boldsymbol{\beta} = K_+^{-1} \left[\begin{array}{c} K \\ \mathbf{k}^T \end{array}\right] \boldsymbol{\alpha} = \left[\begin{array}{c} \boldsymbol{\alpha} \\ 0 \end{array}\right], \tag{5.40}$$

  which is unfortunately only a trivial solution. Nevertheless, in the downdating, this measure will yield more useful results.

- In the *data space*, we could for example assess the differences of evaluations by the corresponding models at the current $m+1$ support vectors, resulting in an error vector $\mathbf{e}$ that we can express in a general form as:

$$\left[\begin{array}{c} \mathbf{e}_m \\ e_{m+1} \end{array}\right] = \hat{\mathbf{y}}_+ - \hat{\mathbf{y}} = \left[\begin{array}{ccc} \mathbf{1} & K & \mathbf{k} \\ 1 & \mathbf{k}^T & k \end{array}\right] \boldsymbol{\beta} - \left[\begin{array}{cc} \mathbf{1} & K \\ 1 & \mathbf{k}^T \end{array}\right] \boldsymbol{\alpha} \tag{5.41}$$

  where the entries are computed as

$$\begin{aligned} e_i &= (\hat{\mathbf{y}}_+)_i - (\hat{\mathbf{y}})_i \\ &= f_+(\mathbf{x}_i) - f(\mathbf{x}_i) \\ &= \left(\beta_0 + \sum_{j=1}^{m+1} \beta_j k(\mathbf{x}_i, \mathbf{x}_j)\right) - \left(\alpha_0 + \sum_{j=1}^{m} \alpha_j k(\mathbf{x}_i, \mathbf{x}_j)\right). \end{aligned} \tag{5.42}$$

  If for instance the L_2 norm of the error (5.41) is larger than a certain chosen treshold, the new sample improves the prediction significantly and one keeps f_+ as the current estimate. The optimal solution minimizing this norm with respect to $\boldsymbol{\beta}$ yields

$$\boldsymbol{\beta} = \left[\begin{array}{ccc} \mathbf{1} & K & \mathbf{k} \\ 1 & \mathbf{k}^T & k \end{array}\right]^+ \left[\begin{array}{cc} \mathbf{1} & K \\ 1 & \mathbf{k}^T \end{array}\right] \boldsymbol{\alpha}, \tag{5.43}$$

  which is unfortunately an expensive cost of $O(m^3)$ operations.

Both for extension and approximation, the error in feature space amounts to $\|\boldsymbol{\Delta}\|^2_{\mathcal{H}_k} = k\ \beta^2_{m+1}$, while the error in data space of eq. (5.41) yields $\mathbf{e} = k/\beta_{m+1}$. So the corresponding update quality measures turn out to be proportional to

the kernel value $k := k(\mathbf{x}_{m+1}, \mathbf{x}_{m+1})$ of the point and the magnitude of the estimated coefficient, which agrees with intuition. For the recursive updating solution in feature space the error of eq. (5.39) amounts to

$$\|\boldsymbol{\Delta}_+\|^2_{\mathcal{H}_k} = \mathbf{q}^T_{1:m+1}\ K_+\ \mathbf{q}_{1:m+1}, \tag{5.44}$$

while the error in data space of eq. (5.41) yields

$$\mathbf{e} = \begin{bmatrix} \mathbf{0} \\ -1/b \end{bmatrix}. \tag{5.45}$$

In order to minimize this error, the best points to select are thus those with a small Schur complement of A, and as such it forms an interesting subset selection approach.

5.6 Downdating

To decrease the dimension of the support vector set of f_+, one excludes one basis vector. This yields a reduced model with one parameter less:

$$f_- = \gamma_0 + \sum_{i=1}^{m} \gamma_i \phi_i \tag{5.46}$$

where the $(m+1)$ parameters in $\boldsymbol{\gamma}$ are to be determined again. Next to the issue of determining the coefficients of the new, decreased expansion, another issue is how to select the basis vector to leave out. Both issues are not necessarily independent of each other and ideally each point has to be assessed for selection, which involves downdating quality measures, as discussed in the next section.

5.6.1 Downdate criteria

Like in subsection we can again start from the error measure in feature and data space.

Error in feature space

In *feature space* we can measure the length of the difference vector between the newly updated model f_+ and the downdated model f_-:

$$\begin{aligned} \|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k} &:= \|f_+ - f_-\|^2_{\mathcal{H}_k} \\ &= \|f_+\|^2_{\mathcal{H}_k} + \|f_-\|^2_{\mathcal{H}_k} - 2\langle f_+, f_-\rangle_{\mathcal{H}_k} \\ &= \boldsymbol{\beta}^T K_+ \boldsymbol{\beta} + \boldsymbol{\gamma}^T K \boldsymbol{\gamma} - 2\boldsymbol{\gamma}^T \begin{bmatrix} K & \mathbf{k} \end{bmatrix} \boldsymbol{\beta}. \end{aligned} \tag{5.47}$$

Clearly this expression is related to (5.39), but now the error of a downdated model on a certain representer, relative to an updated model, is considered. Intuitively, the length (5.47) indicates the degree that the solution is 'lifted' out of the plane of the m representers by the update. A few more remarks can be made here.

- If the omitted representer of the downdate is the one that had been added in the updating, and we suppose information of the new sample has been only partially excluded in the downdating process, then the downdating can be considered as a form of updating. In this respect (5.47) is yet another update quality measure. Then it measures the novelty of a new sample and comparison with a threshold can indicate the supportive character of the added representer, which would be useful for the nonzero sparsity window case.
- If the omitted representer of the downdate is different from the one that had been added in the updating, (5.47) is a means to assess the error of the downdate and it yields a selection criterion when the downdate for every representer is the subset is considered.
- A third measure, indicating the degree of 'rotation' that a new input exerts on the previous solution,

$$\begin{aligned} \|\boldsymbol{\Delta}_-\|^2_{\mathcal{H}_k} &= \|f - f_-\|^2_{\mathcal{H}_k} \\ &= \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + \boldsymbol{\gamma}^T K \boldsymbol{\gamma} - 2\boldsymbol{\gamma}^T K \boldsymbol{\alpha}, \end{aligned} \tag{5.48}$$

  completes the vector triangle between the old model f, the updated model f_+ and the downdated model f_-:

$$\boldsymbol{\Delta}_+ = \boldsymbol{\Delta}_- + \boldsymbol{\Delta}_r. \tag{5.49}$$

  The difference vector $\boldsymbol{\Delta}_+$ can be decomposed into a component within the subspace spanned by a smaller set of m representers and a residual component along the excluded representer. The relative lengths of these components indicate the effect of the choice of representers and quantify the model differences in feature space.
- If the length (5.48) is large, the new input may be important, even if the residual component is small. Instead of only taking into account the length of the residual component from eq. (5.47), a ratio to (5.48) seems more useful.

Error in data space

In the *data space*, we could for example assess the differences of evaluations by the corresponding models at the current $m + 1$ support vectors, resulting in a

error vector $\mathbf{e}$ that we can express in a general form as:

$$\begin{bmatrix} \mathbf{e}_m \\ e_{m+1} \end{bmatrix} = \hat{\mathbf{y}}_+ - \hat{\mathbf{y}}_- = \begin{bmatrix} \mathbf{1} & K & \mathbf{k} \\ 1 & \mathbf{k}^T & k \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \mathbf{1} & K \\ 1 & \mathbf{k}^T \end{bmatrix} \boldsymbol{\gamma}, \tag{5.50}$$

where the entries are computed as

$$\begin{aligned} e_i &= (\hat{\mathbf{y}}_+)_i - (\hat{\mathbf{y}}_-)_i \\ &= f_+(\mathbf{x}_i) - f_-(\mathbf{x}_i) \\ &= \left(\beta_0 + \sum_{j=1}^{m+1} \beta_j k(\mathbf{x}_i, \mathbf{x}_j) \right) - \left(\gamma_0 + \sum_{j=1}^{m} \gamma_j k(\mathbf{x}_i, \mathbf{x}_j) \right). \end{aligned} \tag{5.51}$$

Taking the 2-norm of an error vector will take $O(m^2)$ operations, while taking the sum of the absolute values of the entries is most cheap.

The optimal solution minimizing this norm with respect to $\boldsymbol{\gamma}$ yields

$$\boldsymbol{\gamma} = \begin{bmatrix} \mathbf{1} & K \\ 1 & \mathbf{k}^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{1} & K & \mathbf{k} \\ 1 & \mathbf{k}^T & k \end{bmatrix} \boldsymbol{\beta}. \tag{5.52}$$

Using the recursive generalized form (see formula (5.85)) for the matrix inversion, and performing multiplications in the right order, only $O(m^2)$ operations are required for this solution.

In the next sections we discuss specific downdating approaches which have a different impact on the model quality and on the degree they cancel out information of a selected sample. The downdating quality measures induced by the downdating methods will induce useful expressions for selection.

5.6.2 Truncation

The most simple form of reduction might be truncation: one simply discards a certain basis vector and the remaining coefficients are retained. If we switch a selected vector with the last vector and apply truncation of the $(m+1)$th term, then we formally have that

$$\boldsymbol{\gamma} = \boldsymbol{\beta}_{0:m}. \tag{5.53}$$

This approach requires negligible computational requirements.

Selection by truncation

Requiring that a minimal error is introduced, downdating quality measures of subsection 5.6.1 give a direct criterion for selection of a point that can be omitted.

In feature space the error of eq. (5.47) amounts to

$$\|\mathbf{\Delta}_r\|^2_{\mathcal{H}_k} = k\ \beta^2_{m+1}, \tag{5.54}$$

while the error in data space of eq. (5.50) yields

$$\mathbf{e} = \beta_{m+1} \begin{bmatrix} \mathbf{k} \\ k \end{bmatrix}. \tag{5.55}$$

Clearly the entries of the error can be obtained without any extra computation, which makes the selection cheap. Both error results indicate that the degradation is (i) proportional to the parameter coefficient of the truncated representer and (ii) proportional to the kernel value (which is often a distance). Both approaches have separately been treated in literature as follows.

Selection by minimal coefficient

Pruning away the representer with the smallest coefficient [164] is a reasonable choice, based on the assumption that the representer with the smallest weight has the least effect on the model. An advantage is that it is compuationally inexpensive, but a disadvantage is that if the kernel is not with a local support, the effect on the model is not negligible any more.

Selection by entropy

Selection of samples which have a smallest sum of kernel column entries can be shown to lead to a subset with high entropy, and thus uniform density. This approach has been introduced as a selection mechanism (subsampling from a large dataset) with Fixed-Size LS-SVM [165].

The appropiate criterion is borrowed from information theory where information content and redundancy can be quantified through the concept of (generalized Renyi) entropy [30]:

$$H_r(X) = \frac{1}{1-r} \log \left(\int_X p(\mathbf{x})^r d\mathbf{x} \right) \tag{5.56}$$

where $p(\mathbf{x})$ is the pdf of a continuous random variable X, r is the order with $0 < r < +\infty$ and $r \neq 1$. For $r \to 1$ one obtains the Shannon entropy

$$H(p) = -\int_X p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}. \tag{5.57}$$

Yet, the latter form is not easily amenable to simple estimation algorithms, so the generalization is preferred, and we further restrict to the case of order $r = 2$ because it allows easy computation and interpretation in the context of kernels.

In practice one estimates the pdf $p(\mathbf{x})$ of the subset by a density estimation with kernels [66]:

$$\hat{p}(\mathbf{x}) = -\frac{1}{m}\sum_{i=1}^{m} k(\mathbf{x}_i, \mathbf{x}). \tag{5.58}$$

Substitution in the *quadratic Renyi entropy* then yields

$$H_2(p) = -\log \int_{X_m} \hat{p}(\mathbf{x})^2 d\mathbf{x} \approx -\log \frac{1}{m^2}\sum_{i,j=1}^{m} k(\mathbf{x}_i, \mathbf{x}_j) = -\log \frac{1}{m^2}\mathbf{1}^T K_{mm}\mathbf{1} \tag{5.59}$$

where one approximates the integral by a sum and $\mathbf{1}$ is a column vector of ones. Observe that the sum of the entries of the kernel matrix K_{mm} is a measure for uniformity. This is a rather inexpensive criterion that selects points with maximal column sum in the kernel matrix, in order to omit them. It only requires thus m kernel evaluations, and it is applicable to any kernel applicable for density estimation.

5.6.3 Approximation

One level further, the loss in model quality by truncation of one representer ϕ_i may be compensated by making an approximation of one representer by remaining ones, or an approximation by projection of the model on a subset, based on some criterion.

Instead of plainly truncating, one could substitute the representer with a linear combination of the remaining vectors, a concept used in [13]. This leads to an adjustment of the model parameters that does not degrade the model as much, but implies an extra computational cost which is unavoidable [152] .

Let's assume that the selected element is placed as the last, $(m+1)$th term. To find the linear combination we can optimize the length of the residual vector in a least squares sense

$$\min_{\boldsymbol{\xi}\in\mathbb{R}^m} \|\phi_{m+1} - \sum_{j=1}^{m} \xi_j \phi_j\|^2_{\mathcal{H}_k}. \tag{5.60}$$

Setting the derivative with respect to $\boldsymbol{\xi}$ equal to zero yields the optimal solution

$$\boldsymbol{\xi} = K^{-1}\mathbf{k}, \tag{5.61}$$

where K is the $m \times m$ kernel matrix with the m previous kernels and

$$\mathbf{k} := [k(\mathbf{x}_1, \mathbf{x}_{m+1}), k(\mathbf{x}_2, \mathbf{x}_{m+1}), \ldots, k(\mathbf{x}_m, \mathbf{x}_{m+1})]^T. \tag{5.62}$$

Substitution in the updated model with $m+1$ representers,

$$f_+ = \beta_0 + \sum_{i=1}^{m} \beta_i \ \phi_i + \beta_{m+1} \ \phi_{m+1}, \tag{5.63}$$

yields then a reduced model based on m representers:

$$f_- = \beta_0 + \sum_{i=1}^{m} (\beta_i + \beta_{m+1}\xi_i)\ \phi_i = \beta_0 + \sum_{i=1}^{m} \gamma_i\ \phi_i. \tag{5.64}$$

where the $\boldsymbol{\gamma}$ vector is determined as

$$\boldsymbol{\gamma} = \begin{bmatrix} \beta_0 \\ \boldsymbol{\beta}_{1:m} + \beta_{m+1}\ \boldsymbol{\xi} \end{bmatrix} = \boldsymbol{\beta}_{0:m} + \beta_{m+1} \begin{bmatrix} 0 \\ \boldsymbol{\xi} \end{bmatrix}. \tag{5.65}$$

- In the above only the coefficients of the representers have been computed and the bias γ_0 equals β_0. To determine an optimal bias one additionally can solve

$$\min_{\gamma \in \mathbb{R}} \| \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \gamma - \left(\begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} - A_{:,1:m}\boldsymbol{\gamma} \right) \|^2, \tag{5.66}$$

  which yields in $O(m^2)$ operations

$$\gamma_0 = \frac{\mathbf{1}^T}{m} \left(\begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} - A_{:,1:m}\boldsymbol{\gamma} \right). \tag{5.67}$$

- The new coefficient vector $\boldsymbol{\gamma}$ can be obtained via different criteria. If we minimize eq. (5.47) with respect to $\boldsymbol{\gamma}$ one obtains again the same solution of eq. (5.65). Three other equivalent formulations can be immediately pointed out for eq. (5.60). Indirectly it is equivalent to requiring that the residual between the representer and the linear combination are orthogonal to the $j = 1, \ldots, m$ representers, which equals:

$$\langle \phi_j, \phi_{m+1} - \sum_{i=1}^{m} \gamma_i \phi_i \rangle_{\mathcal{H}_k} = 0. \tag{5.68}$$

  Secondly, a direct approach to determine $\boldsymbol{\gamma}$ is to substitute not just the newly added representer, but the new *model* by a reduced model. To find the linear combination we can optimize again the length of the residual vector in a least squares sense

$$\min_{\boldsymbol{\gamma} \in \mathbb{R}^m} \| \sum_{i=1}^{m+1} \beta_i \phi_i - \sum_{i=1}^{m} \gamma_i \phi_i \|^2_{\mathcal{H}_k}, \tag{5.69}$$

  and it yields both again the same solution as in eq. (5.65)

$$\boldsymbol{\gamma} = \begin{bmatrix} \beta_0 \\ K^{-1} \begin{bmatrix} K & \mathbf{k} \end{bmatrix} \boldsymbol{\beta}_{1:m} \end{bmatrix} = \boldsymbol{\beta}_{0:m} + \beta_{m+1} \begin{bmatrix} 0 \\ \boldsymbol{\xi} \end{bmatrix} \tag{5.70}$$

  Thirdly, on its turn, the latter formulation is equivalent to requiring that the residuals between the models are orthogonal to the span of m representers, which equals

$$\langle \phi_i, \sum_{j=1}^{m+1} \beta_j \phi_j - \sum_{l=1}^{m} \gamma_l \phi_l \rangle_{\mathcal{H}_k} = 0. \tag{5.71}$$

Selection by approximation

In feature space the error of eq. (5.47) yields

$$\begin{aligned} \|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k} &= k\ \beta^2_{m+1} - 2\ \boldsymbol{\xi}^T\mathbf{k}\ \beta^2_{m+1} + \boldsymbol{\xi}^T K\boldsymbol{\xi}\ \beta^2_{m+1} \\ &= (k - \mathbf{k}^T\boldsymbol{\xi})\ \beta^2_{m+1}. \end{aligned} \tag{5.72}$$

The error in data space of eq. (5.50) yields

$$\begin{aligned} \mathbf{e}_m &= \left[\begin{array}{ccc} \mathbf{1} & K & \mathbf{k} \end{array}\right]\boldsymbol{\beta} - \left[\begin{array}{cc} \mathbf{1} & K \end{array}\right]\boldsymbol{\gamma} \\ &= (\beta_0 - \gamma_0)\ \mathbf{1}, \end{aligned} \tag{5.73}$$

while the error at the selected data point is

$$\begin{aligned} e_{m+1} &= \left[\begin{array}{ccc} 1 & \mathbf{k}^T & k \end{array}\right]\boldsymbol{\beta} - \left[\begin{array}{cc} 1 & \mathbf{k}^T \end{array}\right]\boldsymbol{\gamma} \\ &= \left[\begin{array}{ccc} 1 & \mathbf{k}^T & k \end{array}\right]\boldsymbol{\beta} - \big(\gamma_0 + \mathbf{k}^T(\boldsymbol{\beta} + \boldsymbol{\xi})\big) \\ &= (k - \mathbf{k}^T\boldsymbol{\xi})\ \beta_{m+1} + (\beta_0 - \gamma_0) \\ &= \varepsilon_{m+1}\ \beta_{m+1} + (\beta_0 - \gamma_0). \end{aligned} \tag{5.74}$$

Thus, apart from a fixed shift due to an eventual different bias, the prediction of the reduced model does not differ with the updated model in the remaining support vectors and all error is due to the omitted point.

One may observe that the ε_{m+1} appeared before in expression (5.36) and it has the form of the Schur complement of K in an extended matrix K_+. One can always permute the jth element with the $(m+1)$th, and in general the residual error for substituting the term $\beta_i\phi_i$ becomes

$$\varepsilon_i\beta_i^2 = \frac{\beta_i^2}{\left[K_+^{-1}\right]_{ii}}. \tag{5.75}$$

Thus $(m+1)$ residuals can be computed, which can be compared and employed as a selection method. In this manner, the element that can be best represented (substituted with least error) by the other vectors is chosen. This approach is particularly cheap since only m operations are needed, assuming the inverse of K_+ is available. Expression (5.75) is also encountered in the field of Gaussian Processes [36], which bear a close connection with LS-SVM. Through optimizing a Kullback-Leibler distance measure, the above error (5.75) with an additional term can be derived, but apparently without much additional improvement for the selection process.

The residual error (5.60) at the optimum of eq. (5.61) amounts to

$$\varepsilon_{m+1} = k - \mathbf{k}^T K^{-1}\mathbf{k}, \tag{5.76}$$

where $k := k(\mathbf{x}_{m+1}, \mathbf{x}_{m+1})$. This value has a geometrical interpretation. When decomposing the new vector $\mathbf{k}$ in an orthogonal and parallel component to

the span of the m other representers, ε_{m+1} is in fact the squared length of the orthogonal component. And the second term of (5.76) equals the squared length of the parallel component $\sum_{i=0}^{m} \xi_i \phi_i$:

$$\begin{aligned} \|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k} &= \|f_+\|^2_{\mathcal{H}_k} - \langle f_+, f_- \rangle_{\mathcal{H}_k} \\ &= \boldsymbol{\beta}^T K_+ \boldsymbol{\beta} - \boldsymbol{\gamma}^T \begin{bmatrix} K & \mathbf{k} \end{bmatrix} \boldsymbol{\beta} \end{aligned} \tag{5.77}$$

In [50] this measure is used to assess the novelty of the extra input. Remark that at the optimum the error can also be written as:

$$\begin{aligned} \|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k} &= \boldsymbol{\beta}^T K_+ \boldsymbol{\beta} - \boldsymbol{\beta}^T \begin{bmatrix} K & \mathbf{k} \end{bmatrix}^T K^{-1} \begin{bmatrix} K & \mathbf{k} \end{bmatrix} \boldsymbol{\beta} \\ &= \boldsymbol{\beta}^T \left(K_+ - \tilde{K}_+ \right) \boldsymbol{\beta} \end{aligned} \tag{5.78}$$

where $\tilde{K}_+$ is the well-known Nyström low-rank approximation to K_+ [195].

Selection by employing KPCA

In [152] both the selection of the vector and the solution for ξ were related to the eigenvectors of K_+ in feature space, obtained by KPCA. Including also the variation $j \in \{1, \ldots, m+1\}$ over all vectors, criterion (5.60) becomes

$$\min_{j, \boldsymbol{\xi} \in \mathbb{R}^m} \|\beta_j \phi_j - \sum_{i \neq j} \xi_i \phi_i\|^2_H. \tag{5.79}$$

This function may be regarded as a minimization of the norm of a linear combination of feature vectors. By a change of variables,

$$\begin{aligned} \eta_i &= -\xi_i / \beta_j \qquad \text{for } i \neq j, \\ \eta_j &= 1, \\ \mathbf{u} &= \frac{\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|}, \end{aligned} \tag{5.80}$$

eq. (5.79) transforms into

$$\min_{j, \mathbf{u} \in \mathbb{R}^m, \|\mathbf{u}\|=1} \left| \frac{\beta_j}{u_j} \right|^2 \mathbf{u}^T K_+ \mathbf{u} \tag{5.81}$$

where $u_j = 1/\|\eta\|$. Rather than minimizing this function directly, it may be approximated by observing that the 2nd factor is the Rayleigh quotient, which is minimal for eigenvectors $\mathbf{u}_i$ (i denoting here the column index) and equals then the corresponding eigenvalue λ_i. This reduces the search to locating the minimum element of the $(m+1) \times (m+1)$ matrix E of error values

$$E_{ij} = \left| \frac{\beta_j}{u_{ji}} \right|^2 \lambda_i. \tag{5.82}$$

This will determine which jth basis vector to select that can be best approximated by the others, based on the eigenset of K_+. The selection requires $O(m^3)$ operations.

From (5.80) it follows that, for $i \neq j$, we obtain a vector $\boldsymbol{\xi}$ with entries

$$\xi_i = -\frac{\beta_j}{u_{jj}} u_{ij}. \tag{5.83}$$

such that

$$\boldsymbol{\gamma} = \begin{bmatrix} \beta_0 \\ \boldsymbol{\beta}_{1:m} + \boldsymbol{\xi}. \end{bmatrix} \tag{5.84}$$

5.6.4 Recursive

The recursive updating formula (5.35) shows how to efficiently calculate A_+^{-1} using A^{-1}. However, the formula is not directly useful for a downdate, where we need to recover A^{-1} using A_+^{-1}. For that, one needs another block matrix inversion formula [73] for A_+:

$$\begin{bmatrix} A & \mathbf{a} \\ \mathbf{a}^T & a \end{bmatrix}^{-1} = \begin{bmatrix} S_a^{-1} & -S_a^{-1}\mathbf{a}a^{-1} \\ -a^{-1}\mathbf{a}^T S_a^{-1} & a^{-1} + a^{-1}\mathbf{a}^T S_a^{-1}\mathbf{a}a^{-1} \end{bmatrix} \tag{5.85}$$

where $S_a = A - \mathbf{a}a^{-1}\mathbf{a}^T$ is the Schur complement of a. Denote $B_+ = A_+^{-1}$, then this formula applied to B_+ gives

$$\begin{bmatrix} B & \mathbf{b} \\ \mathbf{b}^T & b \end{bmatrix}^{-1} = \begin{bmatrix} S_b^{-1} & -S_b^{-1}\mathbf{b}b^{-1} \\ -b^{-1}\mathbf{b}^T S_b^{-1} & b^{-1} + b^{-1}\mathbf{b}^T S_b^{-1}\mathbf{b}b^{-1} \end{bmatrix}. \tag{5.86}$$

Because $B_+^{-1} = A_+$ we have that $A = (B - \mathbf{b}b^{-1}\mathbf{b}^T)^{-1}$ and thus for the new coefficients it yields

$$\begin{aligned} \boldsymbol{\gamma} &= \left(B - \mathbf{b}b^{-1}\mathbf{b}^T\right) \begin{bmatrix} 0 \\ \mathbf{y}_{1:m} \end{bmatrix} \\ &= \left(\begin{bmatrix} B & \mathbf{b} \end{bmatrix} - \begin{bmatrix} \mathbf{b}b^{-1}\mathbf{b}^T & \mathbf{b} \end{bmatrix} \right) \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \\ &= \boldsymbol{\beta}_{0:m} - b^{-1}\,\mathbf{b} \begin{bmatrix} \mathbf{b}^T & b \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \\ &= \boldsymbol{\beta}_{0:m} - \frac{\beta_{m+1}}{b}\,\mathbf{b}. \end{aligned} \tag{5.87}$$

The result is equivalent with the batch LS-SVM solution and all information of the new sample is completely excluded from the model and no residual influence of that datapoint is left in the model (which makes it a bit less useful as candidate). It is an often overlooked fact that pruning the LS-SVM with one datapoint comes at virtually no cost at all if the inverse of the full solution is

available, yet recently this has been exploited in the context of leave-one-out cross validation [23]. In that publication an alternative derivation of the above expression for $\boldsymbol{\gamma}$ is given that makes use of the formula (5.35).

Downdating	$\boldsymbol{\gamma}$
Truncation	$\boldsymbol{\beta}_{0:m}$
Approximation	$\boldsymbol{\beta}_{0:m} + \beta_{m+1} \begin{bmatrix} 0 \\ \boldsymbol{\xi} \end{bmatrix}$
Recursive	$\boldsymbol{\beta}_{0:m} - \frac{\beta_{m+1}}{b}\, \mathbf{b}$

Table 5.4: Overview of the expressions of the new coefficient parameter vector $\boldsymbol{\gamma}$ for a downdate of the last element. One can see that the new coefficient vector is generically formed by adding each time a residual to the 'old' parameter vector $\boldsymbol{\beta}_{0:m}$.

Selection by recursion

Substitution of the recursive downdate solution (5.87) into the feature space error (5.47) yields

$$\begin{aligned} \|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k} &= k\ \beta^2_{m+1} + \frac{2}{b}\mathbf{b}^T_{1:m}\mathbf{k}\ \beta^2_{m+1} + \frac{\beta^2_{m+1}}{b^2}\mathbf{b}^T_{1:m}K\mathbf{b}_{1:m} \\ &= \beta^2_{m+1}\left(k + \frac{2}{b}\mathbf{b}^T_{1:m}\mathbf{k} + \frac{1}{b^2}\mathbf{b}^T_{1:m}K\mathbf{b}_{1:m}\right), \end{aligned} \tag{5.88}$$

where b is the $(m+1)$th diagonal element of A^{-1}_+.

And when we evaluate the recursive downdate solution (5.87) into the error

vector (5.48), we obtain

$$\begin{aligned}
\begin{bmatrix} 0 \\ \mathbf{e} \end{bmatrix} &= \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & K_+ \end{bmatrix} \left(\boldsymbol{\beta} - \begin{bmatrix} \boldsymbol{\gamma} \\ 0 \end{bmatrix} \right) \\
&= \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & K_+ \end{bmatrix} \left(B_+ - \begin{bmatrix} B - \mathbf{b}b^{-1}\mathbf{b}^T & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \right) \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \\
&= \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & K_+ \end{bmatrix} \left(-b^{-1} \begin{bmatrix} \mathbf{b} \\ b \end{bmatrix} \begin{bmatrix} \mathbf{b}^T & b \end{bmatrix} \right) \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \\
&= \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & K_+ \end{bmatrix} \left(-\beta_{m+1} b^{-1} \begin{bmatrix} \mathbf{b} \\ b \end{bmatrix} \right) \\
&= -\beta_{m+1} b^{-1} \left(A_+ - \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \nu^{-1} I_{m+1} \end{bmatrix} \right) \begin{bmatrix} \mathbf{b} \\ b \end{bmatrix} \\
&= -\beta_{m+1} b^{-1} \left(\begin{bmatrix} 0 \\ \mathbf{0} \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{b}_{1:m}/\nu \\ b/\nu \end{bmatrix} \right) \\
&= \beta_{m+1} \begin{bmatrix} 0 \\ \mathbf{b}_{1:m}/(\nu b) \\ \frac{1}{\nu} - \frac{1}{b} \end{bmatrix},
\end{aligned} \tag{5.89}$$

where we used eq. (5.26) and formula (5.85). For $\nu = \infty$ the error is located at the omitted point, while for $\nu \neq \infty$ the error is distributed over all points. It is an advantage that the entries of the error require no extra computation, but can be almost directly obtained from the matrix A_+^{-1}. The dependency of the regularization constant of the LS-SVM becomes very explicit in this case.

Selection	$\|\boldsymbol{\Delta}_r\|^2_{\mathcal{H}_k}$	$\mathbf{e}$
Truncation	$k\ \beta_{m+1}^2$	$\beta_{m+1} \begin{bmatrix} \mathbf{k} \\ k \end{bmatrix}$
Approximation	$(k - \mathbf{k}^T\boldsymbol{\xi})\ \beta_{m+1}^2$	$(k - \mathbf{k}^T\boldsymbol{\xi})\ \beta_{m+1}$
Recursive	$\beta_{m+1}^2 \left(k + \frac{2}{b}\mathbf{b}_{1:m}^T\mathbf{k} + \frac{1}{b^2}\mathbf{b}_{1:m}^T K \mathbf{b}_{1:m} \right)$	$\beta_{m+1} \begin{bmatrix} 0 \\ \mathbf{b}_{1:m}/(\nu b) \\ \frac{1}{\nu} - \frac{1}{b} \end{bmatrix}$

Table 5.5: Overview of the expressions of the selection criteria corresponding to the three downdating schemes, according to the error measure in feature space (left column) and data space (right column).

5.7 Experiments with online LS-SVM

5.7.1 Algorithm

For convenience we assume dealing with stationary data so that selection of a subset is most useful, and the time stamp of a data point does not play a role (otherwise the oldest point is omitted or a forgetting factor is introduced).

On the one hand a smaller number of representers implies that we do not use all information available, which can decrease the model quality. On the other hand the removal of basis vectors is an extra form of regularization, compensating for overfitting to the training data set, which can as such increase the model quality. The optimal size of m depends on the data and only an exhaustive (but unfortunately practically unfeasible) search over all possible combinations of subset sizes could determine this.

A practical means of sparsity control therefore is a window around m in which the number of kernels may vary. On one side, we can fix the subset size and -always alternately- perform an up- and downdate. With such a *zero sparsity window* the model quality relies fully on the choice of m. It is very sensitive to a possible underestimation or overestimation of m, but has the advantage that sparsity and computational requirements are fully controlled. It is this path that we will follow in the sequel. We can summarize the scheme essentially as:

- Start with an initial LS-SVM model based on m representers.
- For 1 to $n - m$ do
  - Update LS-SVM by inclusion of a new datapoint (use table 5.3).
  - Select a datapoint (use table 5.5) and swap it with the last positioned representer in the expansion.
  - Downdate LS-SVM by exclusion of the last datapoint (use table 5.4).

On the other side, with a *nonzero sparsity window*, one can allow the subset size to vary more dynamically and let appropiate criteria decide *when* to up- or downdate. Multiple consecutive up- or downdates are possible then. Firstly, this shifts in fact only the problem of determining m to determining those criteria (or any parameters involved in them, like thresholds for instance). Secondly, we might have to choose an upper limit as to bound the computational requirements, and a lower limit to enforce a minimal model size. These limits (forming a window) determine the degree of sparsity control, but require again a fixed choice. This alternative approach is not discussed here, since it is an entire subject on itself that would lead us too far.

5.7.2 Comparison

First, we compared the three schemes of updating by extension, approximation and recursive (see Table 5.3). In Figure 5.2 we show a typical example of the difference in performance, average Mean Square Error (MSE) on an independent test set versus support vector subset size (in % of total dataset). The test was run on the one-dimensional artificial sinc function, consisting of 150 training points and a test set of 80, with added noise of variance 0.1. We used the common Gaussian kernel and the optimal set of hyperparameters (ν, h) was tuned by standard 10-fold cross-validation (CV) model selection on the full training set over a wide grid of values.

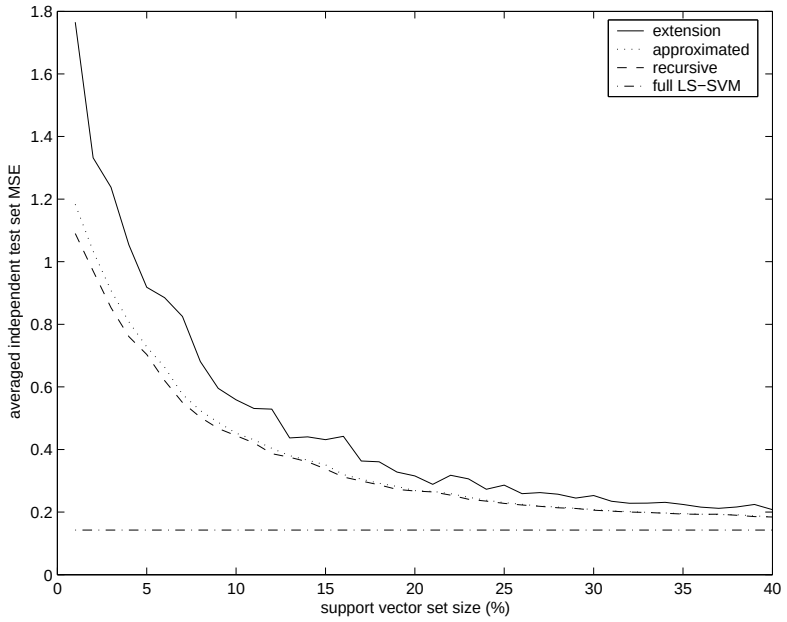


Figure 5.2: Average MSE on independent test set versus support vector subset size (in % of total dataset). Three updating schemes are applied to artificial sinc data. The difference is most notable for extension since it is the most crude form of updating, but on most datasets the difference is not significant, and the variance is typically much larger.

In this test, the updating by extension and approximation are at each step based on the batch LS-SVM solution *with one representer less*, so each update point on the curve is relative to a previous batch solution. A 100 randomisations were performed on the order of the training set datapoints. One can observe, on one hand that the extension has least average performance, which can be expected since it is a crude way of estimation of the coefficient of the new

included representer. On the other hand, using least squares approximation for only this coefficient, gives performance close to the exact updating by recursion. The variances on the curves amount to 25% of the mean value and except for very small support vector set sizes, the differences between the methods are not significantly different. The same picture is apparent in realworld datasets.

Second, we show an example of downdating by truncation under different selection criteria: random (s_0), minimal coefficients (s_1), entropy (s_2), truncation (s_3), approximation (s_4), recursive data space (s_5), recursive feature space (s_6), regularized sum (s_7) and regularized in 1 point (s_8). We demonstrate this by example on the UCI Auto dataset [15]. This data set consists of 392 cases having 7 input variables, where the task is to predict the fuel consumption. We picked at random a training set of size 300 and a test set of size 92 (the 10-fold CV tuned parameters for the full set were $h^2 = 56.57$ and $\nu = 58.44$, with log(MSE)=-2.33).

In Figure 5.3 we show the average MSE on an independent test set versus support vector subset size (in % of total dataset) on the Auto dataset. The downdating is at each step based on the batch LS-SVM solution *with one representer more*. A 100 randomisations were performed on the order of the training datapoints. The upper line (s_0) is the random vector selection which serves as a reference baseline for the performance and is of course the worst case, with large standard deviations. Selection based on the regularized least introduced error made in just one point, with formula (5.16), is second worst. Just like noted in [85], evaluation of the error in the omitted point only leads to poor results, it is important to take into account the global error in all points for the selection. Average behaviour is noted for entropy based selection, which is quite contrary to reported good results [54]. We can conclude that its 'statistical' application (thus subsampling many times from a given set, instead of one time) is crucial in order to select good points. Further, all other selection methods perform undiscernably good.

Third, on the Boston Housing data set, we performed some experiments with online sparse approximation by LS-SVM machines of different sizes of support vector set. This data set [77] consists of 506 cases having 13 input variables. The aim is to predict the housing prices. We picked at random a training set of size 400 and a test set of size 106 (the 10-fold CV tuned parameters for the full set were $h^2 = 15.96$ and $\nu = 32.35$, with log(MSE)=-1.96).

A comparison was made of all possible combinations between updating (from Table 5.3), downdating (from Table 5.4) and selection schemes (from Table 5.5) on several UCI benchmark datasets. Because downdating by recursion excludes the selected point totally, we leave out these three combinations. Poor results were noted for the *extension + truncation* and *extension + approximation* combinations; these turn out to be too crude methods for the online approach. This narrows down the experiments to 4 combinations: (i) extension+approximation, (ii) approximation+truncation, (iii)

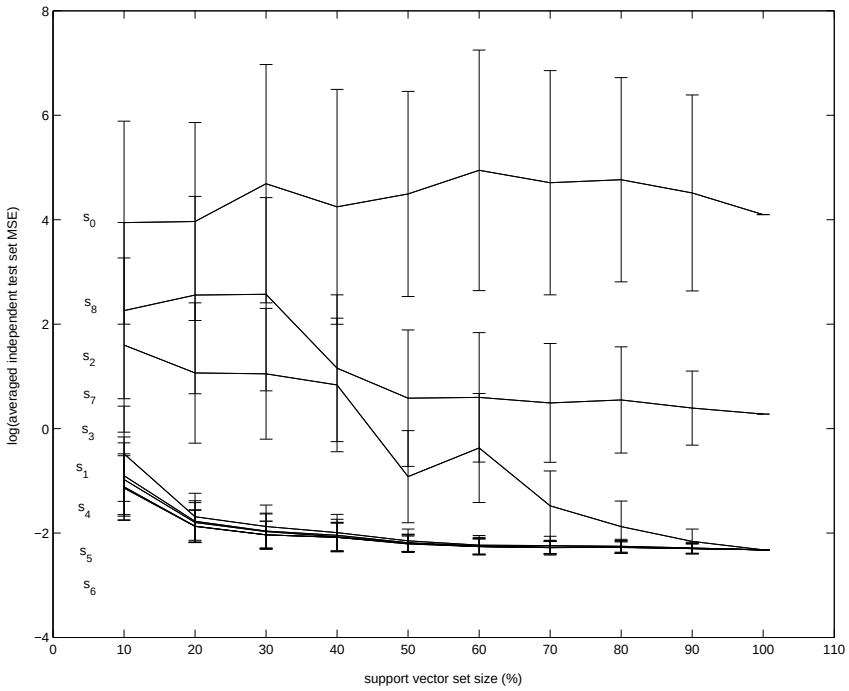


Figure 5.3: Average MSE on independent test set versus support vector subset size (in % of total dataset) for *downdating by truncation* on the AUTO dataset. Random selection (upper curve, s_0) performs worst, followed by selection based on regularized least introduced error in 1 point (s_8); these methods are clearly not good and exhibit large variances. Entropy based selection (s_2) does not perform as good as expected, but it might be reasonable since it is not applied in a statistical sense. Other methods like minimal coefficients (s_1), truncation (s_3), approximation (s_4), recursive data space (s_5), recursive feature space (s_6), regularized sum (s_7) perform practically equally well and show that our discerned set of selection criteria make indeed a significant contribution when downsizing a dataset.

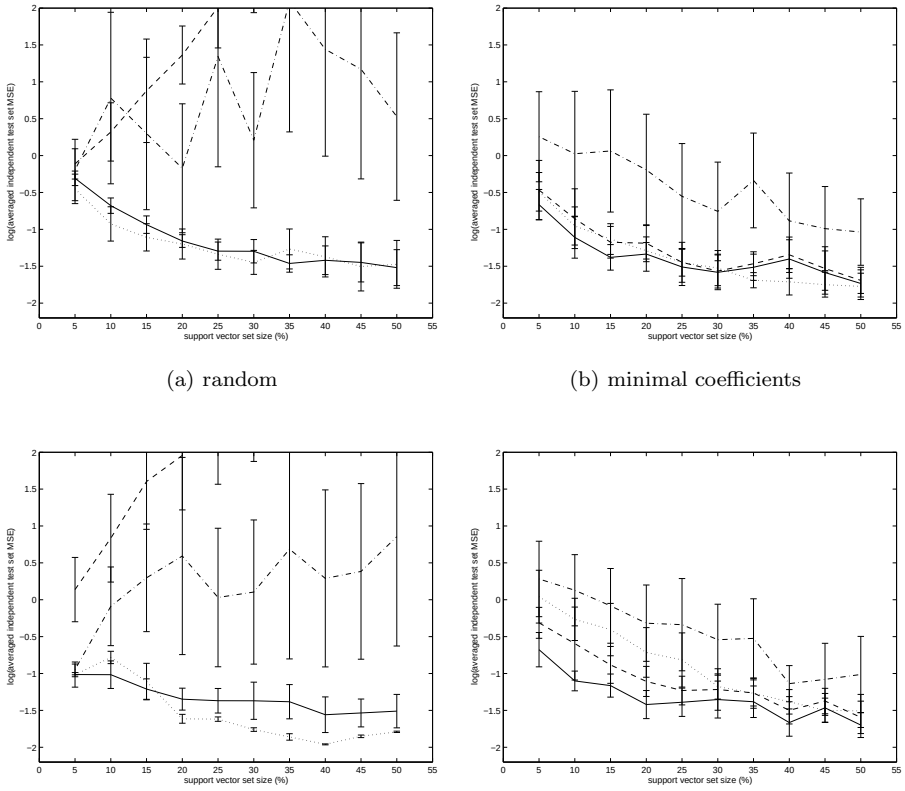
recursive+truncation and (iv) recursive+approximation.

We tested these 4 online approaches with 8 selection criteria. In Figure 5.4 and 5.5 we set out the average MSE on an independent test set versus support vector subset size (in % of total dataset, ranging from 5% to 50%) on the Boston data set. The average is over randomisations with different datapoints from the training set.

Concerning performance, approach (iv) delivers consistently best results (exept in the case of the selection by truncation) and it shows (in the selection by approximation case) that a sparse approximation of about 30% support vectors suffices to equal the performance on the full set. One may readily observe that method (iii) underperforms in comparison to the other methods.

The huge standard variations show that there is a lot of instability caused by truncation or that the recursive updating is sensitive to the error in the downdated solution vector. Approach (ii) is -overall- second best, followed by (i). These performances also agree with the proportion of information content that is preserved at the up/downdating steps.

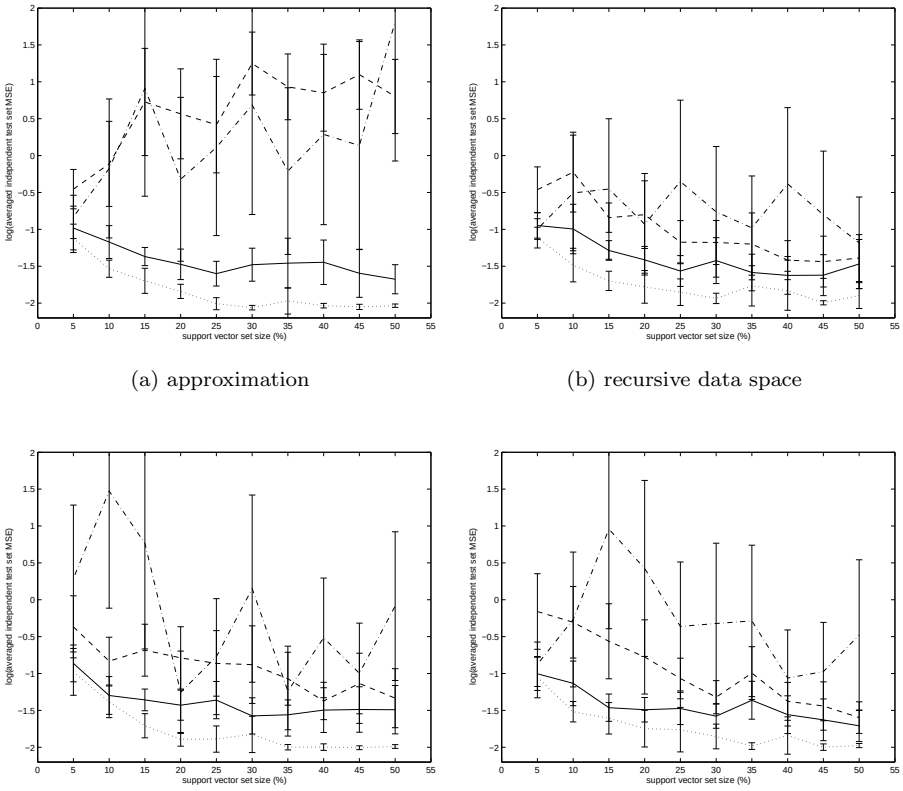
Concerning the criteria, under the cases of selection by approximation, recursive data space and recursive feature space, one reaches best performance and there is a significant influence from the selection criterion in method (iv). Selection by the regularized system of de Kruif -with our adaptation of taking the sum of the errors over his criterion- yields equally good results, whereas selection by minimal coefficients or truncation yield barely improvement over the random selection, which serves here as the reference. The selection by entropy again does not seem to deliver a definite improvement for the methods. The same conclusions could be drawn from these tests on other UCI datasets (Servo, Machine).



(a) random (b) minimal coefficients

(c) entropy (d) truncation

Figure 5.4: Average MSE on independent test set versus support vector subset size (in % of total dataset). Comparison of three tracking methods (dashed: extension+truncation; full: approximation+truncation; dashdotted: recursive+truncation; dotted: recursive+approximation) for different selection criterions on the Boston Housing data set.



(a) approximation

(b) recursive data space

(c) recursive feature space

(d) regularized

Figure 5.5: Average MSE on independent test set versus support vector subset size (in % of total dataset). Comparison of four tracking methods (dashed: extension+truncation; full: approximation+truncation; dashdotted: recursive+truncation; dotted: recursive+approximation) for different selection criterions on the Boston Housing data set.

5.8 Conclusions

We compared different pruning methods for sparse LS-SVMs: four support value based pruning schemes and two (for pruning adapted) subset selection algorithms against random pruning and backward selection on independent test sets in a set of benchmark experiments. From these results we conclude that omitting a point based upon its weighed support value in the case $\gamma \neq \infty$ is rarely yielding satisfactory pruning results. We suggest to use instead as criterion the *sum* of these values at *all* training datapoints, yielding significant improvements.

Pruning were the point is omitted that has the smallest support value, weighed by the corresponding diagonal element of the inverted kernel matrix, achieved overall excellent pruning results, although in theory the formula is not intended for cases were $\gamma \neq \infty$. The subset selection based pruning algorithms perform overall as second best. If we also take into account the computational cost, standard pruning based on the support values remains most cheap and cheerful.

In addition, we also considered the LS-SVM in an online setting and gathered incremental and decremental approaches for updating and downdating the LS-SVM for use in an online setting. We presented these in a structured scheme with alternating updating and downdating, combined with selection criteria. We derived a clear taxonomy of selection criteria that are naturally *induced* by the error measures of the downdating methods, in data space as well as in feature space.

In a series of experiments, we made then a comparison of performance on some benchmark datasets for a collection of up-and downdating combinations with a fixed size of support vector set and equiped with different selection mechanisms. From our experiments we may conclude that the recursive updating combined with approximated downdating is the recommended method, and that for selection the recursive solution in data space is computationally most inexpensive and chooses significantly better support vectors.

Chapter 6

Updating and Tracking the Dominant Kernel Eigenspace

The dominant set of eigenvectors of the symmetric kernel matrix is used in many important kernel methods (like e.g. kernel Principal Component Analysis, feature approximation, denoising, compression, prediction) in the machine learning area. Yet in the case of dynamic and/or large scale data, the batch calculation of the Gram matrix and the computational demands of the eigenvector decomposition limit the applicability.

In this chapter we aim to resolve this limitation. We propose an efficient incremental approach for fast calculation of the dominant kernel eigenbasis, which allows to track the kernel eigenspace dynamically [82]. We developed a numerical algorithm to compute the approximation based on a given best rank estimate and successive minimal SVD and QR decompositions. Experiments show that our updating scheme delivers a numerically stable and accurate approximation for eigenvalues and eigenvectors, matching the batch algorithm results excellently [81].

6.1 Introduction

Since the introduction of Support Vector Machines (SVM) [183] many learning algorithms are being transferred to a kernel representation [154, 165]. The important benefit lies in the fact that nonlinearities can be allowed, while avoiding to solve a nonlinear optimization problem. The transfer is implicitly

accomplished by means of a nonlinear map into a Reproducing Kernel Hilbert Space H_k (RKHS) [190].

When n data points $\mathbf{x}_i \in \mathbb{R}^p$ are given, a square symmetric $n \times n$ kernel matrix K with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ can be computed, where the kernel k provides a similarity measure between pairs of data points, $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j)$.

A whole myriad of methods (e.g. Kernel Principal Components Analysis (KPCA) [155], Fixed Size Least Squares Support Vector Machines (FS-LSSVM) [165], de-noising [153, 120, 97, 147], data modelling [174], etc.) rely, either directly or indirectly, on the dominant m eigenvectors of K, where $m \ll n$. The usual tool for computing this eigenspace is the Singular Value Decomposition (SVD) because of its high numerical accuracy [73]. The drawback is that standard routines for the batch computation of an SVD require $O(n^3)$ operations (or $O(mn^2)$, for instance in deflation schemes [73]).

It is clear that for large datasets (as for example in image processing, computer vision or object recognition), the kernel method with its powerful advantage of dealing with nonlinearities, is computationally severely limited. For large datasets an eigendecomposition of K can simply become too time-consuming. But also memory is an issue as $O(n^2)$ memory units are required. In addition, computations may have to be repeated several times if the problem is nonstationary. Some applications require adaptive algorithms. A truly online algorithm for extracting kernel principal components has not been reported so far, although the problem is important and attempts to tackle it are presently undertaken [146, 101, 104].

The literature that spreads out over the last three decades has spent considerable effort on dealing with these issues, with a common denominator called SVD updating [20, 19, 127, 25, 108, 7], the process of using the previous eigenspace when a new data instance arrives. Most of them are methods in which the eigenspace is incrementally updated when more rows (or more columns) are added to the involved matrix.

Yet, in the particular context of kernel algorithms, where the matrix dimensions correspond to the number of training data points, those updating schemes cannot be applied in a straightforward manner. The kernel matrix namely expands *both* in row and column dimension when a point is added. To our present knowledge, this case has not been addressed yet. In this chapter we present a fast online updating and downdating procedure for the dominant eigenspace in case of a *growing* symmetric square matrix, with a computational complexity of $O(nm^2)$ and memory requirements of $O(nm)$ per iteration, and with very limited loss of accuracy. For the situation of subspace tracking, where one keeps the dimension of the subspace fixed, also an efficient downsizing mechanism is proposed.

6.2 Updating and downdating

In this section we describe the updating in detail for one step of the iterative procedure and show that it compares favorably to the complexity of a batch SVD. Due to the particular structure of our updating scheme, downdating will just be a special case of updating.

6.2.1 Updating

For the update we assume that we have the (approximate) dominant $n \times m$ eigenspace U_{nm} of the square symmetric $n \times n$ kernel matrix K_n, based on n data points. We suppose that also an estimate for the number of dominant eigenvectors/values m is available. Obtaining m can be a subject entirely on itself, and is considered outside the scope of this thesis.

The goal of updating is to obtain the $(n+1) \times m$ eigenspace $U_{(n+1)m}$ of the $(n+1) \times (n+1)$ kernel matrix K_{n+1}, based on the previous n data points and one extra data point. In the kernel matrix an update is located in the last row and column, which expands K_n *both* in row and column dimension when a point is added:

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \tag{6.1}$$

where $\mathbf{a}$ is a $n \times 1$ vector of kernel entries $a_i = k(\mathbf{x}_i, \mathbf{x}_{n+1})$ and scalar $b = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})$. In the following we derive an efficient formula for the dominant eigenspace of K_{n+1} based on the dominant eigenspace of K_n.

The key observation is that K_{n+1} can be expanded as a stripped matrix to which a difference of 2 rank-one matrices is added. We propose the following possible, convenient expansion:

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{b} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \begin{bmatrix} \mathbf{a}^T & b \end{bmatrix} - \frac{1}{b} \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{a}^T & 0 \end{bmatrix} \tag{6.2}$$

where $\mathbf{0}$ is a $n \times 1$ vector of zeros.

Let us denote the given rank-m SVD of the submatrix $K_n \approx U_{nm} \Sigma_m U_{nm}^T$. We then simply substitute this in (6.2) and rewrite the expression with block matrices:

$$K_{n+1} \approx \begin{bmatrix} U_{nm} \\ \mathbf{0}^T \end{bmatrix} \Sigma_m \begin{bmatrix} U_{nm}^T & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \frac{1}{b} \begin{bmatrix} \mathbf{a}^T & b \end{bmatrix} - \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} \frac{1}{b} \begin{bmatrix} \mathbf{a}^T & 0 \end{bmatrix}. \tag{6.3}$$

We can then organize the different vectors all together in a convenient symmetric factorization:

$$K_{n+1} \approx \begin{bmatrix} U_0 & A \end{bmatrix} \begin{bmatrix} \Sigma_m & O \\ O^T & D_u \end{bmatrix} \begin{bmatrix} U_0^T \\ A^T \end{bmatrix} \tag{6.4}$$

in which O represents the $n \times 2$ matrix of zeros, and in which we denote

$$U_0 := \begin{bmatrix} U_{nm} \\ \mathbf{0}^T \end{bmatrix}. \tag{6.5}$$

The $(n+1) \times 2$ matrix A contains the extra columns,

$$A := \begin{bmatrix} \mathbf{a} & \mathbf{a} \\ b & 0 \end{bmatrix}, \tag{6.6}$$

and the diagonal 2×2 matrix D contains the coefficients of the rank-1 updates,

$$D_u := \begin{bmatrix} 1/b & 0 \\ 0 & -1/b \end{bmatrix}. \tag{6.7}$$

With the above rearrangement we have obtained now an eigenspace, with a set of orthonormal eigenvectors in U_0, extended with the two columns of A, giving the matrix $\begin{bmatrix} U_0 & A \end{bmatrix}$. The two column vectors of A disturb the orthogonality of this matrix. To restore this orthogonality, we need to compute the part of A that is orthogonal to the eigenspace. So we must decompose the vectors of A into a component $A_{U_0^\perp}$ orthogonal and a component A_{U_0} parallel to the column space of U_0:

$$A = A_{U_0^\perp} + A_{U_0} = (I_n - U_0 U_0^T)A + U_0 U_0^T A. \tag{6.8}$$

This splits up the extending contribution of A into an orthogonal component that increases the rank of the eigenspace, while the parallel component will cause the eigenvectors to be rotated, see Figure 6.1.

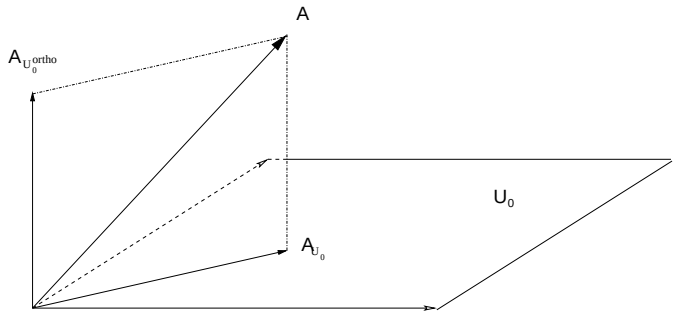


Figure 6.1: The previous eigenspace U_0 is being extended with information from A, which can be decomposed into a parallel and orthogonal component relative to U_0.

This leads to a factorization

$$\begin{bmatrix} U_0 & A \end{bmatrix} = \begin{bmatrix} U_0 & (I_n - U_0 U_0^T)A \end{bmatrix} \begin{bmatrix} I_m & U_0^T A \\ O^T & I_2 \end{bmatrix}. \tag{6.9}$$

To make the column vectors of $A_{U_0^\perp}$ mutually orthogonal, we compute the QR-decomposition

$$Q_A R_A \xleftarrow{\text{QR}} A_{U_0^\perp}, \tag{6.10}$$

in which $A_{U_0^\perp}$ is only of dimension $(n+1)\times 2$. This yields the QR decomposition

$$\left[\begin{array}{cc} U_0 & A \end{array}\right] = \left[\begin{array}{cc} U_0 & Q_A \end{array}\right] \left[\begin{array}{cc} I_m & U_0^T A \\ O^T & R_A \end{array}\right] =: Q_u R_u. \tag{6.11}$$

So far, only $O(nm)$ operations are needed for the decomposition of A into $A_{U_0^\perp}$ (if we perform multiplications in the most efficient order) and its QR-decomposition [73]. In comparison with computing the QR factorization (6.11) from scratch, the number of computations has been limited by exploiting the mutual orthonormality of the columns of U_0. Substitution into (6.4) yields

$$K_{n+1} \approx \left[\begin{array}{cc} U_0 & Q_A \end{array}\right] \left[\begin{array}{cc} I_m & U_0^T A \\ O^T & R_A \end{array}\right] \left[\begin{array}{cc} \Sigma_m & O \\ O^T & D_u \end{array}\right] \left[\begin{array}{cc} I_m & O \\ A^T U_0 & R_A \end{array}\right] \left[\begin{array}{c} U_0^T \\ Q_A^T \end{array}\right]. \tag{6.12}$$

In order to obtain the optimal eigenvectors, a small SVD on the three middle matrices is necessary. The smallest singular value and corresponding eigenvector will then be discarded, which costs in total only a negligible $O(m^3)$ operations:

$$K_{n+1} \approx Q_u \left(R_u \left[\begin{array}{cc} \Sigma_m & O \\ O^T & D_u \end{array}\right] R_u^T \right) Q_u^T \overset{SVD}{=} Q_u V' \Sigma'_m V'^T Q_u^T. \tag{6.13}$$

It finally takes then $O(nm^2)$ operations to obtain an updated eigenspace

$$U' = Q_u V', \tag{6.14}$$

which is spent on the rotation of the orthonormal eigenspace Q_u according to V'. When the updating is followed by a downdating and downsizing, this last expensive operation may be postponed untill the end of the iteration step.

6.2.2 Downdating

For downdating, i.e. excluding the influence of a point, usually a forgetting factor is introduced for the previous eigenspace. This has on the one hand the advantage to smoothen data signal variations, but on the other hand it is only suitable for slowly varying processes. It is often more desirable to have a faster tracking response that can catch sudden signal changes, which requires that the net contribution of a point can be excluded [12].

The above key element in (6.2) of the updating scheme was that the expanded matrix can be written in terms of a stripped matrix to which a difference of 2 rank-one matrices is added. It supplies in fact immediately the right tool

for performing a downdate, because in this case we only have other rank-one matrices:

$$\begin{bmatrix} 0 & 0 \\ 0^T & K_n \end{bmatrix} = \begin{bmatrix} d & \mathbf{c}^T \\ \mathbf{c} & K_n \end{bmatrix} - \frac{1}{d} \begin{bmatrix} d \\ \mathbf{c} \end{bmatrix} \begin{bmatrix} d & \mathbf{c}^T \end{bmatrix} + \frac{1}{d} \begin{bmatrix} 0 \\ \mathbf{c} \end{bmatrix} \begin{bmatrix} 0 & \mathbf{c}^T \end{bmatrix}. \quad (6.15)$$

We do not repeat the procedure for obtaining the eigenspace because it is completely analogous with the updating, only the matrices are slightly different.

However there is one difference concerning dimensions here. Whereas the row dimension of the eigenspace for an update grows from n to $n+1$, the reverse is not the case for a downdate. If the kernel matrix is exactly rank-n, then the first row of U' will be zero. However, in practice this will never be exactly the case. In the following section we propose a downsizing scheme to reduce the dimension of the kernel matrix.

6.3 Downsizing

As mentioned in the previous section, for each update the row dimension of the eigenspace matrix increases. This is not practical for on-line applications where one wants a subspace tracker, where the dimension of the eigenspace is to remain constant. This means we must downsize the matrix after each update, thereby preserving orthogonality. In this section we propose an efficient scheme to accomplish that and recover the orthogonal eigenspace, up to a unitary transformation.

In downsizing, the first row of U', obtained in (6.14) and denoted by the column vector v, must be deleted, such that we can continue with the resulting part V:

$$U' = \begin{bmatrix} v^T \\ V \end{bmatrix}. \quad (6.16)$$

In practice, the columns of V are not exactly mutually orthogonal. Orthogonality may be restored by means of a QR-decomposition of V, but this is expensive. In this section we will present a cheap approximate solution.

Consider

$$U'^T U' = I_m = \begin{bmatrix} v & V^T \end{bmatrix} \begin{bmatrix} v^T \\ V \end{bmatrix} = V^T V + \|v\|^2 \bar{v}\bar{v}^T \quad (6.17)$$

where $\bar{v} = v/\|v\|$.

Basically, we need a transformation matrix M such that V will be orthogonalized:

$$VM = Q \qquad \text{with} \qquad Q^T Q = I_m. \quad (6.18)$$

In this orthogonality condition we substitute (6.17) which yields

$$I_m = M^T V^T V M = M^T (I_m - \|v\|^2 \bar{v}\bar{v}^T) M. \tag{6.19}$$

Exploiting the structure, a convenient choice for M is

$$M = \begin{bmatrix} \bar{v} & \bar{v}_\perp \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{1-\|v\|^2}} & \mathbf{0}^T \\ \mathbf{0} & I_{m-1} \end{bmatrix} =: CJ, \tag{6.20}$$

where $\bar{v}_\perp$ is the $m \times (m-1)$ orthogonal complement of $\bar{v}^T$ such that $C^T C = I_m = CC^T$ and $\mathbf{0}$ is a $(m-1) \times 1$ vector of zeros. The computation of the matrix $\bar{v}_\perp$ costs a negligible number of $O(m^3)$ operations since it is the right nullspace of $\bar{v}^T$ (in Matlab implemented as null($\bar{v}^T$)).

To conclude, the $(n+1) \times m$ eigenspace U' is replaced by the $n \times m$ eigenspace Q:

$$U' = \begin{bmatrix} v^T \\ V \end{bmatrix} \xrightarrow{\text{downsize}} Q := VM. \tag{6.21}$$

The total downsizing cost is dominated by this transformation itself, where M is multiplied with V from the left, which costs $O(nm^2)$ operations.

Of course the matrix with singular values is adapted accordingly to absorb the effect of the transformation matrix:

$$\Sigma'_m \rightarrow M^{-1} \Sigma'_m M^{-T}, \tag{6.22}$$

which has only a negligible cost of $O(m^3)$ operations. If the downsizing is preceded by an update and downdate, the SVD of those steps may both be postponed, to perform it only after downsizing. In this way only one SVD per iteration step is required for tracking.

The schemes of updating and downdating in combination with this downsizing method form a fast dominant eigenspace tracker algorithm that needs per step only $O(nm^2)$ operations and $O(nm)$ memory space. The overall algorithm is summarized in Table 6.1.

$K_n^{(k)}$ is a kernel matrix associated with data points $\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+n-1}$.
Given $U_{nm}^{(k)}\ \Sigma_m^{(k)}\ (U_{nm}^{(k)})^T \xleftarrow[m\text{-rank}]{\text{SVD}} K_n$.
$K_n^{(k+1)}$ is a kernel matrix associated with data points $\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_{k+n}$.
Calculate $K_n^{(k+1)} \approx U_{nm}^{(k+1)}\ \Sigma_m^{(k+1)}\ (U_{nm}^{(k+1)})^T$:

- Update:
  - $U_0 = \begin{bmatrix} U_{nm}^{(k)} \\ \mathbf{0}^T \end{bmatrix}$
  - $Q_A R_A \xleftarrow{\text{QR}} (I - U_0 U_0^T) \begin{bmatrix} \mathbf{a} & \mathbf{a} \\ b & 0 \end{bmatrix}$
  - $Q_u = [U_0\ Q_A]$
  - $\Sigma_u = R_u \begin{bmatrix} \Sigma_m^{(k)} & O \\ O^T & D_u \end{bmatrix} R_u^T$
- Downdate:
  - $Q_B R_B \xleftarrow{\text{QR}} (I - Q_u Q_u^T) \begin{bmatrix} d & 0 \\ \mathbf{c} & \mathbf{c} \end{bmatrix}$
  - $Q_d = [Q_u\ Q_B]$
  - $\Sigma_d = R_d \begin{bmatrix} \Sigma_u & O \\ O^T & D_d \end{bmatrix} R_d^T$
- Downsize: $Q_d \rightarrow$ orthogonal eigenspace T and transformation M
- Perform SVD:
  - $W\ \Sigma_m^{(k+1)}\ W^T \xleftarrow{\text{SVD}} M^{-1}\Sigma_d M^{-T}$
  - discard last 4 singular values and vectors
- Perform rotation: $U_{nm}^{(k+1)} = TW$.

Table 6.1: Schematic high-level overview of the proposed algorithm for tracking the dominant kernel kernel eigenspace.

6.4 Experiments

We implemented the proposed up/downdating and downsizing schemes in Matlab. A straightforward comparison of our (non-optimized) code with the built-in non-recursive Matlab SVDS routine shows that a computational gain is indeed achieved, especially for large dimensions (1000 and more) and few eigenvectors (order 10-100). Yet, in the experiments we primarily aim to characterize how well the true eigenvectors are approximated whilst tracking (i.e. up/downdating and downsizing) (static) datasets.

We considered three representative data sets that are typically assessed in the kernel based machine learning domain, on which kernel methods were applied which required the dominant kernel eigenspace. We employed the radial basis function (RBF) kernel, with h the kernel width parameter:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{h^2}\right). \tag{6.23}$$

In this experimental section we will illustrate that, in addition to (i) reliable *tracking*, our algorithm can also (ii) yield the dominant eigenspace of a *very large* kernel matrix (say of order 10,000 $\times$ 10,000 or more), by consecutively applying updates on a given rank-m SVD of a kernel matrix.

6.4.1 Eigenspace Tracking

The Abalone benchmark data set [15] has $n = 3000$ training instances, with dimension $p = 7$. The RBF kernel width was fixed at 18.57. The order of the data points is taken at random. In Figure 6.2 the eigenspectrum of the symmetric kernel matrix K is shown, which has a quite steep exponentially decreasing behaviour, typical in most kernel based applications. The first 10 eigenvalues have a distinct interspacing and they cover 90% of the total mass of the energy. Based on this visual inspection we expect good tracking results for the eigenspace of dimension 500×8. We take some margin and track the m=20 first eigenvectors to compare it with the batch SVD.

In order to assess the accuracy and stability of the tracking algorithm, we employed the following measures:

- The relative error between the eigenvalue computed by SVD tracking $\tilde{\lambda}_i$ and the one by batch SVD λ_i is given by

$$E_\lambda^{rel} = \frac{\lambda - \tilde{\lambda}}{\lambda}. \tag{6.24}$$

  In Figure 6.3 we depict the average of this measure over 2500 iteration steps, per eigenvalue and shown with a standard one-deviation band.

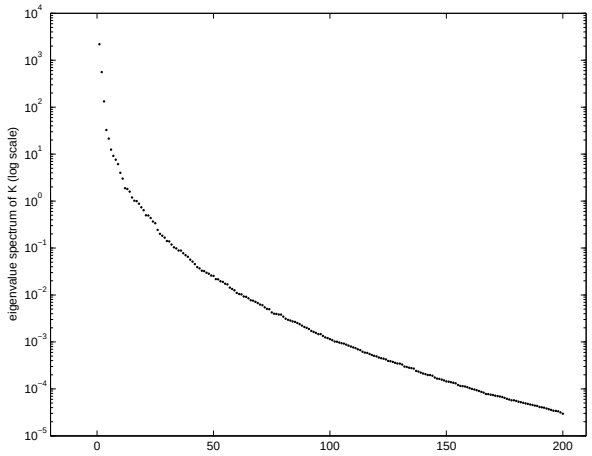


Figure 6.2: Eigenspectrum of the symmetric kernel matrix K for the Abalone benchmark data set, shown for the first 300 (of 3000) values.

One may observe that the first 10 dominant eigenvalues are very well approximated, with relative errors ranging from 0.01% up to 1%. The average accuracy decreases exponentially towards the last computed component, while the standard deviation increases accordingly. Only eigenvalues separated with a relatively large gap are being tracked well, which is inherent in an approximation scheme [73]. Simple tests verified the fact that enlarging the column dimension of the eigenspace, decreases the relative error. Enlarging the row dimension has the same influence, but to a less extent.

In Figure 6.4 we show the relative error of the eigenvalues versus iteration step, for some eigenvalues (as indicated by the number) of the kernel matrix based on the Abalone data set. The dominant eigenvalues remain stable and are consistently approximated during the iterations from 0.001 up to 1 percent error.

- A good low-rank approximation delivers a faithful representation of the original matrix. Therefore, an overall measure of accuracy often used is the relative error based on a matrix norm:

$$E_{\mu}^{rel} = \frac{\|K - \tilde{K}\|_F}{\|K\|_F}, \tag{6.25}$$

where $\tilde{K}$ is a rank-m approximation of K and $\|\cdot\|_F$ denotes the Frobenius norm. In this case we have a low-rank approximation $\tilde{K} = \tilde{U}\tilde{S}\tilde{U}^T$ where $\tilde{U}$ is the eigenspace of K and $\tilde{S}$ is the matrix with singular values on the diagonal, both resulting from our algorithm. In our experiments we compare the relative error of the rank-m approximation obtained

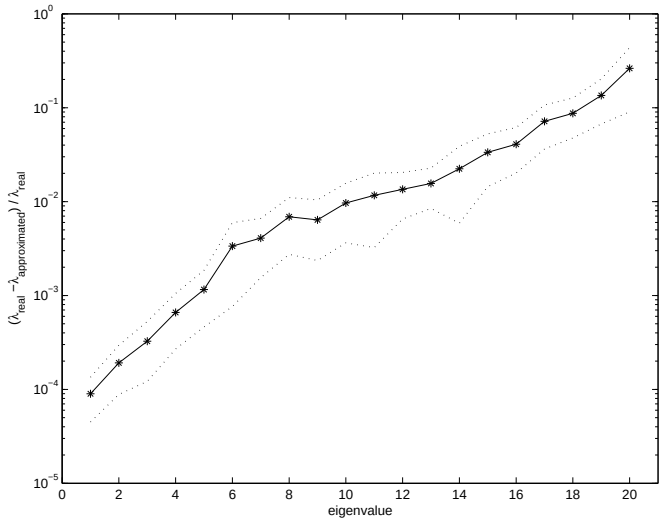


Figure 6.3: Relative error between the eigenvalues computed by SVD tracking and by batch SVD, averaged over 2500 iteration steps, with a standard deviation band equal to one (dotted), for the tracked 500×20 kernel matrix of the Abalone data set. The estimated dominant (first 10) eigenvalues remain accurate.

by our algorithm and the best m-rank approximation obtained by the batch SVD. Figure 6.5 shows for the Abalone data set example that the tracked eigenspace approximates the batch eigenspace very closely and consistently.

- The subspace angles [73] between the dominant eigenspaces of $\tilde{K}$ and K are direct indicators of how close these eigenspaces are. Figure 6.6 shows subspace angles between eigen*vectors* versus the eigenvector number (circles), for the kernel matrix, based on the Abalone data set. Also shown is the subspace angle between the eigen*spaces* versus the dimension of the eigenspaces (squares). The subspace angle can range from zero (totally linearly dependent and coincident) to $\pi/2$ (orthogonal). The subspace angle value of 10^{-2} corresponds to deviations of half a degree, so the dominant (first 10) eigenvectors are quite well approximated for the largest part, taking into account the upper band of a standard deviation equal to one (dotted). The approximation is found to be closely connected to the gap between the successive eigenvalues. From the moment the eigenspectrum becomes flatter (between eigenvalue 5 and 10), the corresponding errors increase.

- Another important criterion concerns the orthogonality of the eigenspace during successive iterations. This can be monitored by the scalar

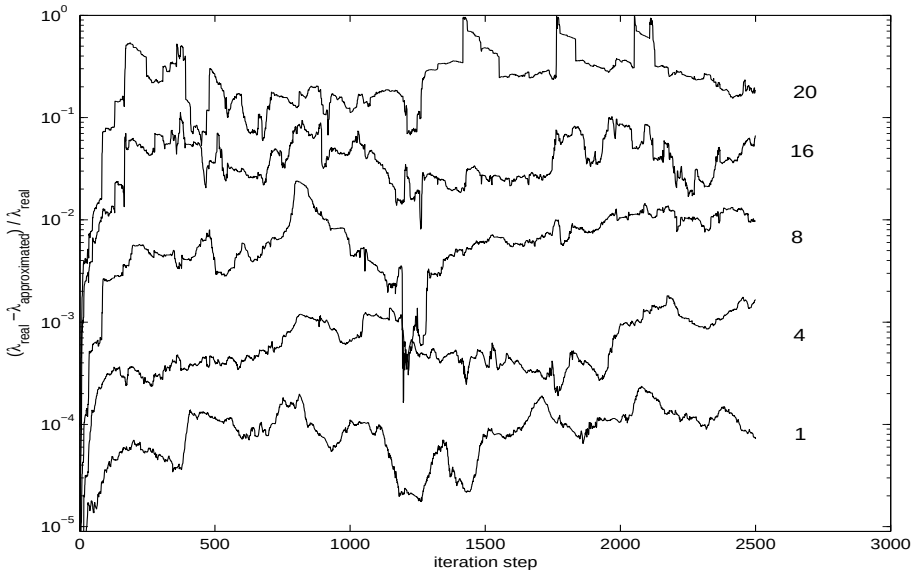


Figure 6.4: Relative error of the eigenvalues versus iteration step, obtained by SVD tracking, for some eigenvalues (as indicated by the number), for the tracked 500×20 kernel matrix of the Abalone data set. The estimated dominant eigenvalues remain accurate during the iteration process.

orthogonality measure

$$E_{\perp} = \|I - \tilde{U}^T \tilde{U}\|_2. \tag{6.26}$$

A zero value indicates perfect orthogonality. In Figure 6.7 we show the typical evolution of this measure per iteration step, for the Abalone data set. In all experiments the orthogonality is very well-preserved and shows values in the range $\left[10^{-15}, 10^{-13}\right]$. When the row dimension is relatively small, reorthogonalization (by e.g. QR) may be necessary at certain intervals. In practice one can compare the orthogonality between the first and last eigenvector, for instance by taking the inner product and comparing against some preset tolerance. This renders the algorithm numerically more robust.

A second large data set considered for tracking is the Adult UCI data set [15], which consists of $45,222$ cases having 14 input variables. The aim is to predict whether the income of a person is greater than 50K using several census parameters, such as age, education, marital status, etc. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 33000$. We used a Gaussian kernel with $h^2 = 29.97$ derived from preliminary experiments.

The spectrum for the whole set cannot be calculated with present standard PC

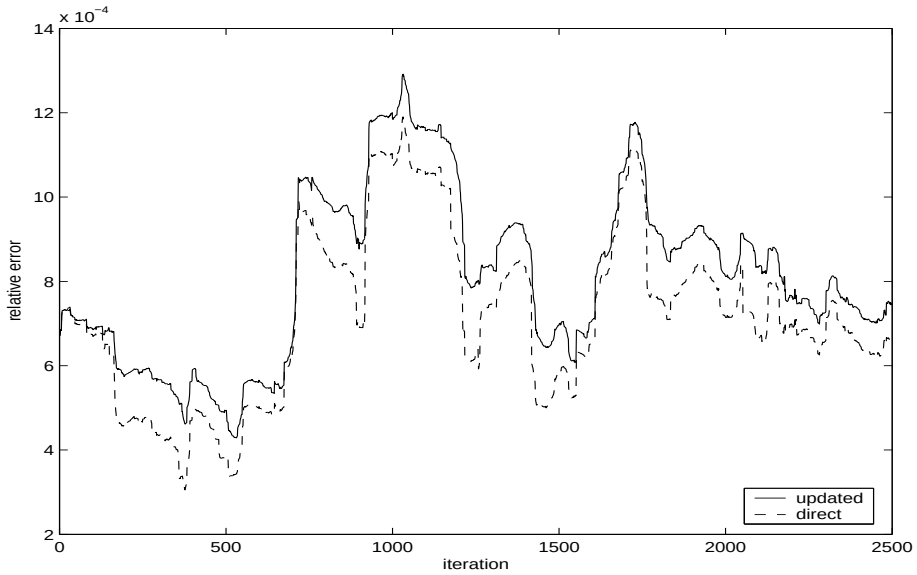


Figure 6.5: Relative error versus iteration step, shown for the tracked m-dimensional eigenspace and the m-rank eigenspace obtained by a batch SVD, for the tracked 500×20 kernel matrix of the Abalone data set. The two plotted lines follow each other closely, indicating good approximation of the eigenspace.

equipment, but estimates for different subsets indicate that 50 eigenvalues will suffice for capturing most of the energy of the spectrum. We tracked therefore a 500×50 dimensional eigenspace. The tracked spectra show on average a flat behaviour for the last 30 eigenvalues. The first 14 eigenvalues are most distinct and are covering 90% of the eigenvalue spectrum, having an average relative eigenvalue error level of 0.05%.

The relative error, defined in (6.25), for the rank-14 approximation obtained by our algorithm and the batch computation over 32500 iterations differs by $(1.2 \pm 0.2)\%$. Tracking a 1000×50 subspace improves this down to a $(0.8 \pm 0.3)\%$ error difference, which indicates reliable close tracking. Also orthogonality remains well-preserved in the range $\left[10^{-14}, 10^{-12}\right]$, and in total 6 additional re-orthogonalizations (by a QR routine) were performed. Also on this data set we obtain a very good tracking accuracy for the dominant eigenspace and eigenvalues. We conclude that, given the fact that the spectrum is continuous, the tracking results of our algorithm are accurate.

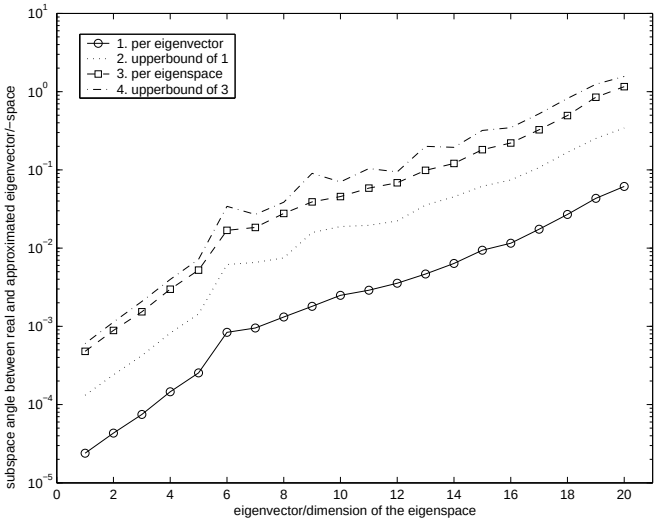


Figure 6.6: Subspace angle between the real and tracked eigen*vector*, shown per eigenvector (circles), with a standard deviation upper bound equal to one (dotted), for the tracked 500×20 kernel matrix of the Abalone data set. Subspace angle of the real and tracked eigen*spaces* (squares) versus the dimension of the eigenspace, with a standard deviation upper band equal to one (dash-dot).

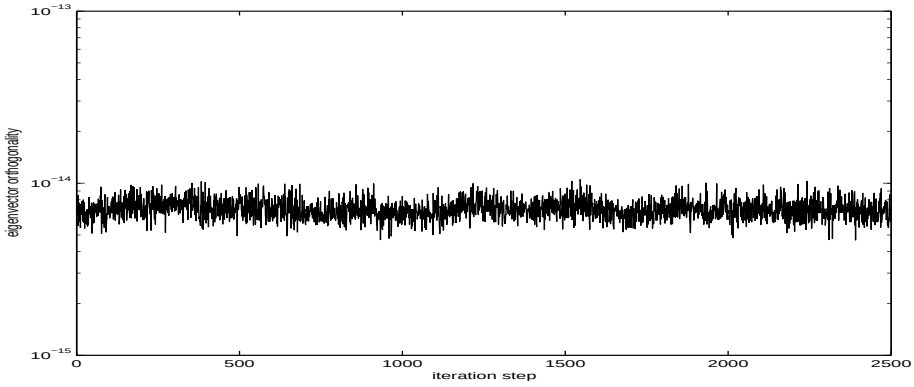


Figure 6.7: Orthogonality of the eigenvectors versus iteration step, for the tracked 500×20 kernel matrix of the Abalone data set. Smaller values indicate better preservation of mutual orthogonality between the eigenvectors.

6.4.2 Eigenspace Updating

In case when the dominant eigenspace of a very large matrix is needed (not fitting into memory or taking too many hours to compute), our updating procedure offers a considerably faster solution. The data is consecutively added to some initial m-rank SVD decomposition and one obtains the $n \times m$-sized dominant eigenspace and eigenvalues with a minimum of calculations. Again high accuracy rates are obtained, as experiments demonstrate.

In Figure 6.8 we plotted the relative errors for each of the 20 first eigenvalues, for different initial eigenbasis dimensions, with n fixed at 1000 and m varied over 20, 40, 60, 100. One may observe very low error values never exceeding one percent. In comparison these values are better than the tracking results because in the mere updating, no downsizing is involved (which would incur some additional loss of numerical precision).

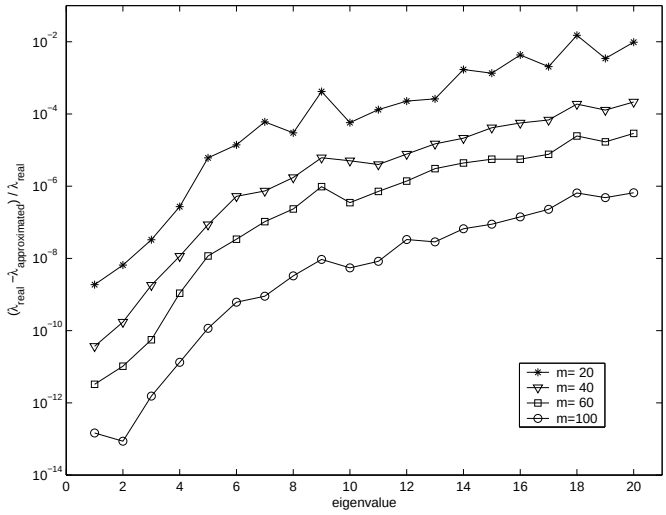


Figure 6.8: Relative errors for each of the 20 first eigenvalues, for different initial eigenbasis dimensions m=20,40,60,100 (while n was fixed at 1000), for the full kernel matrix of the Abalone data set. The updating procedure delivers eigenvalues with high accuracy.

In Figure 6.9 we assess the subspace angles of the eigenvectors in comparison with those obtained by batch SVD. Again one may observe that the eigenvectors are very well approximated: the deviations from the dominant real ones are significantly smaller than one degree.

Table 6.2 shows relative errors, defined in (6.25), and their ratios for different initial eigenbasis dimensions and approximation ranks. From the ratio for the rank-m approximation we see that the difference can vary up to a few percent,

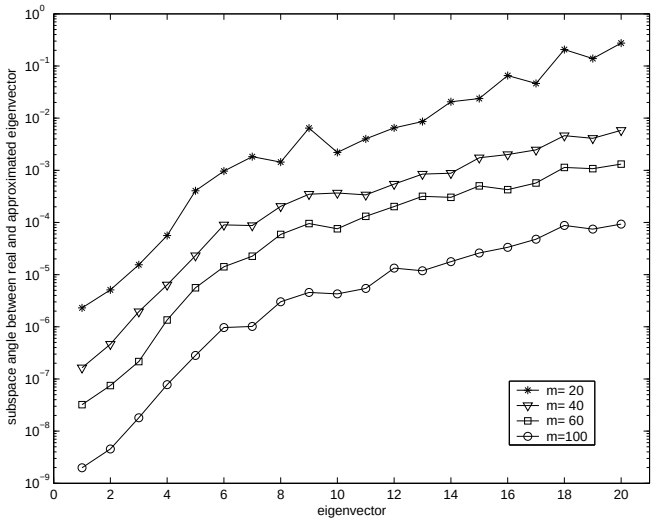


Figure 6.9: Subspace angles of the 20 first eigenvectors, for different initial eigenbasis dimensions m=20,40,60,100 (while n was fixed at 1000), for the full kernel matrix of the Abalone data set. The updating procedure delivers eigenvectors with high accuracy.

while the dominant rank-8 approximation are upper bounded by $10^{-5}\%$ ratios. Orthogonality was on average preserved up to values of 10^{-14}. We conclude that all measures demonstrate a good to excellent level of numerical precision for the obtained eigenspace and eigenvalues, which is related to the gap between the eigenvalues.

		relative error				
		rank-m approximation			rank-8 approximation	
initial n	m	updating	batch	ratio	updating	ratio
500	20	0.0133082	0.0132063	1.0077	0.050801115068	1.000002791643
500	40	0.0050548	0.0047341	1.0677	0.050800975593	1.000000033648
1000	20	0.0132701	0.0132063	1.0048	0.050801161542	1.000003693981
1000	40	0.0050505	0.0047341	1.0668	0.050800975200	1.000000025900
1000	60	0.0025403	0.0024522	1.0359	0.050800973968	1.000000001648
1000	100	0.0008069	0.0007887	1.0230	0.050800973884	1.000000000004

Table 6.2: Relative errors for the rank-m and rank-8 approximations of the full kernel matrix, obtained by the SVD updating algorithm and the batch SVD, for the Abalone data set. The close to one ratios indicate good preservation of accuracy. The result of the batch rank-8 approximation is 0.050800973884.

A third, large data set was used to demonstrate the updating results on the commonly known and widely applied USPS digits data [74]. This set consists of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. There are 7291 instances of dimension 256.

Given that the RAM memory of the available computer is 1.5 Gigabytes, the SVD routine can be just fit into memory without any memory problems from the Matlab compiler (contrary to the SVDS routine). The full batch SVD computation takes about 20 hours on a 1GHz PC, while a computation of the 100-dimensional subspace and the corresponding eigenvalues takes 30 minutes with our algorithm. We note that the algorithm was a straightforward non-optimized Matlab script. More efficient coding (for instance in C) can reduce the computation time to a couple of minutes.

We used the RBF kernel with $h = 16$ [152]. The eigenspectrum may be quite steep, but no distinct gap is identifiable. The first 100 eigenvalues were taken on a visual inspection (covering 65% of the total mass of the eigenvalue spectrum, one would need an aditional 1550 eigenvalues to reach 90%).

In Table 6.3 we report on the relative error of the rank-100 approximation (in Frobenius norm, with $\|K\|_F = 1.9097e+03$) and the mean of the relative error of the corresponding eigenvalues. The very close to one ratios, with deviations around one percent or less, indicate that we have again very good coincidence with the batch eigensubspace. Also the eigenvalues remain well-approximated within 2 percent, due to a slight bias of underestimation. We may conclude that even starting with a 100×100 SVD leads to good results for the dominant eigenspace and eigenvalue estimation of a 7291×7291 square kernel matrix. Orthogonality remained well-preserved in the range $\left[10^{-15}, 10^{-14}\right]$, and no re-orthogonalizations were required.

We conclude that the measures for the accuracy of the approximated dominant eigenspace and eigenvalues of a large kernel matrix indicate very good and stable agreement with those computed by batch solutions.

initial n	rel. error of rank-100 approximation		avg. rel. error
	updating	ratio	of eigenvalues [%]
100	0.03273108890634	1.01538	2.22
500	0.03270590442933	1.01469	2.09
1000	0.03269606721720	1.01431	1.80
1500	0.03261049908786	1.01160	1.44
2000	0.03253083860862	1.00917	1.08
2500	0.03245233729252	1.00673	0.79
3000	0.03241386375776	1.00554	0.64

Table 6.3: Relative errors for the rank-100 approximations of the full kernel matrix, obtained by the SVD updating algorithm and the batch SVD, for the USPS data set with $h = 16$. The close to one ratios indicate good preservation of accuracy. The result of the batch rank-100 approximation is 0.0322351367302. The average relative error of the eigenvalues amounts to 2 percent or less. Both eigenvectors and eigenvalues are thus well-approximated by the proposed algorithm.

6.5 Conclusions

In this chapter we introduced a novel up- and downdating algorithm for the dominant eigenspace of a square large scale symmetric matrix, in which the adaptation simultaneously occurs *both* in the row and columns. Additionally, a downsizing mechanism was proposed such that the algorithm is also capable of tracking the dominant eigenspace (while the dimension remains constant).

The dominant eigenspace of such a matrix is relevant for several kernel based methods in machine learning [165] for tasks in classification, pattern recognition, function estimation, system identification and signal processing.

Although these kernel methods are powerful models for dealing with nonlinearities, they are severely limited when it comes to large scale applications. In that case the computation times often become unacceptable or the kernel matrix cannot even be completely stored in memory. Furthermore there is no true on-line kernel principal component analysis algorithm known that can handle time-varying data. Our algorithm fills this gap, in an efficient manner with a time complexity of $O(nm^2)$ and memory requirements of $O(nm)$ per iteration.

In the experiments we focused on the accuracy and stability of the algorithm and found very good to excellent approximation of the eigenvalues and eigenvectors, with preservation of orthogonality and small relative error for the resulting low-rank approximations.

Chapter 7

Generalized Higher-Order Rayleigh Quotient Iteration

In this chapter we enter the field of multilinear algebra, which involves higher-order tensors. In contrast to matrices, tensors can be viewed as quantities of which the elements are addressed by more than two indices. In the literature these higer-order equivalents of vectors (first order) and matrices (second order) are called higher-order tensors, multidimensional arrays or multiway arrays.

For a lot of applications in science and engineering the existing framework of vectors and matrices appears to be insufficient or inappropiate. There is often a need for more dimensions or a denser representation. In that sense, tensors naturally extend (or generalize) the classical matrix algebra. Consequently, the extension also brings along a need for a transfer of the classical algorithms. This challenging and often non-trivial task forms the focus of ongoing research in the multilinear algebra domain.

An important example of a typical topic is dimensionality reduction, where the goal is to extract invariant subspaces of systems described by (high-dimensional) matrices. With respect to the numerical aspects of Principal Component Analysis (PCA) world-wide scientific efforts are made. This has led to powerful routines (direct or iterative) for the computation of the Eigenvalue Decomposition (EVD), Singular Value Decomposition (SVD), dominant subspaces, etc. of high-dimensional matrices.

In this chapter we aim to contribute to this research area. We derive a Grassmann-Rayleigh Quotient Iteration for the computation of the best rank-(R_1, R_2, R_3) approximation of higher-order tensors [41]. It consists of a generalization to tensors of the Rayleigh Quotient Iteration (RQI) for the computation of an invariant subspace of a given matrix [2]. It also generalizes

the RQI for the best rank-1 approximation of higher-order tensors [200]. Each iteration step of our algorithm only requires to solve a linear set of equations. The rate of convergence is quadratic.

7.1 Introduction

As the starting point we will consider a real Euclidean vector space $\mathbb{R}^I (I \in \mathbb{N})$, with the conventional inner product. Throughout this chapter we will work with real numbers, although extensions to a complex vector space and numbers are possible via according modifications.

A subspace $\mathbb{X}$ of a vector space $\mathbb{V}$, is a subset of $\mathbb{V}$that is closed under linear combinations of elements in that subset. A basis of a subspace is a set of vectors such that each element of the subspace can be represented as a unique linear combination of these basis vectors. The number R of basis vectors is called the dimension of the subspace. The set of all R-dimensional subspaces is called the Grassmann manifold $G(R, I)$.

A matrix representation $\mathbf{A}$ of a subspace $\mathbb{X}$ is an $(I \times R)$ stacking of a particular choice of I-dimensional basisvectors in R columns and is associated with a linear mapping from $\mathbb{R}^R$ to $\mathbb{R}^I$. As such, a subspace under a linear transformation $\mathbf{A} \in \mathbb{R}^{I \times I}$ is mapped into a subspace $\mathbf{A}\mathbb{X} := \{\mathbf{A}X | X \in \mathbb{X}\}$.

The particular nonzero vectors that remain invariant or are reduced (up to a proportionality constant) under the action of $\mathbf{A}$ such that $\mathbf{A}X = \lambda X$, are called eigenvectors of $\mathbf{A}$, where the proportionality constant λ is called the eigenvalue. The subspaces that remain invariant or are reduced (up to a unitary transformation matrix $\mathbf{U}$) under the action of $\mathbf{A}$ such that $\mathbf{A}X = \mathbf{U}\mathbb{X}$, are called eigenspaces of $\mathbf{A}$.

A symmetric matrix $\mathbf{A}$ can be decomposed into a linear combination of rank-1 matrices spanned by the eigenvectors U_i with the corresponding eigenvalues as coefficients (dyadic decomposition). Together the dominant $M < I$ terms form the best rank-M approximation $\hat{\mathbf{A}}$ to the matrix $\mathbf{A}$ with respect to the Euclidean metric. Thus when

$$\hat{\mathbf{A}} = \sum_{i=1}^{M} \lambda_i \, U_i U_i^T, \tag{7.1}$$

then the approximation error $\|\mathbf{A} - \hat{\mathbf{A}}\|$ is minimal over the manifold of rank-M matrices. In many science fields and engineering applications this lower-dimensional dominant eigenspace, or 'best rank-M' approximation to matrix $\mathbf{A}$, plays a key role since it contains the most compressed, characteristic or structural information of the system under study.

7.1.1 The eigenvector of a matrix

We will consider now some methods that refine eigenvector estimates. The *power method* is one of the most simple methods for computing the eigenvectors and eigenvalues. The basic idea is to apply $\mathbf{A}$ repeatedly to an initial unit-norm vector X_0, so that an iteration on the I-dimensional unit sphere arises essentially by successive matrix-vector products:

$$\begin{aligned} Z_k &= \mathbf{A}X_{k-1} \\ X_k &= \frac{Z_k}{\|Z_k\|_2} \\ \lambda_k &= X_k^T \mathbf{A} X_k. \end{aligned} \tag{7.2}$$

The core of the algorithm is that the component of X_k with the *largest* eigenvalue (in absolute value) is repeatedly magnified relative to the other components, such that the one-dimensional dominant eigenspace is obtained. The first treatment of this iteration is attributed to Muntz [128]. The rate of convergence is linear and depends on the ratio of the second to the first eigenvalue [73]. For a complete treatment of the power method we refer to [193].

When one applies the power method to $\mathbf{A}^{-1}$, the *smallest* eigenvalue is computed, and one speaks of *inverse iteration*. There is no need to compute $\mathbf{A}^{-1}$ explicitly. Instead it suffices to solve successive sets of linear equations in an unknown Z_k:

$$\begin{aligned} \mathbf{A}\ Z_k &= X_{k-1} \quad \text{(solve for } Z_k\text{)} \\ X_k &= \frac{Z_k}{\|Z_k\|_2} \\ \lambda_k &= X_k^T \mathbf{A} X_k, \end{aligned} \tag{7.3}$$

starting from given X_0, $\|X_0\| = 1$. In order to let the iteration converge to a particular *selected* value μ, Wielandt [94] proposed to perform an origin shift and thus apply inverse iteration to the matrix $(\mathbf{A} - \mu\ \mathbf{I})$

$$\begin{aligned} (\mathbf{A} - \mu\ \mathbf{I})\ Z_k &= X_{k-1} \quad \text{(solve for } Z_k\text{)} \\ X_k &= \frac{Z_k}{\|Z_k\|_2} \\ \lambda_k &= X_k^T \mathbf{A} X_k. \end{aligned} \tag{7.4}$$

The *shifted inverse iteration* converges generically to the eigenvector of $\mathbf{A}$ with eigenvalue closest to μ. However, the convergence is still linear [73]. A way of accelerating the convergence is to update the shift during each step of the iteration. In particular, if one defines the *Rayleigh quotient* (RQ) as

$$\lambda = r(X) := \frac{X^T \mathbf{A} X}{X^T X}, \tag{7.5}$$

and takes it as shift, with X equal to the most recent eigenvector estimate, then one obtains the *Rayleigh quotient iteration* (RQI) (starting from given X_0, with $\|X_0\| = 1$):

$$\begin{aligned} \mu_k &= \frac{X_k^T \mathbf{A} X_k}{X_k^T X_k} \\ (\mathbf{A} - \mu_k\, \mathbf{I})\; Z_{k+1} &= X_k \quad \text{(solve for } Z_{k+1}) \\ X_{k+1} &= \frac{Z_{k+1}}{\|Z_{k+1}\|_2}. \end{aligned} \tag{7.6}$$

This iteration alternates between computing the RQ (which relates to the eigenvalue) and computing a linear system (which relates to the eigenvector). The latter step assures that better vector iterates are obtained, while the RQ gives each time the best approximate eigenvalue. The Rayleigh quotient iteration converges globally and cubically for symmetric matrices [73]. It dates back to a publication of Crandall in 1951 [34] and is important because of its rapid convergence properties and its potential use in the shifted QR algorithm [132, 191].

The above mentioned algorithms have an important restriction: they yield isolated eigenvectors, and not eigen*spaces*. Especially in the case that some eigenvalues are clustered, and eigenvectors become ill-conditioned (with respect to perturbations on $\mathbf{A}$), this need for obtaining the eigenspace becomes more explicit. This calls for a need of generalizations of the above methods, that would operate on the set of R-dimensional subspaces of $\mathbb{R}^I$ instead of 1-dimensional subspaces.

7.1.2 The eigenspace of a matrix

For the power method and the shifted inverse iteration method, the generalization is straightforward. One just substitutes vector X by a R-dimensional matrix $\mathbf{X}$, corresponding to a sub*space* $\mathbb{X}$ and orthonormalizes. The generalization of the Rayleigh quotient iteration is however much less obvious since the Rayleigh quotient needs to be properly extended.

Recently Absil *et al.* [2] succeeded in solving this problem for matrices by defining first a *matrix* Rayleigh quotient, which is the $(R \times R)$ matrix given by

$$\mathbf{R}(\mathbf{X}) := (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{U}^T\mathbf{A}\mathbf{U}. \tag{7.7}$$

The RQI can then be directly generalized by substitution of the vector $X \in \mathbb{R}^I$ by a matrix $\mathbf{X} \in \mathbb{R}^{I\times R}$ and $r(X)$ by $\mathbf{R}(\mathbf{X})$. Taking the trace gives then a scalar *generalized Rayleigh quotient* (GRQ)

$$r(\mathbf{X}) = tr((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{A}\mathbf{X}), \tag{7.8}$$

that preserves the properties of the classical RQ. Also cubic convergence is proven to be preserved and the operations cost of $\mathcal{O}(IR^2)$ per iteration is low compared to other methods [2]. Through a geometrical approach of the eigenproblem Absil *et al.* show that different generalizations are possible, as well-defined iterations on the Grassmann manifold. In Figure we depict visually the best rank-R approximation that can be obtained with the RQI. For the complete treatment of this topic we refer to [1].

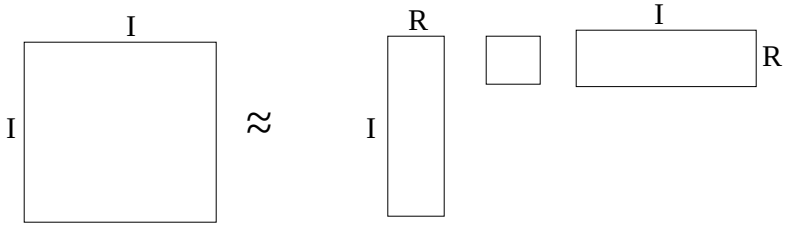


Figure 7.1: Visualization of the best rank-R approximation of an $I \times I$ matrix. Absil *et al.* [2] generalized the computation of one eigenvector of a square matrix by means of the Rayleigh Quotient Iteration to the computation of the $I \times R$ eigenspace of a square matrix.

The classical RQI computes an eigenvector of a symmetric matrix $\mathbf{A}$. One kind of generalization is to have a RQI to compute the eigen*space* of a symmetric matrix $\mathbf{A}$. Another generalization consists of an RQI that would compute the eigenvector of a *tensor* $\mathcal{A}$. This latter issue is the subject of the next section, but first we need to introduce some formal notation and definitions.

7.1.3 Notation and definitions

In this subsection we briefly provide some necessary background, notation conventions and basic definitions that will be useful in the following sections. For notational convenience we mainly focus on real-valued third-order tensors, but in its own right the third-order formulation is important with regard to its direct applicability. The generalization to complex-valued tensors and tensors of order higher than three is straightforward.

Scalars are denoted by lower-case letters (a, b, ...), vectors are written as capitals (A, B, ...), matrices correspond to bold-face capitals ($\mathbf{A}, \mathbf{B}, \ldots$) and tensors are written as calligraphic letters ($\mathcal{A}, \mathcal{B}, \ldots$). In this way, the entry with row index i and column index j in a matrix $\mathbf{A}$, i.e., $(\mathbf{A})_{ij}$, is symbolized by a_{ij}. There is one exception: as we use the characters i, j and r in the meaning of indices (counters), I, J, K and R will be reserved to denote the index upper bounds.

The order of a tensor is determined by the number of indices; an nth-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}$. $\otimes$ denotes the Kronecker product (see (7.9) below).

$\mathbf{I}_N$ is the $N \times N$ identity matrix. $\mathbf{O}(R)$ and $\mathrm{St}(R, I)$ are standard notation for the manifold of $(R \times R)$ orthogonal matrices and the Stiefel manifold of column-wise orthonormal $(I \times R)$ matrices $(I \geqslant R)$, respectively. $\mathrm{qf}(\mathbf{X})$ denotes the orthogonal factor in a QR-decomposition of a matrix $\mathbf{X}$.

Consider a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. We have the following definitions:

- The *Kronecker product* of two matrices $\mathbf{A} \in \mathbb{R}^{n \times p}$ and $\mathbf{B} \in \mathbb{R}^{m \times q}$ is the matrix

$$(\mathbf{A} \otimes \mathbf{B}) = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1p}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2p}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\mathbf{B} & a_{n2}\mathbf{B} & \dots & a_{np}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mn \times pq}. \tag{7.9}$$

  This product has operation order priority over the conventional matrix product (which we mostly don't denote, except sometimes with $\cdot$ for creating clarity in longer expressions), it is not commutative and is sometimes called the *direct product* or the *tensor product*.

- The *matrix unfoldings* $\mathbf{A}_{(1)} \in \mathbb{R}^{I_1 \times I_3 I_2}$, $\mathbf{A}_{(2)} \in \mathbb{R}^{I_2 \times I_1 I_3}$ and $\mathbf{A}_{(3)} \in \mathbb{R}^{I_3 \times I_2 I_1}$ of tensor $\mathcal{A}$ are defined by

$$(\mathbf{A}_{(1)})_{i_1,(i_3-1)I_3+i_2} = (\mathbf{A}_{(2)})_{i_2,(i_1-1)I_1+i_3} = (\mathbf{A}_{(3)})_{i_3,(i_2-1)I_2+i_1} = a_{i_1 i_2 i_3} \tag{7.10}$$

  for all index values.

- The *n-mode product* of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n \mathbf{U}$ is an $(I_1 \times I_2 \times \ldots I_{n-1} \times I_{n+1} \times \ldots \times I_N)$-tensor of which the entries are given by

$$(\mathcal{A} \times_n \mathbf{U})_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} \; u_{j_n i_n}. \tag{7.11}$$

- Consider $\mathbf{U}^{(1)} \in \mathbb{R}^{J_1 \times I_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{J_2 \times I_2}$, $\mathbf{U}^{(3)} \in \mathbb{R}^{J_3 \times I_3}$. Then $\mathcal{B} = \mathcal{A} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$ is a $(J_1 \times J_2 \times J_3)$-tensor of which the entries are given by

$$b_{j_1 j_2 j_3} = \sum_{i_1 i_2 i_3} a_{i_1 i_2 i_3} u^{(1)}_{j_1 i_1} u^{(2)}_{j_2 i_2} u^{(3)}_{j_3 i_3} \;. \tag{7.12}$$

  In terms of the matrix unfoldings, we have, for instance,

$$\mathbf{B}_{(1)} = \mathbf{U}^{(1)} \cdot \mathbf{A}_{(1)} \cdot (\mathbf{U}^{(2)} \otimes \mathbf{U}^{(3)})^T. \tag{7.13}$$

- An *n-mode vector* of $\mathcal{A}$ is an I_n-dimensional vector obtained from $\mathcal{A}$ by varying the index i_n and keeping the other indices fixed. It is a column of $\mathbf{A}_{(n)}$.

- The *n-rank* of a tensor is the obvious generalization of the column (row) rank of matrices: it is defined as the dimension of the vector space spanned by the n-mode vectors and is equal to the rank of $\mathbf{A}_{(n)}$. An important difference with the rank of matrices, is that the different n-ranks of a higher-order tensor are not necessarily the same. A tensor of which the n-ranks are equal to R_n $(1 \leqslant n \leqslant 3)$ is called a *rank-(R_1, R_2, R_3) tensor*. A rank-$(1, 1, 1)$ tensor is briefly called a *rank-1 tensor*. Real-valued tensors are called *supersymmetric* when they are invariant under arbitrary index permutations.
- The *Frobenius-norm* of $\mathcal{A}$ is defined as $\|\mathcal{A}\| = (\sum_{i_1 i_2 i_3} a^2_{i_1 i_2 i_3})^{1/2}$.

For a complete treatment of multilinear algebra, conventions and the choice of representation we refer to [40].

7.2 The dominant k-mode vector of a tensor

The classical Rayleigh quotient iteration provides a 1-dimensional eigenvector X of a symmetric matrix $\mathbf{A}$. As we mentioned above, Absil *et al.* [2] generalized the RQI for the computation of a R-dimensional eigen*space* $\mathbf{X}$ of a matrix $\mathbf{A}$. This allowed to obtain efficiently the best rank-R approximation to a matrix.

De Lathauwer studied a multilinear generalization of the best rank-R approximation problem to *higher-order tensors* [43]. The best rank-one and the best rank-$(R_1, R_2, \ldots, R_n)$ approximation to a tensor $\mathcal{A}$ is computed via higher-order generalizations of the power method and the orthogonal iteration method, respectively. The proposed methods are at most linear and for the best rank-$(R_1, R_2, \ldots, R_n)$ approximation each iteration step consists of solving a partial Singular Value Decomposition (SVD).

In this track of research, Zhang *et al.* [200] have recently proposed a generalized Rayleigh-Newton iteration method to compute the best rank-one approximation to a higher-order tensor $\mathcal{A}$. In Figure 7.2 we depict this visually for a third-order tensor.

In this subsection we will briefly lay out their main line of reasoning.

7.2.1 The generalized Rayleigh quotient

In the case of third order tensors, Zhang *et al.* [200] developed an alternative algorithm for computing the dominant k-mode vector $X^{(k)} \in \mathbb{R}^{I_k \times 1}$ with $k = 1, 2, 3$. Their approach is to introduce the Newton iteration in the the rank-one approximation problem from a least squares primal space formulation point of

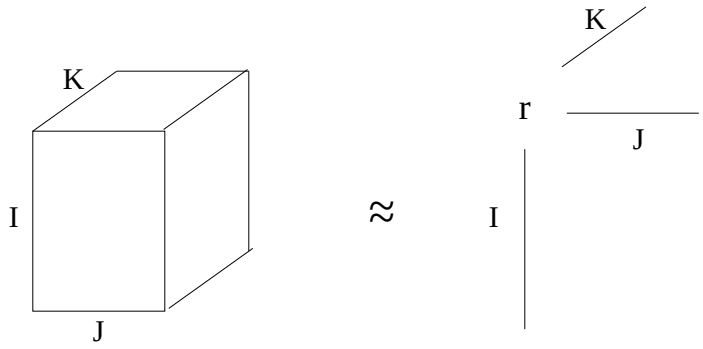


Figure 7.2: Visualization of the best rank-one approximation to an $I \times J \times K$ tensor. Zhang *et al.* [200] generalized the computation of one eigenvector of a square *matrix* by means of the Rayleigh Quotient Iteration to the computation of the dominant rank-1 term of a *tensor*.

view. It is the minimization of the distance between a given tensor and its approximation on the rank-one manifold:

$$\min_{X^{(1)},X^{(2)},X^{(3)}} \sum_{i_1,i_2,i_3} \left(a_{i_1 i_2 i_3} - x_{i_1}^{(1)} x_{i_2}^{(2)} x_{i_3}^{(3)} \right)^2, \tag{7.14}$$

It is remarked that the Rayleigh Quotient is a scalar that functions as a scaling factor λ for the vectors $X^{(1)}, X^{(2)}$ and $X^{(3)}$. One can impose the constraints

$$\|X^{(1)}\|_2 = \|X^{(2)}\|_2 = \|X^{(3)}\|_2 = 1 \tag{7.15}$$

and write (7.14) as

$$\min_{X^{(1)},X^{(2)},X^{(3)}} \sum_{i_1,i_2,i_3} \left(a_{i_1 i_2 i_3} - \lambda x_{i_1}^{(1)} x_{i_2}^{(2)} x_{i_3}^{(3)} \right)^2. \tag{7.16}$$

The scalar *generalized Rayleigh quotient* is then defined as

$$r := r(X^{(1)}, X^{(2)}, X^{(3)}) = \frac{\sum_{i_1,i_2,i_3} a_{i_1 i_2 i_3} \; x_{i_1}^{(1)} x_{i_2}^{(2)} x_{i_3}^{(3)}}{\|X^{(1)}\|_2 \; \|X^{(2)}\|_2 \; \|X^{(3)}\|_2}, \tag{7.17}$$

which takes into account the invariance under scaling of the vectors. When $\|X^{(1)}\|_2 = \|X^{(2)}\|_2 = \|X^{(3)}\|_2 = 1$, then $\lambda = r(X^{(1)}, X^{(2)}, X^{(3)})$ minimizes (7.16), and the minimum value is:

$$\sum_{i_1,i_2,i_3} a_{i_1 i_2 i_3}^2 - r(X^{(1)}, X^{(2)}, X^{(3)})^2 \tag{7.18}$$

Thus (7.14) is equivalent to maximizing $r(X^{(1)}, X^{(2)}, X^{(3)})$ under the constraints (7.15). In other words, the norm of the projection of the given tensor

onto the rank-one manifold is maximized. It is the higher-order generalization of the fact that the computation of the best rank-1 approximation of a matrix $\mathbf{A}$ is equivalent to the determination of the unit-norm vectors U and V such that $\|U^T\mathbf{A}V\|^2$ is maximal [43]. We then obtain the dual problem by using the technique of Lagrange multipliers. Therefore we consider the following combination of r with the constraint terms:

$$\sum_{i_1,i_2,i_3} a_{i_1i_2i_3}\, x_{i_1}^{(1)}x_{i_2}^{(2)}x_{i_3}^{(3)} - \frac{\mu_1}{2}\sum_{i_1}\left(x_{i_1}^{(1)}-1\right) - \frac{\mu_2}{2}\sum_{i_2}\left(x_{i_2}^{(2)}-1\right) - \frac{\mu_3}{2}\sum_{i_3}\left(x_{i_3}^{(3)}-1\right), \tag{7.19}$$

with dual variables μ_1, μ_2, μ_3 (Lagrange multipliers). The optimality conditions per component $x_{i_k}^{(k)}$ yields the system

$$\begin{aligned}
\sum_{i_2,i_3} a_{i_1i_2i_3}\; x_{i_2}^{(2)}x_{i_3}^{(3)} &= \mu_1\; x_{i_1}^{(1)} \\
\sum_{i_1,i_3} a_{i_1i_2i_3}\; x_{i_1}^{(1)}x_{i_3}^{(3)} &= \mu_1\; x_{i_2}^{(2)} \\
\sum_{i_1,i_2} a_{i_1i_2i_3}\; x_{i_1}^{(1)}x_{i_2}^{(2)} &= \mu_1\; x_{i_3}^{(3)}.
\end{aligned} \tag{7.20}$$

To reintroduce r one then multiplies $x_{i_1}^{(1)}, x_{i_2}^{(2)}$ and $x_{i_3}^{(3)}$ to the first, second and third equations and sum over the indices, respectively. This gives $\mu_1 = \mu_2 = \mu_3 = r$ and one may conclude that the above problem (7.14) is equivalent with the system

$$\begin{aligned}
\sum_{i_2,i_3} a_{i_1i_2i_3}\; x_{i_2}^{(2)}x_{i_3}^{(3)} &= r\; x_{i_1}^{(1)} \\
\sum_{i_1,i_3} a_{i_1i_2i_3}\; x_{i_1}^{(1)}x_{i_3}^{(3)} &= r\; x_{i_2}^{(2)} \\
\sum_{i_1,i_2} a_{i_1i_2i_3}\; x_{i_1}^{(1)}x_{i_2}^{(2)} &= r\; x_{i_3}^{(3)} \\
\sum_{i_1i_2,i_3} a_{i_1i_2i_3}\; x_{i_1}^{(1)}x_{i_2}^{(2)}x_{i_3}^{(3)} &= r,
\end{aligned} \tag{7.21}$$

under the constraints (7.15). We have obtained a set of equations for the vectors at the critical points of r. The generalization for n-th order tensors of the above issues was originally elaborated in [43].

7.2.2 The Newton iteration

At this point we have a Rayleigh quotient that yields a best approximate eigenvalue approximation r. But we have no relation between successive iterates. Zhang *et al.* induced this by using Newton's method. In short, Newton's method

[93] gives successive approximation to an equation $f(x) = 0$ in the vicinity of a root based on the first order Taylor expansion of f:

$$x_{k+1} = x_k - (Df(x_k))^{-1} f(x_k), \tag{7.22}$$

where $Df(x_k)$ is the Jacobian of f in x_k. In numerical computations however, the inverse of the Jacobian is never computed and instead one solves a linear system. This splits the iteration into two steps:

$$\begin{aligned} &1. \quad Df(x_k)\ \delta x_k = -f(x_k) \quad \text{(solve for } \delta x_k\text{)} \\ &2. \quad x_{k+1} = x_k + \delta x_k. \end{aligned} \tag{7.23}$$

The idea is then to apply Newton's method, thus to linearize equations (7.21) around the critical point, which yields the first step:

$$\begin{aligned} \sum_{i_2,i_3} a_{i_1 i_2 i_3}\ (x^{(2)}_{i_2}\delta x^{(3)}_{i_3} + x^{(3)}_{i_3}\delta x^{(2)}_{i_2}) - r\delta x^{(1)}_{i_1} &= r x^{(1)}_{i_1} - \sum_{i_2,i_3} a_{i_1 i_2 i_3}\ x^{(2)}_{i_2} x^{(3)}_{i_3} \\ \sum_{i_1,i_3} a_{i_1 i_2 i_3}\ (x^{(1)}_{i_1}\delta x^{(3)}_{i_3} + x^{(3)}_{i_3}\delta x^{(1)}_{i_1}) - r\delta x^{(2)}_{i_2} &= r x^{(2)}_{i_2} - \sum_{i_1,i_3} a_{i_1 i_2 i_3}\ x^{(1)}_{i_1} x^{(3)}_{i_3} \\ \sum_{i_1,i_2} a_{i_1 i_2 i_3}\ (x^{(1)}_{i_1}\delta x^{(2)}_{i_2} + x^{(2)}_{i_2}\delta x^{(1)}_{i_1}) - r\delta x^{(3)}_{i_3} &= r x^{(3)}_{i_3} - \sum_{i_1,i_2} a_{i_1 i_2 i_3}\ x^{(1)}_{i_1} x^{(2)}_{i_2} \end{aligned} \tag{7.24}$$

Remark that nowhere in these equations δr appears, while it is also subject to the linearization through its dependence on $X^{(1)}, X^{(2)}$ and $X^{(3)}$. This is due to a central property of the RQ that for small first order perturbations in the eigenvectors (i.e. in the solutions) the RQ only undergoes variations of second order. It is proven for the generalized RQ in [200]. This is intuitively clear because when the GRQ is optimized in a critical point, the GRQ also has a zero gradient there. It implies also that the fourth equation of (7.21) remains unchanged, so that the value of the RQ seems fixed, which is exactly what is needed in a RQI.

The second step of the Newton iteration introduces the new iterates

$$\bar{x}^{(k)}_{i_k} = x^{(k)}_{i_k} + \delta x^{(k)}_{i_k} \quad k = 1, 2, 3 \tag{7.25}$$

such that we obtain, by substitution in (7.24), the following system:

$$\begin{aligned} -r\bar{x}^{(1)}_{i_1} + \sum_{i_2,i_3} a_{i_1 i_2 i_3}\ (x^{(2)}_{i_2}\bar{x}^{(3)}_{i_3} + x^{(3)}_{i_3}\bar{x}^{(2)}_{i_2}) &= \sum_{i_2,i_3} a_{i_1 i_2 i_3}\ x^{(2)}_{i_2} x^{(3)}_{i_3} \\ -r\bar{x}^{(2)}_{i_2} + \sum_{i_1,i_3} a_{i_1 i_2 i_3}\ (x^{(1)}_{i_1}\bar{x}^{(3)}_{i_3} + x^{(3)}_{i_3}\bar{x}^{(1)}_{i_1}) &= \sum_{i_1,i_3} a_{i_1 i_2 i_3}\ x^{(1)}_{i_1} x^{(3)}_{i_3} \\ -r\bar{x}^{(3)}_{i_3} + \sum_{i_1,i_2} a_{i_1 i_2 i_3}\ (x^{(1)}_{i_1}\bar{x}^{(2)}_{i_2} + x^{(2)}_{i_2}\bar{x}^{(1)}_{i_1}) &= \sum_{i_1,i_2} a_{i_1 i_2 i_3}\ x^{(1)}_{i_1} x^{(2)}_{i_2} \end{aligned} \tag{7.26}$$

This linear system (7.26) in the $\bar{x}^{(k)}_{i_k}$ forms then, combined with (7.17) and the normalization (7.15), a proper generalized Rayleigh quotient iteration scheme for

the best rank-one approximation to tensors. It is also shown that the algorithm locally achieves a quadratic rate of convergence. For the complete treatment we refer to [200].

7.3 The dominant k-mode space of a tensor

The classical Rayleigh quotient iteration provides a 1-dimensional eigenvector X of a symmetric matrix $\mathbf{A}$. As we mentioned above, Absil *et al.* [2] generalized the RQI by computing a *R-dimensional* eigenspace $\mathbf{X}$ of a matrix $\mathbf{A}$. Zhang *et al.* [200] proposed an RQI to compute the best rank-one approximation to a *higher order tensor* $\mathcal{A}$. The logical next step is then to combine both generalizations of the classical RQ and thus compute the dominant k-mode *subspaces* $X^{(k)}$ of a tensor $\mathcal{A}$. In Figure 7.3 we depict this visually for a third order tensor.

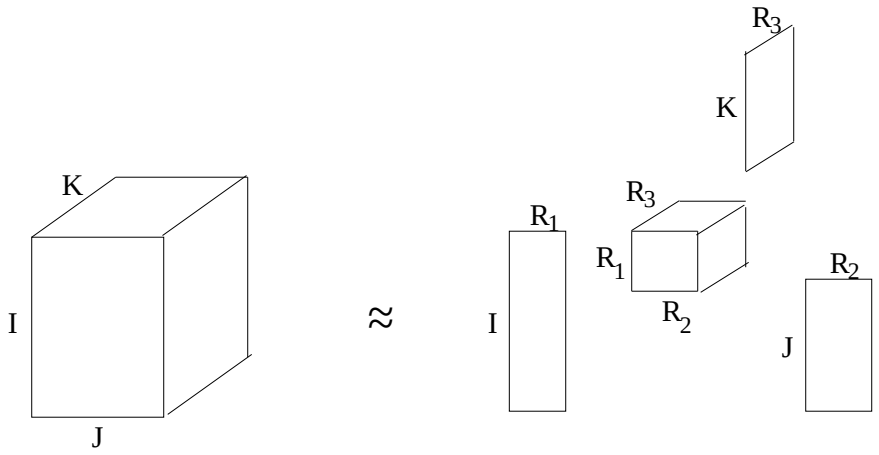


Figure 7.3: Visualization of the best rank-(R_1, R_2, R_3) approximation of an $I \times J \times K$ tensor. We generalized the computation of one eigenvector of a square matrix by means of the Rayleigh Quotient Iteration to the computation of the jointly dominant k-mode subspaces of a tensor.

Although the computation of such subspaces is already shown to be accomplishable via the higher-order Orthogonal Iteration algorithm [42], the specific Rayleigh quotient iteration (with its favorable convergence properties) has been left as an open issue concerning obtaining the jointly dominant subspaces instead of the dominant vectors only. In this chapter we address this more general case. In order to keep the formulas simple, we mainly treat third-order tensors, without loss of generality for the nth-order case.

7.3.1 Higher-Order Grassmann-Rayleigh Quotient Iteration

The goal is to compute the dominant k-mode eigen*spaces* $X^{(k)} \in \mathbb{R}^{I_k \times R_k}$ of a given tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. In other words, we seek a best rank-(R_1, R_2, R_3) approximator $\hat{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ such that the distance (in Frobenius norm) between $\mathcal{A}$ and $\hat{\mathcal{A}}$ is minimized. Consider therefore the minimization of the least-squares cost function

$$f(\hat{\mathcal{A}}) = \|\mathcal{A} - \hat{\mathcal{A}}\|^2, \tag{7.27}$$

under the constraint that $\hat{\mathcal{A}}$ is rank-(R_1, R_2, R_3). This constraint implies that $\hat{\mathcal{A}}$ can be decomposed as

$$\hat{\mathcal{A}} = \mathcal{B} \times_1 \mathbf{X}^{(1)} \times_2 \mathbf{X}^{(2)} \times_3 \mathbf{X}^{(3)}, \tag{7.28}$$

in which $\mathbf{X}^{(n)} \in \mathrm{St}(R_n, I_n)$, $n = 1, 2, 3$, and $\mathcal{B} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$. The minimization of f can be shown [43] to be equivalent to the maximization of

$$g(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}) = \|\mathcal{A} \times_1 \mathbf{X}^{(1)^T} \times_2 \mathbf{X}^{(2)^T} \times_3 \mathbf{X}^{(3)^T}\|^2. \tag{7.29}$$

Thus the norm of the projection of the given tensor $\mathcal{A}$ onto the rank-(R_1, R_2, R_3) manifold is maximized. It is the higher-order generalization of the fact that the computation of the best rank-R approximation of a matrix $\mathbf{A}$ is equivalent to the determination of the unit-norm vectors U_i and V_i such that $\| \sum_{i=2}^{R} U_i^T \mathbf{A} V_i \|^2$ is maximal. The role of $\mathcal{B}$ is not important, because for given $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}$, the optimal $\mathcal{B}$ follows from the linear equation (7.28).

Solving (7.29) with the Lagrange technique becomes lengthy, cumbersome and can be avoided by first writing it alternatively with the the unfolding of $\mathcal{A}$ according to the (first) mode:

$$\|\mathcal{A} \times_1 \mathbf{X}^{(1)^T} \times_2 \mathbf{X}^{(2)^T} \times_3 \mathbf{X}^{(3)^T}\|^2 = \|\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)})\|^2. \tag{7.30}$$

This formulation allows to arrive more directly at optimality conditions. Assume that $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(3)}$ are fixed. From (7.30) we see that $\mathbf{X}^{(1)}$ can only be optimal if its columns span the same subspace as the R_1 dominant left singular vectors of $\tilde{\mathbf{A}}_{(1)} = \mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)})$. A necessary condition is that the column space of $\mathbf{X}^{(1)}$ is an invariant subspace of $\tilde{\mathbf{A}}_{(1)} \cdot \tilde{\mathbf{A}}_{(1)}^T$. Similar conditions can be derived for the other modes. We obtain:

$$\mathbf{X}^{(1)} \cdot \mathbf{W}_1 = \mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)}) \cdot (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)})^T \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)} \tag{7.31}$$

$$\mathbf{X}^{(2)} \cdot \mathbf{W}_2 = \mathbf{A}_{(2)} \cdot (\mathbf{X}^{(3)} \otimes \mathbf{X}^{(1)}) \cdot (\mathbf{X}^{(3)} \otimes \mathbf{X}^{(1)})^T \cdot \mathbf{A}_{(2)}^T \cdot \mathbf{X}^{(2)} \tag{7.32}$$

$$\mathbf{X}^{(3)} \cdot \mathbf{W}_3 = \mathbf{A}_{(3)} \cdot (\mathbf{X}^{(1)} \otimes \mathbf{X}^{(2)}) \cdot (\mathbf{X}^{(1)} \otimes \mathbf{X}^{(2)})^T \cdot \mathbf{A}_{(3)}^T \cdot \mathbf{X}^{(3)} \tag{7.33}$$

for some $\mathbf{W}_1 \in \mathbb{R}^{R_1 \times R_1}$, $\mathbf{W}_2 \in \mathbb{R}^{R_2 \times R_2}$, $\mathbf{W}_3 \in \mathbb{R}^{R_3 \times R_3}$, as determination of the $\mathbf{X}^{(k)}$ is up to a unitary transformation.

This set of equations forms the starting point for the derivation of our new algorithm. Note that only the column spaces of $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(3)}$ are of importance, and not their individual columns. This means that we are actually working on Grassmann manifolds [52]. This has the practical consequence that we work with normalized vectors and matrices.

7.3.2 The Generalized Rayleigh Quotient

For $X \in \mathrm{St}(1, I) \setminus \{0\}$ and a symmetric matrix $\mathbf{A} \in \mathbb{R}^{I \times I}$, the classical scalar Rayleigh quotient [132] was defined as

$$r(X) = X^T \mathbf{A} X \tag{7.34}$$

For $\mathbf{X} \in \mathrm{St}(R, I)$, the *matrix* Rayleigh quotient [2] is defined as

$$\mathbf{R}(\mathbf{X}) = \mathbf{X}^T \mathbf{A} \mathbf{X}, \tag{7.35}$$

with $\mathrm{trace}(\mathbf{R}(\mathbf{X}))$ as the scalar Rayleigh quotient.

For $\mathbf{X}^{(k)} \in \mathrm{St}(1, I)$, $k = 1, 2, 3$, and $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ the scalar generalized Rayleigh quotient [200] was defined as

$$r(X^{(1)}, X^{(2)}, X^{(3)}) = \mathcal{A} \times_1 {X^{(1)}}^T \times_2 {X^{(2)}}^T \times_3 {X^{(3)}}^T. \tag{7.36}$$

For $\mathbf{X}^{(1)} \in \mathrm{St}(R_1, I_1)$, $\mathbf{X}^{(2)} \in \mathrm{St}(R_2, I_2)$, $\mathbf{X}^{(3)} \in \mathrm{St}(R_3, I_3)$ and $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ we define *n-mode Rayleigh quotient matrices* as follows:

$$\mathbf{R}_1(\mathbf{X}) = {\mathbf{X}^{(1)}}^T \cdot \mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)}) \tag{7.37}$$

$$\mathbf{R}_2(\mathbf{X}) = {\mathbf{X}^{(2)}}^T \cdot \mathbf{A}_{(2)} \cdot (\mathbf{X}^{(3)} \otimes \mathbf{X}^{(1)}) \tag{7.38}$$

$$\mathbf{R}_3(\mathbf{X}) = {\mathbf{X}^{(3)}}^T \cdot \mathbf{A}_{(3)} \cdot (\mathbf{X}^{(1)} \otimes \mathbf{X}^{(2)}) \ , \tag{7.39}$$

with $\mathrm{trace}(\mathbf{R}_k(\mathbf{X})^T \cdot \mathbf{R}_k(\mathbf{X})) = \|\mathbf{R}_k(\mathbf{X})\|_F$ as the scalar generalizaed k-mode Rayleigh quotient. This latter definition properly generalizes the existing definitions (7.34–7.36) of Rayleigh quotients associated with an eigenvector, invariant subspace or tensor rank-1 approximation [2, 200].

7.3.3 A central property of the Rayleigh quotient

Before we proceed in an analogous manner as above with the linearization of (7.31–7.33) for the Newton method, we first ascertain in the following that linear perturbations in the eigenspaces will have only a second order effect on the RQ entries.

Let $\mathbf{X}^{(1)} \in \mathrm{St}(R_1, I_1)$, $\mathbf{X}^{(2)} \in \mathrm{St}(R_2, I_2)$, $\mathbf{X}^{(3)} \in \mathrm{St}(R_3, I_3)$ be solutions to (7.31)-(7.33). Then, for small perturbations $\Delta\mathbf{X}^{(1)}$, $\Delta\mathbf{X}^{(2)}$, $\Delta\mathbf{X}^{(3)}$ satisfying

$$\mathbf{X}^{(1)^T}\Delta\mathbf{X}^{(1)} = \mathbf{0}, \quad \mathbf{X}^{(2)^T}\Delta\mathbf{X}^{(2)} = \mathbf{0}, \quad \mathbf{X}^{(3)^T}\Delta\mathbf{X}^{(3)} = \mathbf{0}, \tag{7.40}$$

we have

$$\|\mathbf{R}_n(\mathbf{X})\cdot\mathbf{R}_n(\mathbf{X})^T - \mathbf{R}_n(\mathbf{X}+\Delta\mathbf{X})\cdot\mathbf{R}_n(\mathbf{X}+\Delta\mathbf{X})^T\| = O(\|\Delta\mathbf{X}\|^2) \quad n = 1,2,3\ . \tag{7.41}$$

Let us consider the case $n = 1$. The cases $n = 2, 3$ are completely similar.

By definition, we have

$$\begin{aligned}
&\mathbf{R}_1(\mathbf{X}+\Delta\mathbf{X})\cdot\mathbf{R}_1(\mathbf{X}+\Delta\mathbf{X})^T = \\
&\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+(\Delta\mathbf{X}^{(1)})^T\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\Delta\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\Delta\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\Delta\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\Delta\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)} \\
&+\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\cdot(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})^T\cdot\mathbf{A}_{(1)}^T\cdot\Delta\mathbf{X}^{(1)} + O(\|\Delta\mathbf{X}\|^2)\ .
\end{aligned}$$

In this expansion the first term equals $\mathbf{R}_1(\mathbf{X})\cdot\mathbf{R}_1(\mathbf{X})^T$. The first-order terms vanish, because of (7.31)-(7.33) and (7.40).

7.3.4 The Newton iteration

So now we can consider perturbations $\Delta\mathbf{X}^{(1)}$, $\Delta\mathbf{X}^{(2)}$, $\Delta\mathbf{X}^{(3)}$ satisfying (7.40). Using the above central property of the RQ and saying that $\mathbf{W}_1 = \mathbf{R}_1(\mathbf{X})\cdot\mathbf{R}_1(\mathbf{X})^T$ is only subject to second-order perturbationswe have the following linear expansion of (7.31):

$$\begin{aligned}
&(\mathbf{X}^{(1)}+\Delta\mathbf{X}^{(1)})\cdot\mathbf{R}_1(\mathbf{X})\cdot\mathbf{R}_1(\mathbf{X})^T = \\
&\mathbf{A}_{(1)}\cdot\left[(\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T})\otimes(\mathbf{X}^{(3)}\cdot\mathbf{X}^{(3)^T})\right]\cdot\mathbf{A}_{(1)}^T\cdot(\mathbf{X}^{(1)}+\Delta\mathbf{X}^{(1)}) + \\
&\mathbf{A}_{(1)}\cdot\Big[(\Delta\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T})\otimes(\mathbf{X}^{(3)}\cdot\mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)}\cdot\Delta\mathbf{X}^{(2)^T})\otimes(\mathbf{X}^{(3)}\cdot\mathbf{X}^{(3)^T}) + \\
&(\mathbf{X}^{(2)}\mathbf{X}^{(2)^T})\otimes(\Delta\mathbf{X}^{(3)}\mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)}\mathbf{X}^{(2)^T})\otimes(\mathbf{X}^{(3)}\Delta\mathbf{X}^{(3)^T})\Big]\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)}
\end{aligned} \tag{7.42}$$

In order to include the second step of the Newton method, we let the (approximate) true solution be given by

$$\overline{\mathbf{X}}^{(n)} = \mathbf{X}^{(n)} + \Delta\mathbf{X}^{(n)} \quad \text{for} \quad n = 1,2,3. \tag{7.43}$$

For iterations on the Stiefel manifold (we assume normalization after each iteration), it can be justified that the conditions (7.40) are indeed met. For this we consider a decomposition of the perturbation parallel and perpendicular to $\mathbf{X}^{(n)}$. It is well-known [52] that, for $\overline{\mathbf{X}}^{(n)}$ to be on the Stiefel manifold, the perturbation can up to first order terms be decomposed as in

$$\overline{\mathbf{X}}^{(n)} = \mathbf{X}^{(n)}(\mathbf{I} + \Delta\mathbf{E}_1^{(n)}) + (\mathbf{X}^{\perp})^{(n)}\Delta\mathbf{E}_2^{(n)}, \tag{7.44}$$

in which $\Delta\mathbf{E}_1^{(n)} \in \mathbb{R}^{R_n \times R_n}$ is skew-symmetric and $(\mathbf{X}^{\perp})^{(n)} \in \mathrm{St}(I_n - R_n, I_n)$ perpendicular to $\mathbf{X}^{(n)}$. As a first order approximation we have now

$$\overline{\mathbf{X}}^{(n)} \cdot (\mathbf{I} - \Delta\mathbf{E}_1^{(n)}) = \mathbf{X}^{(n)} + (\mathbf{X}^{\perp})^{(n)}\Delta\mathbf{E}_2^{(n)} \ . \tag{7.45}$$

Because of the skew symmetry of $\Delta\mathbf{E}_1^{(n)}$, the matrix $\overline{\mathbf{X}}^{(n)} \cdot (\mathbf{I} - \Delta\mathbf{E}_1^{(n)})$ is in first order column-wise orthonormal, and it has the same column space as $\overline{\mathbf{X}}^{(n)}$. Because only this column space is of importance (and not the individual columns), the perturbation effectively is consisting of a perpendicular part only, which (7.45) implies that we can indeed limit ourselves to perturbations satisfying (7.40).

From (7.42) we have, using relation (7.43), that

$$\begin{aligned}
&\overline{\mathbf{X}}^{(1)} \cdot \mathbf{R}_1(\mathbf{X}) \cdot \mathbf{R}_1(\mathbf{X})^T = \\
&\mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) \cdot \mathbf{A}_{(1)}^T \cdot (\overline{\mathbf{X}}^{(1)} - 4\mathbf{X}^{(1)}) + \\
&\mathbf{A}_{(1)} \cdot \Big[(\overline{\mathbf{X}}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)} \cdot \overline{\mathbf{X}}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) + \\
&(\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\overline{\mathbf{X}}^{(3)} \cdot \mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \overline{\mathbf{X}}^{(3)^T}) \Big] \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)}
\end{aligned} \tag{7.46}$$

Exploiting the symmetry of the problem, we obtain similar expressions for the 2-mode and 3-mode Rayleigh quotient matrices. The global set consists of linear equations in $\overline{\mathbf{X}}^{(1)}$, $\overline{\mathbf{X}}^{(2)}$, $\overline{\mathbf{X}}^{(3)}$. This means that it can be written in the form

$$\mathbf{M}_{\mathcal{A},\mathbf{X}}\overline{X} = B_{\mathcal{A},\mathbf{X}}, \tag{7.47}$$

in which the coefficients of matrix $\mathbf{M}_{\mathcal{A},\mathbf{X}} \in \mathbb{R}^{(I_1R_1+I_2R_2+I_3R_3)\times(I_1R_1+I_2R_2+I_3R_3)}$ and column vector $B_{\mathcal{A},\mathbf{X}} \in \mathbb{R}^{I_1R_1+I_2R_2+I_3R_3}$ depend on $\mathcal{A}$ and $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$ and in which the coefficients of $\overline{\mathbf{X}}^{(1)}$, $\overline{\mathbf{X}}^{(2)}$, $\overline{\mathbf{X}}^{(3)}$ are stacked in $\overline{X}$. (Explicit expressions for $\mathbf{M}_{\mathcal{A},\mathbf{X}}$ and $B_{\mathcal{A},\mathbf{X}}$ are given in the next section.) Hence, given $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$ and the associated n-mode Rayleigh quotient matrices, $\overline{\mathbf{X}}^{(1)}$, $\overline{\mathbf{X}}^{(2)}$, $\overline{\mathbf{X}}^{(3)}$ can be estimated by solving a square linear set of equations in $I_1R_1 + I_2R_2 + I_3R_3$ unknowns.

The resulting algorithm is summarized in Table 7.1. The algorithm can be initialized with the truncated components of the Higher-Order Singular Value

Decomposition [42]. This means that the columns of $\overline{\mathbf{X}}_0^{(n)}$ are taken equal to the dominant left singular vectors of $\mathbf{A}_{(n)}$, $n = 1, 2, 3$. In [173, 42] it is explained that this truncation yields a good, but not the optimal approximation (in contrast to the matrix case).

7.3.5 Convergence rate

It can be shown that the convergence of our algorithm (see Table 7.1) is quadratic. Let $\overline{\mathbf{X}}^{(1)}, \overline{\mathbf{X}}^{(2)}, \overline{\mathbf{X}}^{(3)}, \mathbf{R}_1(\overline{\mathbf{X}}), \mathbf{R}_2(\overline{\mathbf{X}}), \mathbf{R}_3(\overline{\mathbf{X}})$ correspond to a nonzero solution to (7.31–7.33). If $\mathbf{M}_{\mathcal{A},\overline{\mathbf{X}}}$ is nonsingular, then our algorithm (Table 7.1) converges to $(\overline{\mathbf{X}}^{(1)}\mathbf{Q}_1, \overline{\mathbf{X}}^{(2)}\mathbf{Q}_2, \overline{\mathbf{X}}^{(3)}\mathbf{Q}_3)$, with $\mathbf{Q}_1 \in \mathbf{O}(R_1)$, $\mathbf{Q}_2 \in \mathbf{O}(R_2)$, $\mathbf{Q}_3 \in \mathbf{O}(R_3)$, quadratically in a neighbourhood of $(\overline{\mathbf{X}}^{(1)}, \overline{\mathbf{X}}^{(2)}, \overline{\mathbf{X}}^{(3)})$.

Because $\overline{\mathbf{X}}^{(1)}, \overline{\mathbf{X}}^{(2)}, \overline{\mathbf{X}}^{(3)}, \mathbf{R}_1(\overline{\mathbf{X}}), \mathbf{R}_2(\overline{\mathbf{X}}), \mathbf{R}_3(\overline{\mathbf{X}})$ give a solution to (7.31)-(7.33), we have

$$\mathbf{M}_{\mathcal{A},\overline{\mathbf{X}}}\overline{X} - B_{\mathcal{A},\overline{\mathbf{X}}} = \mathbf{0} \ . \tag{7.48}$$

Consider from (7.43) the relations

$$\begin{aligned} \mathbf{X}^{(1)} &= \overline{\mathbf{X}}^{(1)} - \Delta\mathbf{X}^{(1)}, \\ \mathbf{X}^{(2)} &= \overline{\mathbf{X}}^{(2)} - \Delta\mathbf{X}^{(2)}, \\ \mathbf{X}^{(3)} &= \overline{\mathbf{X}}^{(3)} - \Delta\mathbf{X}^{(3)}, \end{aligned} \tag{7.49}$$

with $\Delta\mathbf{X}^{(1)}$, $\Delta\mathbf{X}^{(2)}$, $\Delta\mathbf{X}^{(3)}$ satisfying (7.40). Because of the above central property of the RQ and (7.46) we have

$$\mathbf{M}_{\mathcal{A},\mathbf{X}}\overline{X} - B_{\mathcal{A},\mathbf{X}} = \mathcal{O}(\|\Delta\mathbf{X}\|^2) \ . \tag{7.50}$$

Because $\mathbf{M}_{\mathcal{A},\overline{\mathbf{X}}}$ is nonsingular, we can write:

$$\begin{aligned} \left(\|\Delta\overline{\mathbf{X}}_{k+1}^{(1)}\|^2 + \|\Delta\overline{\mathbf{X}}_{k+1}^{(2)}\|^2 + \|\Delta\overline{\mathbf{X}}_{k+1}^{(3)}\|^2\right)^{1/2} &= \|\overline{X} - \overline{X}_{k+1}\| \\ &= \|\overline{X} - \mathbf{M}_{\mathcal{A},\mathbf{X}_k}^{-1} B_{\mathcal{A},\mathbf{X}_k}\| \\ &= \mathcal{O}(\|\mathbf{M}_{\mathcal{A},\mathbf{X}_k} B_{\mathcal{A},\mathbf{X}_k} - \overline{X}\|) \\ &= \mathcal{O}(\|\Delta\mathbf{X}_k\|^2) \ . \end{aligned} \tag{7.51}$$

This equation indicates that the convergence is quadratic. Finally, we verify that

$$\|\Delta\overline{\mathbf{X}}_{k+1}^{(n)}\|^2 = \mathcal{O}(\min_{\mathbf{Q}\in\mathbf{O}(R_n)} \|\mathrm{qf}(\overline{\mathbf{X}}_{k+1}^{(n)}) - \mathbf{X}^{(n)}\mathbf{Q}\|^2), \quad n = 1, 2, 3 \ . \tag{7.52}$$

This means that the normalization in step 1 of Algorithm 7.1 does not decrease the convergence rate. This quadratic rate of convergence is a big improvement over the alternative existing algorithms discussed in [43], of which the convergence is at most linear.

Given initial estimates $\overline{\mathbf{X}}_0^{(1)} \in \mathbb{R}^{I_1 \times R_1}$, $\overline{\mathbf{X}}_0^{(2)} \in \mathbb{R}^{I_2 \times R_2}$, $\overline{\mathbf{X}}_0^{(3)} \in \mathbb{R}^{I_3 \times R_3}$
Iterate until convergence:

1. Normalize to matrices on Stiefel manifold:

$$\mathbf{X}_k^{(1)} = \text{qf}(\overline{\mathbf{X}}_k^{(1)}) \qquad \mathbf{X}_k^{(2)} = \text{qf}(\overline{\mathbf{X}}_k^{(2)}) \qquad \mathbf{X}_k^{(3)} = \text{qf}(\overline{\mathbf{X}}_k^{(3)})$$

2. Compute n-mode Rayleigh quotient matrices:

$$\begin{aligned} \mathbf{R}_1(\mathbf{X}_k) &= \mathbf{X}_k^{(1)^T} \cdot \mathbf{A}_{(1)} \cdot (\mathbf{X}_k^{(2)} \otimes \mathbf{X}_k^{(3)}) \\ \mathbf{R}_2(\mathbf{X}_k) &= \mathbf{X}_k^{(2)^T} \cdot \mathbf{A}_{(2)} \cdot (\mathbf{X}_k^{(3)} \otimes \mathbf{X}_k^{(1)}) \\ \mathbf{R}_3(\mathbf{X}_k) &= \mathbf{X}_k^{(3)^T} \cdot \mathbf{A}_{(3)} \cdot (\mathbf{X}_k^{(1)} \otimes \mathbf{X}_k^{(2)}) \end{aligned}$$

3. Solve the linear set of equations

$$\mathbf{M}_{\mathcal{A},\mathbf{X}_k} \overline{X}_{k+1} = B_{\mathcal{A},\mathbf{X}_k}$$

Table 7.1: A high-level GRQI formulation for the computation of the best rank-(R_1, R_2, R_3) approximation of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. The detailed solution of the linear system is worked out in Section 7.4.

7.4 Computing the set of linear equations

In this section we will provide explicit expressions for $\mathbf{M}_{\mathcal{A},\mathbf{X}}$ and $B_{\mathcal{A},\mathbf{X}}$ in (7.47). Behind the concise formulation of (7.47), some considerable implementation work is hidden. The frequently occuring Kronecker products in (7.46) must be unraveled and grouped in order to make computation feasible. First we briefly provide a set of calculus rules that are necessary in order to derive the linear set of equations.

7.4.1 Calculus rules and definitions

In this section, we review some of the properties of tensor products, introduce some definitions and notations, in preparation for some of the manipulations that will follow.

- For matrices $\mathbf{A} \in \mathbb{R}^{n \times p}$,$\mathbf{B} \in \mathbb{R}^{m \times q}$ and compatible matrices $\mathbf{C}$ and $\mathbf{D}$, we have
$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}). \tag{7.53}$$
- Multiplication by a scalar a:
$$a(\mathbf{A} \otimes \mathbf{B}) = (a\mathbf{A}) \otimes \mathbf{B},$$

$$(\mathbf{A} \otimes \mathbf{B})a = \mathbf{A} \otimes (\mathbf{B}a).$$

- Associativity:
$$(\mathbf{A} \otimes \mathbf{B}) \otimes C = \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}). \tag{7.54}$$

- Transposition:
$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T. \tag{7.55}$$

- The *vec* operator creates a column vector from a matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$ by stacking the p column vectors of $\mathbf{A} = [\mathbf{A}_1 \mathbf{A}_2 \dots \mathbf{A}_p]$ below one another:
$$\mathrm{vec}(\mathbf{A}) = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_p \end{bmatrix} \in \mathbb{R}^{np \times 1}. \tag{7.56}$$

- The *vec* operator (of a product of matrices) and the tensor product are related by
$$\mathrm{vec}(\mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B}). \tag{7.57}$$
which implies that also $\mathrm{vec}(\mathbf{AB}) = (I_B \otimes \mathbf{A}) \cdot \mathrm{vec}(\mathbf{B}) = (\mathbf{B}^T \otimes I_A) \cdot \mathrm{vec}(\mathbf{A})$.

- $\mathbf{P}_{I,J} \in \mathbb{R}^{IJ \times IJ}$ is a permutation matrix of which the nonzero entries are given by
$$\mathbf{P}_{I,J}\left((i-1)J + j, (j-1)I + i\right) = 1 \qquad \text{for} \qquad 1 \le i \le I, \quad 1 \le i \le J. \tag{7.58}$$

- For a given matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ we have
$$\mathrm{vec}(\mathbf{A}^T) = \mathbf{P}_{I,J} \cdot \mathrm{vec}(\mathbf{A}). \tag{7.59}$$

- $\mathbf{K}_{n,I,N} \in \mathbb{R}^{NI \times I}$ is the nth block column of $\mathbf{I}_{NI \times NI}$.

- $\mathbf{L}_{I,J,N} \in \mathbb{R}^{IJNN \times IJ}$ is defined as
$$\mathbf{L}_{I,J,N} = \begin{bmatrix} \mathbf{I}_{J \times J} \otimes \mathbf{K}_{1,I,N} \\ \mathbf{I}_{J \times J} \otimes \mathbf{K}_{2,I,N} \\ \vdots \\ \mathbf{I}_{J \times J} \otimes \mathbf{K}_{N,I,N} \end{bmatrix}. \tag{7.60}$$

- The following decomposition is useful for isolating a separate *vec* operation (on the right-hand argument matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$):
$$\mathrm{vec}(\mathbf{I}_{N \times N} \otimes \mathbf{A}) = \mathbf{L}_{I,J,N} \cdot \mathrm{vec}(\mathbf{A}). \tag{7.61}$$

- $\mathbf{R}_{I,N} \in \mathbb{R}^{INN \times I}$ is defined as

$$[\ \mathrm{vec}(\mathbf{K}_{1,N,I}), \mathrm{vec}(\mathbf{K}_{2,N,I}), \ldots, \mathrm{vec}(\mathbf{K}_{I,N,I})\ ]\,. \tag{7.62}$$

- The following decomposition is useful for isolating a separate *vec* operation (on the left-hand argument matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$):

$$\mathrm{vec}(\mathbf{A} \otimes \mathbf{I}_{N \times N}) = (\mathbf{I}_{J \times J} \otimes \mathbf{R}_{I,N}) \cdot \mathrm{vec}(\mathbf{A}). \tag{7.63}$$

- A useful example of the combined use of these relations is given by

$$\begin{aligned}
&\mathrm{vec}\,(\mathbf{A} \otimes \mathbf{B}) \\
&= \mathrm{vec}\,(\mathbf{I}_n\mathbf{A} \otimes \mathbf{B}\mathbf{I}_q) \\
&= \mathrm{vec}\,((\mathbf{I}_n \otimes \mathbf{B}) \cdot (\mathbf{A} \otimes \mathbf{I}_q) \cdot \mathbf{I}_{pq}) \\
&= [\mathbf{I}_{pq} \otimes (\mathbf{I}_n \otimes \mathbf{B})] \cdot \mathrm{vec}\,(\mathbf{A} \otimes \mathbf{I}_q) \\
&= [\mathbf{I}_{pq} \otimes (\mathbf{I}_n \otimes \mathbf{B})] \cdot (\mathbf{I}_p \otimes \mathbf{R}_{n,q}) \cdot \mathrm{vec}(\mathbf{A}).
\end{aligned}$$

7.4.2 Explicit expression of the linear equations

In order to separate the parts with the unknowns $\overline{\mathbf{X}}^{(1)}, \overline{\mathbf{X}}^{(2)}, \overline{\mathbf{X}}^{(3)}$ and the previous iterates $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$, we appropriately vectorize the subsequent terms of (7.46):

$$\begin{gathered}
\overline{\mathbf{X}}^{(1)} \cdot \mathbf{R}_1(\mathbf{X}) \cdot \mathbf{R}_1(\mathbf{X})^T = \\
\mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) \cdot \mathbf{A}_{(1)}^T \cdot (\overline{\mathbf{X}}^{(1)} - 4\mathbf{X}^{(1)}) + \\
\mathbf{A}_{(1)} \cdot \Big[(\overline{\mathbf{X}}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)} \cdot \overline{\mathbf{X}}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) + \\
(\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\overline{\mathbf{X}}^{(3)} \cdot \mathbf{X}^{(3)^T}) + (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \overline{\mathbf{X}}^{(3)^T}) \Big] \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)}.
\end{gathered}$$

For the first term on the left hand side of equation (7.46) we have

$$\begin{aligned}
&\mathrm{vec}\left(\overline{\mathbf{X}}^{(1)} \cdot \mathbf{R}_1(\mathbf{X}) \cdot \mathbf{R}_1(\mathbf{X})^T\right) \\
&= \left(\left(\mathbf{R}_1(\mathbf{X}) \cdot \mathbf{R}_1(\mathbf{X})^T\right)^T \otimes \mathbf{I}_{I_1}\right) \cdot \mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right) \\
&= \mathbf{M}_{1,1b} \cdot \mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right).
\end{aligned} \tag{7.64}$$

For the first term on the right hand side of equation (7.46) we have

$$\begin{aligned}
&\mathrm{vec}\left(\mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) \cdot \mathbf{A}_{(1)}^T \cdot \overline{\mathbf{X}}^{(1)}\right) \\
&= \left[\mathbf{I}_{R_1} \otimes \left(\mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) \cdot \mathbf{A}_{(1)}^T\right)\right] \cdot \mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right) \\
&= \mathbf{M}_{1,1a} \cdot \mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right).
\end{aligned} \tag{7.65}$$

For the second term on the right hand side of equation (7.46) we have

$$\text{vec}\left(\mathbf{A}_{(1)} \cdot (\mathbf{X}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}) \cdot \mathbf{A}_{(1)}^T \cdot (-4\mathbf{X}^{(1)})\right) = T_1. \tag{7.66}$$

For the third term on the right hand side of equation (7.46) we have

$$\begin{aligned}
&\text{vec}\left(\mathbf{A}_{(1)} \cdot \left[(\overline{\mathbf{X}}^{(2)} \cdot \mathbf{X}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T})\right] \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)}\right) \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \text{vec}\left\{\left(\overline{\mathbf{X}}^{(2)} \cdot \mathbf{X}^{(2)^T}\right) \otimes \left(\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}\right)\right\} \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \text{vec}\left\{\left(\overline{\mathbf{X}}^{(2)} \otimes \mathbf{X}^{(3)}\right) \cdot \left(\mathbf{X}^{(2)^T} \otimes \mathbf{X}^{(3)^T}\right)\right\} \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \left((\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)}) \otimes \mathbf{I}_{I_2 I_3}\right) \cdot \text{vec}\left(\overline{X}^{(2)} \otimes \mathbf{X}^{(3)}\right) \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \left((\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)}) \otimes \mathbf{I}_{I_2 I_3}\right) \cdot \\
&\quad \left(\mathbf{I}_{R_2 R_3} \otimes (\mathbf{I}_{I_2} \otimes \mathbf{X}^{(3)})\right) \cdot (\mathbf{I}_{R_2} \otimes \mathbf{R}_{I_2,R_3}) \cdot \text{vec}\left(\overline{\mathbf{X}}^{(2)}\right) \\
&= \mathbf{M}_{1,2a} \cdot \text{vec}\left(\overline{\mathbf{X}}^{(2)}\right).
\end{aligned} \tag{7.67}$$

For the fourth term on the right hand side of equation (7.46) we have

$$\begin{aligned}
&\text{vec}\left(\mathbf{A}_{(1)} \cdot \left[(\mathbf{X}^{(2)} \cdot \overline{\mathbf{X}}^{(2)^T}) \otimes (\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T})\right] \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)}\right) \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \text{vec}\left\{\left(\mathbf{X}^{(2)} \cdot \overline{\mathbf{X}}^{(2)^T}\right) \otimes \left(\mathbf{X}^{(3)} \cdot \mathbf{X}^{(3)^T}\right)\right\} \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \left(\mathbf{I}_{I_2 I_3} \otimes (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)})\right) \cdot \text{vec}\left(\overline{\mathbf{X}}^{(2)^T} \otimes \mathbf{X}^{(3)^T}\right) \\
&= \left[\left(\mathbf{X}^{(1)^T} \cdot \mathbf{A}_{(1)}\right) \otimes \mathbf{A}_{(1)}\right] \cdot \left(\mathbf{I}_{I_2 I_3} \otimes (\mathbf{X}^{(2)} \otimes \mathbf{X}^{(3)})\right) \cdot \\
&\quad \mathbf{P}_{I_2 I_3, R_2 R_3} \cdot \left(I_{R_2 R_3} \otimes (\mathbf{I}_{I_2} \otimes \mathbf{X}^{(3)})\right) \cdot (\mathbf{I}_{R_2} \otimes \mathbf{R}_{I_2,R_3}) \cdot \text{vec}\left(\overline{\mathbf{X}}^{(2)}\right) \\
&= \mathbf{M}_{1,2b} \cdot \text{vec}\left(\overline{\mathbf{X}}^{(2)}\right).
\end{aligned} \tag{7.68}$$

For the fifth term on the right hand side of equation (7.46) we have

$$\begin{aligned}
&\mathrm{vec}\left(\mathbf{A}_{(1)}\cdot\left[(\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T})\otimes(\overline{\mathbf{X}}^{(3)}\cdot\mathbf{X}^{(3)^T})\right]\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)}\right)\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\mathrm{vec}\left\{\left(\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T}\right)\otimes\left(\overline{\mathbf{X}}^{(3)}\cdot\mathbf{X}^{(3)^T}\right)\right\}\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\left((\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\otimes\mathbf{I}_{I_2I_3}\right)\cdot\mathrm{vec}\left(\mathbf{X}^{(2)}\otimes\overline{\mathbf{X}}^{(3)}\right)\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\left((\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\otimes\mathbf{I}_{I_2I_3}\right)\cdot\\
&\qquad\left(\mathbf{I}_{R_2R_3}\otimes(\mathbf{X}^{(2)}\otimes\mathbf{I}_{I_3})\right)\cdot\mathbf{L}_{I_2,R_3,R_2}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right)\\
&=\mathbf{M}_{1,3a}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right).
\end{aligned}\tag{7.69}$$

For the sixth term on the right hand side of equation (7.46) we have

$$\begin{aligned}
&\mathrm{vec}\left(\mathbf{A}_{(1)}\cdot\left[(\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T})\otimes(\mathbf{X}^{(3)}\cdot\overline{\mathbf{X}}^{(3)^T})\right]\cdot\mathbf{A}_{(1)}^T\cdot\mathbf{X}^{(1)}\right)\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\mathrm{vec}\left\{\left(\mathbf{X}^{(2)}\cdot\mathbf{X}^{(2)^T}\right)\otimes\left(\mathbf{X}^{(3)}\cdot\overline{\mathbf{X}}^{(3)^T}\right)\right\}\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\left(\mathbf{I}_{I_2I_3}\otimes(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\right)\cdot\mathrm{vec}\left(\mathbf{X}^{(2)^T}\otimes\overline{\mathbf{X}}^{(3)^T}\right)\\
&=\left[\left(\mathbf{X}^{(1)^T}\cdot\mathbf{A}_{(1)}\right)\otimes\mathbf{A}_{(1)}\right]\cdot\left(\mathbf{I}_{I_2I_3}\otimes(\mathbf{X}^{(2)}\otimes\mathbf{X}^{(3)})\right)\cdot\\
&\qquad\mathbf{P}_{I_2I_3,R_2R_3}\cdot\left(\mathbf{I}_{R_2R_3}\otimes(\mathbf{X}^{(2)}\otimes\mathbf{I}_{I_3})\right)\cdot\mathbf{L}_{I_2,R_3,R_2}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right)\\
&=\mathbf{M}_{1,3b}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right).
\end{aligned}\tag{7.70}$$

By defining now the proper combinations of the above closed expressions,

$$\begin{aligned}
\mathbf{M}_{11}&=\mathbf{M}_{1,1a}-\mathbf{M}_{1,1b}\\
\mathbf{M}_{12}&=\mathbf{M}_{1,2a}+\mathbf{M}_{1,2b}\\
\mathbf{M}_{13}&=\mathbf{M}_{1,3a}+\mathbf{M}_{1,3b},
\end{aligned}\tag{7.71}$$

we obtain the first set of equations in the *vectorized* unknowns $\overline{\mathbf{X}}^{(1)},\overline{\mathbf{X}}^{(2)},\overline{\mathbf{X}}^{(3)}$

$$\mathbf{M}_{11}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right)+\mathbf{M}_{12}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(2)}\right)+\mathbf{M}_{13}\cdot\mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right)=T_1.\tag{7.72}$$

Due to the inherent symmetry, the two other analogous relations can be easily derived. This finally leads to the system

$$\begin{bmatrix}\mathbf{M}_{11}&\mathbf{M}_{12}&\mathbf{M}_{13}\\ \mathbf{M}_{21}&\mathbf{M}_{22}&\mathbf{M}_{23}\\ \mathbf{M}_{31}&\mathbf{M}_{32}&\mathbf{M}_{33}\end{bmatrix}\cdot\begin{bmatrix}\mathrm{vec}\left(\overline{\mathbf{X}}^{(1)}\right)\\ \mathrm{vec}\left(\overline{\mathbf{X}}^{(2)}\right)\\ \mathrm{vec}\left(\overline{\mathbf{X}}^{(3)}\right)\end{bmatrix}=\begin{bmatrix}T_1\\ T_2\\ T_3\end{bmatrix},\tag{7.73}$$

which contains the particular expressions of linear system (7.47):

$$\mathbf{M}_{\mathcal{A},\mathbf{X}}\overline{X} = B_{\mathcal{A},\mathbf{X}}.$$

For the particular implementation of the higher-order Grassmann-Rayleigh quotient iteration (HOGRQI) for the computation of the best rank-(R_1, R_2, R_3) approximation of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ we developed original Matlab code.

7.5 Example

In this section we give a concrete illlustration of a best-rank approximation to a tensor and point out its use and potential in different fields. Consider a tensor $\mathcal{A} \in \mathbb{R}^{3\times 2\times 2}$, defined by the following matrix unfolding:

$$\mathbf{A}_{(1)} = \left[\begin{array}{cc|cc} 0 & -1 & 1 & 4 \\ 2 & -2 & 3 & -5 \\ 4 & 3 & 5 & -6 \end{array}\right]$$

The left singular matrices and singular values of $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}, \mathbf{A}_{(3)}$ are given by

$$\mathbf{U}^{(1)} = \begin{bmatrix} -0.2465 & -0.4993 & 0.8306 \\ 0.5217 & 0.6539 & 0.5479 \\ 0.8167 & -0.5684 & -0.0994 \end{bmatrix}, \quad S^{(1)} = [11.0753, 35498, 3.2768],$$

$$\mathbf{U}^{(2)} = \begin{bmatrix} 0.1715 & 0.9852 \\ 0.9852 & -0.1715 \end{bmatrix}, \qquad S^{(2)} = [10.6975, 5.6181, 0]$$

$$\mathbf{U}^{(3)} = \begin{bmatrix} 0.5105 & 0.8599 \\ 0.8599 & 0.5105 \end{bmatrix}, \qquad S^{(3)} = [10.5162, 5.9506, 0].$$

Truncation of the HOSVD after the first term in each mode gives a rank-1 approximation of $\mathcal{A}$ of which the Frobenius norm equals 10.0470. This value is increased to $\lambda = 10.1693$ by our GRQI algorithm. The evolution of the norm during the iteration is given by the solid line in Figure 7.4. The best rank-(1,1,1) approximation $\lambda \times_1 X^{(1)} \times_2 X^{(2)} \times_3 X^{(3)}$ is given by

$$X^{(1)} = \begin{bmatrix} -0.2515 \\ 0.6035 \\ 0.7567 \end{bmatrix}, \qquad X^{(2)} = \begin{bmatrix} -0.5765 \\ 0.8171 \end{bmatrix}, \qquad X^{(3)} = \begin{bmatrix} 0.1344 \\ 0.9909 \end{bmatrix}.$$

For a best rank-(2,2,1) approximation of $\mathcal{A}$, a mere truncation of the HOSVD after the corresponding terms in each mode gives an approximation of which the Frobenius norm equals 10.2684. The Frobenius norm of our GRQI algorithm increases the norm to 10.5162. The evolution of the norm during the GRQI algorithm is given by the dashed line in Figure 7.4. The best rank-(2,2,1)

approximation of $\mathcal{A}$, decomposed as $\mathcal{B} \times_1 \mathbf{X}^{(1)} \times_2 \mathbf{X}^{(2)} \times_3 \mathbf{X}^{(3)}$ as in (7.28), is given by

$$\mathbf{X}^{(1)} = \begin{bmatrix} -0.3038 & -0.3962 \\ 0.5493 & -0.8159 \\ 0.7784 & 0.4211 \end{bmatrix}, \quad \mathcal{B} = \left[\begin{array}{cc|c} 10.1046 & -0.7110 \\ -0.7693 & -2.7182 \end{array}\right]$$

$$\mathbf{X}^{(2)} = \begin{bmatrix} -0.1513 & -0.9885 \\ -0.9885 & 0.1513 \end{bmatrix}, \qquad \mathbf{X}^{(3)} = \begin{bmatrix} -0.5105 \\ 0.8599 \end{bmatrix}.$$

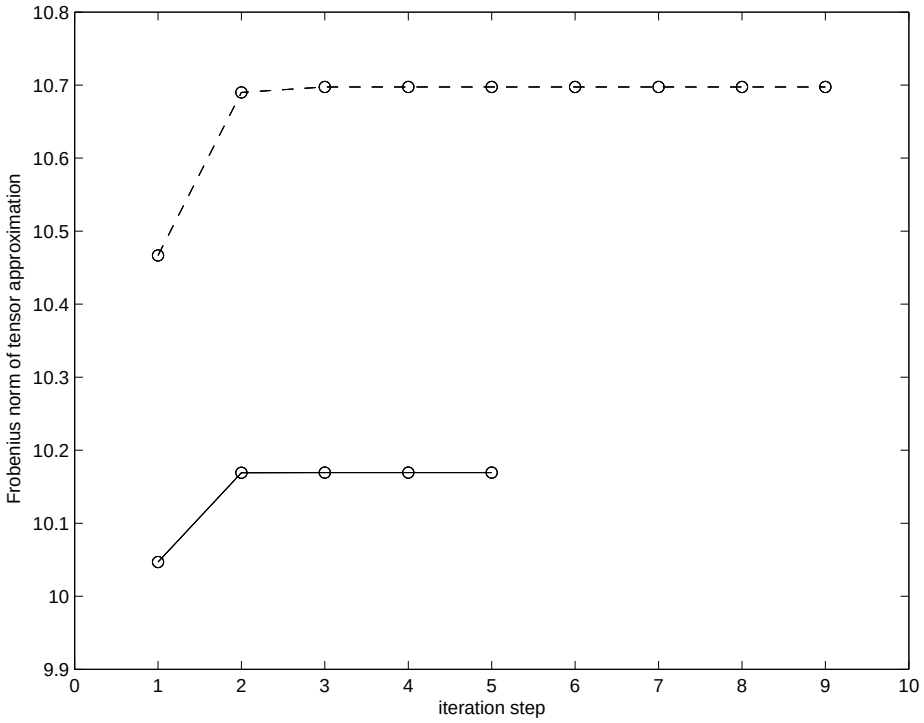


Figure 7.4: Evolution of the Frobenius norm of the approximation $\hat{\mathcal{A}}$ of a $(3 \times 2 \times 2)$ tensor $\mathcal{A}$, during the Generalized Rayleigh quotient Iteration, as a function of the iteration step. Solid line: computation of the best rank-(1,1,1) approximation. Dashed line: computation of the best rank-(2,2,1) approximation. Quick convergence to the optimal maximum value is attained through our GRQI algorithm with a quadratic convergence rate.

That the potential application domain of our algorithm is highly extensive, needs no comment since best-rank approximation is prevalent in almost any science and engineering field. For example in telecommunications one considers blind separation of signals having specific properties (non-Gaussian/not-circular/time-correlated/non-stationary). In psychometrics and chemometrics one requires sophisticated dimensionality reduction schemes of multiway

datasets or preprocessing for canonical decompositions. In the signal processing community one needs optimal dimensionality reduction in blind identification, cumulant slice selection in signal processing, variants of Canonical Correlation Analysis, etc.

An important example of a typical application is Independent Component Analysis (ICA). Many ICA applications involve high-dimensional data in which, however, only a few sources have significant contributions. Examples are nuclear magnetic resonance (NMR), electro-encephalography (EEG), magneto-encephalography (MEG), hyper-spectral image processing, data analysis, etc.

To reduce the computational complexity and to decrease the variance of the results, one may wish to reduce the dimensionality of the problem from the number of observation channels, which will be denoted by I, to the number of sources, denoted by R. If one wishes to avoid a classical prewhitening, for the reasons given in [199], then the solution can be obtained by means of a so-called best rank-(R_1, R_2, R_3) approximation of a higher-order tensor [43].

Several ICA-methods are based on the joint diagonalization of a set of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_J \in \mathbb{R}^{I \times I}$. In the absence of noise, these matrices satisfy

$$\mathbf{A}_j = \mathbf{M} \cdot \mathbf{D}_j \cdot \mathbf{M}^T, \qquad j = 1, \ldots, J$$

in which $\mathbf{M}$ is the mixing matrix and $\mathbf{D}_j \in \mathbb{R}^{R \times R}$ are diagonal. These matrices can be stacked in a tensor $\mathcal{A} \in \mathbb{R}^{I \times I \times J}$. Because the columns of all $\mathbf{A}_j$ are linear combinations of the columns of $\mathbf{M}$, the 1-mode vector space of $\mathcal{A}$ is the column space of $\mathbf{M}$ and its 1-mode rank equals R. Because of the symmetry, the 2-mode vector space also coincides with the column space of $\mathbf{M}$ and the 2-mode rank is also equal to R. It can be verified that the 3-mode vectors are linear combinations of the vectors $(\mathbf{D}_1(r,r), \ldots, \mathbf{D}_J(r,r))^T$, $r = 1, \ldots, R$. This is shown in detail in [44]. Hence the 3-mode rank is bounded by R.

A dimensionality reduction can thus be achieved by computing the best rank-(R, R, R) approximation of $\mathcal{A}$. A difference with section 7.3.1 is that now $\mathbf{X}^{(1)} = \mathbf{X}^{(2)}$, $\overline{\mathbf{X}}^{(1)} = \overline{\mathbf{X}}^{(2)}$, $\mathbf{R}_1(\mathbf{X}) = \mathbf{R}_2(\mathbf{X})$, because of the symmetry. On one hand, when $R < J$, this can simply be inserted in (7.46). Equation (7.47) then becomes a square set in $(I + J)R$ unknowns. On the other hand, when $R \geqslant J$, the computation can further be simplified. In this case, no dimensionality reduction in the third mode is needed, and $\mathbf{X}^{(3)}$ can be fixed to the identity matrix. Equation (7.46) reduces to

$$\begin{aligned} \overline{\mathbf{X}}^{(1)} \cdot \mathbf{R}_1(\mathbf{X}) \cdot \mathbf{R}_1(\mathbf{X})^T = \\ \mathbf{A}_{(1)} \cdot \left[(\mathbf{X}^{(1)} \cdot \mathbf{X}^{(1)^T}) \otimes \mathbf{I} \right] \cdot \mathbf{A}_{(1)}^T \cdot (\overline{\mathbf{X}}^{(1)} - 2\mathbf{X}^{(1)}) \\ + \mathbf{A}_{(1)} \cdot \left[(\overline{\mathbf{X}}^{(1)} \cdot \mathbf{X}^{(1)^T} + \mathbf{X}^{(1)^T} \cdot \overline{\mathbf{X}}^{(1)}) \otimes \mathbf{I} \right] \cdot \mathbf{A}_{(1)}^T \cdot \mathbf{X}^{(1)} \ . \end{aligned} \tag{7.74}$$

(Note that the factor 4 in (7.46) has been replaced by a factor 2, because two of the terms in (7.42) vanish.) Equation (7.47) now becomes a square set in

IR unknowns. This kind of tensors with symmetry along the third mode, such that $a_{ijk} = a_{jik}$, occurs frequently and therefore we implemented it separately. For an explicit demonstration of the usage of our algorithm in a specific ICA application we refer the reader to [44]. Currently our algorithm is finding continued applications in signal processing [131].

7.6 Conclusions

In this chapter we considered a topic in the field of multilinear algebra. In the area of dimensionality reduction we focused on the Rayleigh quotient iteration for obtaining an eigenvector to a matrix. We followed two connections in the literature that generalize the algorithm in different aspects. Absil *et al.* devise an algorithm that yields the dominant eigenspace of a matrix [2]. Zhang *et al.* proposed an algorithm to obtain the dominant rank-1 term of a tensor [200]. We have complemented their results with the most general case of computing the jointly dominant k-mode subspaces of a tensor.

In particular, we have derived a higher-order Grassmann-Rayleigh Quotient Iteration for the computation of the best rank-(R_1, R_2, R_3) approximation of higher-order tensors [41]. The convergence is quadratic and each iteration step merely involves solving a square set of linear equations. This is a big improvement over the algorithm discussed in [43], of which the convergence is at most linear and of which each iteration involves the partial computation of a number of SVDs. We provided analytical closed formulas and a Matlab implementation for the RQI algorithm, and illustrated its application potential in an Independent Component Analysis context.

Chapter 8

Conclusions

In this chapter we give a summarized overview of our contributions, the conclusions and we point out some topics for further research.

8.1 Summary

In **Chapter 1** we described in intuitive terms the general context, the basic research questions, the structure and main contributions of the thesis.

In **Chapter 2** we highlighted basic concepts of machine learning, creating a background for the next chapters with an overview of the paradigm on which kernel models are founded. Hereby we mainly followed the statistical learning point of view, the cornerstone of Vapnik's Support Vector Machines (SVM).

In the field of data mining and machine learning we focused on the subject of learning by examples. The concept of the utilized performance measure serves as the guideline towards the development of a theory of learning. The natural working hypothesis, empirical risk, is recognized as a practical inference principle that needs to be extended to structural risk in order to reach a properly behaving solution. Another aspect of performance is the capability of modeling nonlinearities, where kernels have been used within Support Vector Machines.

We discussed then how these two aspects are found to be naturally unified in the particular structure of Reproducing kernel Hilbert spaces. We gave an overview of the Support Vector Machines (SVM), that accelerated the proliferation of the kernel paradigm. Finally, we discussed approaches to determine the hyperparameters of kernel models through the concepts of model selection and assessment, which were mainly used throughout our experiments.

In **Chapter 3** we focused on extensions of the SVM and its methodology. Next to the class of Least Squares Support Vector Machines (LS-SVM), which regularizes by penalty of the norm of the parameter vector, we placed another class of penalized kernel models by considering the decomposition of least squares regression models in feature space with regularization by projection into a subspace. In this decomposition we recognized a natural and very general framework in which methods like Kernel Principal Component Analysis (KPCA), Kernel Canonical Correlation Analysis (KCCA) and Kernel Partial Least Squares (KPLS) (with variants) are subcases with an advantageous choice of subspace. We illustrated and compared the models on some data sets and demonstrated some applications with successful results, with equal performance compared to LS-SVM regression [83].

Many methods are called kernel machines once they are endowed with a kernel based dot product: in that way they are limited to the use of the so-called kernel trick. But it does not imply that they are motivated according to the principled primal-dual optimization approach of SVMs. One of the examples where such a derivation was provided with this conceptual interpretation is the Least Squares Support Vector Machine (LS-SVM). We additionally made such a derivation for the Kernel Partial Least Squares method (KPLS) [84].

The concept of SVM's to introduce nonlinearity through the use of kernels has in fact a very broad applicability, and its proliferation into other domains is often fruitful. We employed KCCA to generalize a method in the field of Subspace Identification to find the state in a class of nonlinear (instead of linear) Hammerstein-Wiener systems. We illustrated the method on some examples, showing that proper identification of the nonlinear state-space model is achieved [71].

In **Chapter 4** we further enhanced the least squares subspace regression framework of the previous chapter. We employed an approximation, via the Nyström approximation, of the subspaces computed on a reduced part of the training data set. In this manner we obtained a generalization of the Fixed-Size LS-SVM models: a subset based least squares subspace restricted regression in feature space [87].

These methods operate in primal space, which has two important advantages: a small number of regression coefficients (which allows a fast training) and a sparse kernel expansion (which allows fast evaluation). Precisely their combination enables to process large data sets. And in our framework these two advantages are distinct model choices, through (i) a restriction of the search space to a subspace and (ii) a particular choice of basis vectors in feature space. We showed that our model is not just a kernelization of a single model, but comprises a whole general class of least squares kernel models at once.

We illustrated different sparse restricted regression models by some results on an artificial and a three real-world data sets, including a large scale data set,

demonstrating that a sparse restricted regression model is effectively capable of dealing with a large dataset [86]. We further demonstrated the application of our class of models in a comparative study on the Silver Box benchmark data, with excellent results [54].

In **Chapter 5** we explore possible pathways to introduce also, like in previous chapter done for subspace kernel regression, sparsity in the LS-SVM model. It is a highly desirable characteristic of SVM to have the final solution only dependent on a subset of the training data, because it yields the method fast in evaluation at new data.

First we performed a study of four pruning capable methods and proposed an alternative selection criterion for one of the methods. We then experimentally verified the performances on a battery of UCI benchmark data sets, and formulated practical guidelines as which method to use [85].

Secondly, in contrast to pruning, we considered also growing, and thus regarded the LS-SVM in an online setting. Different approaches of updating were considered, with corresponding errors in data and feature space. Also downdate errors in data and feature space, different approaches of downdating and their -naturally- induced selection criteria were distinguished for the LS-SVM case and structured in a clear taxonomy. Finally, we illustrated and compared some useful combinations of the methods.

In **Chapter 6** we elaborated also an updating and downdating procedure, but now for the computation of the dominant principal components of the kernel matrix. The dominant eigenspace of the kernel matrix is of central importance in many kernel methods, yet in the case of dynamic and/or large scale data, the batch calculation nature and computational demands of the eigenvector decomposition limit these methods in numerous applications.

We proposed an efficient and fast incremental alternative updating/downdating approach, especially suited for the kernel context induced peculiarity that the adaptation simultaneously occurs both in the rows and columns. We additionally developed an efficient way to downsize the row dimension of an updated eigenspace. In this way we achieved a true on-line algorithm for eigenspace tracking, with a time complexity of $O(nm^2)$ and memory requirements of $O(nm)$ per iteration.

We provided a Matlab implementation and presented extensive experiments on some typical benchmark data sets to demonstrate that accuracy and stability of the eigenvalues and eigenvectors are well-preserved during and after iteration, in comparison to the classical batch computed solution [82, 81].

In **Chapter 7** we connected to the previous chapter with the computation of the dominant eigenspace of a generalization of matrices, namely higher-order tensors. For the computation we focused on the Rayleigh Quotient Iteration (RQI). This algorithm for obtaining an eigenvector of a matrix has been long

known and is appreciated for its favorable convergence properties.

Complementing two existing generalizations of the RQI algorithm, (i) that computes the dominant eigen*space* of a matrix and (ii) that computes the dominant eigenvector of a *tensor*, we developed a generalization of the Rayleigh Quotient for obtaining the dominant k-mode eigenspaces of tensors.

In other words, we have derived a higher-order Grassmann-Rayleigh Quotient Iteration for the computation of the best rank-(R_1, R_2, R_3) approximation of higher-order tensors [41]. The convergence is quadratic and each iteration step merely involves solving a square set of linear equations, which is a big improvement over alternative algorithms. We provided analytical closed formulas and a Matlab implementation for the RQI algorithm, and illustrated its application potential.

Support Vector Machines and Statistical Learning Theory have set the scene for a myriad of new questions and extensions, that provided a fertile breeding-ground for research. In this thesis we investigated mainly extensions from the perspective of the class of LS-SVMs. We provided a primal-dual interpretation for the class of kernel partial least Squares, distinguished a natural subspace regression framework in Reproducing Kernel Hilbert space with enhancements for dealing with large data sets, improved a method for obtaining a sparse LS-SVM, efficiently computed the dominant eigenspace of the kernel matrix and derived a Rayleigh Quotient iteration for obtaining a best rank-(R_1, R_2, R_3) approximation of higher-order tensors. In our subjects we have added empirically obtained results to illustrate the potential of our contributions.

8.2 Further research

Research never ends, in this sense this thesis is merely a snapshot of work in a limited time. There are certainly additional leads and challenges that need further investigation. We sum up a few possibilities that could be undertaken.

Choosing a right basis for the projection of the representers influences the quality of the kernel regression model. This forms an apart subject on itself where techniques from projection pursuit could be implemented, adapted or enhanced to the nonlinear kernel context. It would be a great advantage if the subspace choice could be made completely data-driven. Of course the choice of the kernel is an obvious topic that should be dealt with first and most of all, but it is a difficult problem.

When we added sparsity to subspace kernel regression, we used a random subsample selection for the data points of the reduced set. Other subsampling selection mechanisms could be motivated that would render the subspace choice or model performance more optimal (or robust to outliers).

In this context of sparsity, a taxonomy of the extension of LS-SVM to an online setting had been given with some preliminary experiments. A full empirical treatment, including for example non-stationary data sets with multiple sweeping and stability issues is an entire subject on itself.

At the moment kernels are penetrating the static modelling field, while the combination with more complex dynamical models is largely circumvented. Within our research group the first steps have been undertaken to introduce the use of kernels into other domains like Subspace Identification. The peculiar way of the implicit kernel map that introduces nonlinear capabilities for algorithms is a powerful instrument that has not yet been fully exploited.

Concerning the updating and tracking of the kernel principal components of the kernel matrix, we devised a downsizing algorithm to keep the row dimension of the eigenspace fixed. In this step we recover successfully the orthogonality, but nevertheless throw away information. Alternatives that minimize this would render the algorithm even more accurate and would probably decrease the need of occasional re-orthogonalizations. Also an extension to compute the eigenspace corresponding to the smallest eigenvalues would add value to the general applicability of our method.

Finally, our generalized definition of the Rayleigh Quotient could be adapted to a form that has the normalization included (instead of working on the Grassmann manifold). The extensions, reformulation of the algorithm and computational details must then be revisited. Also the computation of particular eigenvectors/spaces (in contrast to the dominant eigenspace) is an open research topic.

Bibliography

[1] P. Absil. *Invariant Subspace Computation: A geometric approach*. PhD thesis, University of Liege, Applied Sciences, Department of Electricity, Electronics and Informatics, institute of Montefiore, 2003.

[2] P. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren. A Grassmann-Rayleigh Quotient Iteration for computing invariant subspaces. *SIAM review*, 44(1):57–73, 2002.

[3] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] H. Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203–217, 1973.

[5] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 686:337–404, 1950.

[6] F.R. Bach and M.I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[7] R. Badeau, G. Richard, and B. David. Sliding window adaptive SVD algorithms. *IEEE Transactions on Signal Processing*, 52(1):1–10, january 2004.

[8] E.W. Bai. An optimal two-stage identification algorithm for Hammerstein-Wiener nonlinear systems. *Automatica*, 4(3):333–338, 1998.

[9] E.W. Bai. A blind approach to the Hammerstein-Wiener model identification. *Automatica*, 38:967–979, 2002.

[10] C.T.H. Baker. *The numerical treatment of integral equations*. Oxford Clarendon Press, 1977.

[11] M. Barker and W. Rayens. Partial least squares for discrimination. *Journal of Chemometrics*, 17:166–173, 2003.

[12] M. Basseville and I. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, Inc., April 1993.

[13] G. Baudat and F. Anouar. Kernel based methods and function approximation. In *IJCN*, pages 1244–1249, Washington DC, July 2001.

[14] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, 1995.

[15] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[16] M. Borga. *Learning Multidimensional Signal Processing*. PhD thesis, Department of Electrical Engineering, Linkoping University, 1998.

[17] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[18] O. Bousquet and A. Eliseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, March 2002.

[19] J. R. Bunch and J. P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, 31:111–129, 1978.

[20] P. Businger. Updating a singular value decomposition. *BIT*, 10:376–385, 1970.

[21] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS 2000)*, volume 13, pages 409–415, Cambridge, MA, 2001. MIT Press.

[22] G. C. Cawley and N. L. C. Talbot. Improved sparse least-squares support vector machines. *Neurocomputing*, 48:1025–1031, October 2002.

[23] G. C. Cawley and N. L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17:1467–1475, (in press) 2004.

[24] G.C. Cawley and N.L.C. Talbot. Efficient formation of a basis in a kernel induced feature space, April 2002.

[25] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59(5):321–332, 1997.

[26] F.H.I. Chang and R. Luus. A noniterative method for identification using the Hammerstein model. *IEEE Transactions on Automatic Control*, 16:464–468, 1971.

[27] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, 2:302–309, March 1991.

[28] H-M. Chi and O.K. Ersoy. Recursive update algorithm for least squares support vector machines. *Neural Processing Letters*, 17(2):165–173, April 2003.

[29] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[30] T. M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[31] D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Ann. Stat.*, 18:1676–1695, 1990.

[32] P. Crama. *Identification of block-oriented nonlinear models*. PhD thesis, Vrije Universiteit Brussel, Dept. ELEC, June 2004.

[33] P. Crama and J. Schoukens. Hammerstein-Wiener system estimator initialization. In *Proc. of the International Conference on Noise and Vibration Engineering (ISMA2002), Leuven*, pages 1169–1176, 16-18 September 2002.

[34] F. Crandall. Iterative procedures related to relaxation methods for eigenvalue problems. *Proc. Roy. Soc. London*, 207(1090):416–423, 1951.

[35] N. Cristianini and J. shawe Taylor. *Support vector machines*. Cambridge university press, 2000.

[36] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Comput.*, 14(3):641–668, 2002.

[37] F. Cucker and S. Smale. On the mathematical foundations of learning theory. *Bulletin of the AMS*, 39:1–49, 2002.

[38] S. de Jong and A. Phatak. Partial least squares regression. In S. Van Huffel, editor, *Recent advances in Total Least Squares Techniques and Errors-in-Variables modeling*, pages 311–338. SIAM, Philadelphia, 1997.

[39] B. J. de Kruif and T. J. A. de Vries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14(3):696–702, May 2003.

[40] L. De Lathauwer. *Signal Processing Based on Multilinear Algebra*. PhD thesis, K.U.Leuven, E.E. Dept., September 1997. 256 p.

[41] L. De Lathauwer, L. Hoegaerts, and J. Vandewalle. A Grassmann-Rayleigh Quotient Iteration for dimensionality reduction in ICA. In *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, Granada, Spain, September 2004.

[42] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, April 2000.

[43] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,...,rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, April 2000.

[44] L. De Lathauwer and J. Vandewalle. Dimensionality reduction in ICA and Rank-$(r_1, r_2, \ldots, r_n)$ reduction in multilinear algebra. In *Proc. of the Fifth International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, pages 295–302, Granada, Spain, September 2004.

[45] J. de Leeuw. Nonlinear principal component analysis. In *Compstat proceedings in computational statistics*, Physica, pages 77–86, Vienna, 1982.

[46] B. De Moor. *Mathematical concepts and techniques for modeling of static and dynamic systems*. PhD thesis, Katholieke universiteit Leuven, K.U.Leuven (Leuven, Belgium), 1988.

[47] B. De Moor, T. Van Gestel, L. Hoegaerts, K. Pelckmans, and I. Goethals. Modelling, scenario and risk analysis of peak loads in the belgian high-voltage electricity network. Technical report, KULeuven Electrical Engineering (ESAT-SCD/SISTA), ELIA NV (Belgian System Operator), Leuven, Belgium, 2002.

[48] B. De Moor, P. Van Overschee, L. Hoegaerts, K. Pelckmans, and O. Barrero. Softsensors for the chemical process industry. Technical report, KULeuven Electrical Engineering (ESAT-SCD/SISTA), IPCOS-ISMC NV, BASF Antwerpen NV, Leuven, Belgium, 2001.

[49] K.I. Diamantras and S.Y. Kung. *Principal component neural networks*. John Wiley and Sons, NY, 1996.

[50] T.J. Dodd, V. Kadirkamanathan, and R.F. Harrison. Function estimation in Hilbert space using sequential projections. In *Proceedings of the IFAC Conference on Intelligent Control Systems and Signal Processing (ICONS 2003)*, pages 113–118, University of Algarve, Portugal, 2003.

[51] T.J. Dodd, B. Mitchinson, and R.F. Harrison. Sparse stochastic gradient descent learning in kernel models. In *Proceedings of the Second*

International Conference on Computational Intelligence, Robotics and Autonomous Systems (Singapore), 2003.

[52] A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20:303–353, 1998.

[53] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, London, 1993.

[54] M. Espinoza, K. Pelckmans, L. Hoegaerts, J.A.K. Suykens, and B. De Moor. A comparative study of LS-SVMs applied to the silver box identification problem. In *Proceedings of the 6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004)*, Stuttgart, Germany, September 2004.

[55] T. Evgeniou. *Learning with kernel machine architectures*. PhD thesis, MIT, 2000.

[56] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[57] A.H. Falkner. Iterative technique in the identification of a non-linear system. *International Journal of Control*, 1:385–396, 48.

[58] W. Favoreel, B. De Moor, and P. Van Overschee. Subspace identification of bilinear systems subject to white inputs. *IEEE Transactions on Automatic Control*, 44(6):1157–1165, 1999.

[59] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in knowledge discovery and data mining*. AAAI Press/MIT Press, 1996.

[60] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(2):243–264, 2002.

[61] R. A. Fisher. *Contributions to Mathematical Statistics*. John Wiley & Sons, New York., 1950.

[62] I. Frank and J. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35:109–148, 1993.

[63] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computing*, 23:881–890, 1974.

[64] P. Geladi and B.R. Kowalski. Partial least squares regression: a tutorial. *Analytica chimica*, pages 185:1–17, 1986.

[65] A. Gifi. *Nonlinear multivariate Analysis*. Chichester, John Wiley and Sons, 1990.

[66] M. Girolami. Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Computation*, 14(3):669–688, 2002.

[67] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[68] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[69] R. Gittins. *Canonical analysis*, volume 12. Springer-Verlag, biomathematics edition, 1985.

[70] I. Goethals. *Subspace Identification for linear, Hammerstein and Hammerstein-Wiener systems*. PhD thesis, KULeuven, Applied Sciences, Electrical Engineering, Belgium, Leuven, May 2005.

[71] I. Goethals, L. Hoegaerts, V. Verdult, Suykens J.A.K., and De Moor B. Hammerstein-wiener subspace identification using kernel canonical correlation. *Submitted for publication.*, 2005.

[72] I. Goethals, K. Pelckmans, J.A.K. Suykens, and B. De Moor. Subspace identification of hammerstein systems using least squares support vector machines. Technical Report 04-114, ESAT-SISTA, K.U.Leuven (Leuven Belgium), *Submitted for publication*, 2004.

[73] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland 21218, 2nd edition, 1989.

[74] I. Guyon, I. Poujaud, L. Personnaz, and G. Dreyfus. Comparing different neural network architectures for classifying handwritten digits. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-89)*, volume II, pages 127–132, Washington, D.C, 1989. IEEE Press, New York.

[75] Z.F. Hao, S. Yu, X.W Yang, F. Zhao, R. Hu, and Y.C. Liang. Online LS-SVM learning for classification problems based on incremental chunk. In *Advances in Neural Networks, Lecture Notes in Computer Science*, volume 3173, pages 558–564. Springer-Verlag, Berlin, Germany, 2004.

[76] W. Hardle and J. Marron. Optimal bandwidth selection in nonparametric regression function estimation. *Ann. Statist.*, 13:1465–1481, 1985.

[77] D. Harrison and D.L. Rubinfeld. Hedonic prices and the demand for clean air. *J. Environ. Economics & Management*, 5:81–102, 1978.

[78] B. Hassibi, D. G. Stork, and G. Wolf. Optimal brain surgeon and general network pruning. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages 293–300, San Francisco, CA, Apr 1993.

[79] R. Hastie, T.and Tibshirani and J. Friedman. *The elements of statistical learning*. Springer Series in Statistics, New York, 2001.

[80] I. Helland. On the structure of partial least squares regression. *Communications in Statistics- Simulation and Computations*, 17:581–607, 1988.

[81] L. Hoegaerts, L. De Lathauwer, I. Goethals, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Efficiently updating and tracking the dominant kernel principal components. *Submitted for publication*, 2005.

[82] L. Hoegaerts, L. De Lathauwer, J.A.K. Suykens, and J. Vandewalle. Efficiently updating and tracking the dominant kernel eigenspace. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2004)*, Leuven, Belgium, July 2004.

[83] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Kernel PLS variants for regression. In *Proc. of the 11th European Symposium on Artificial Neural Networks (ESANN2003)*, pages 203–208, Bruges, Belgium, April 2003.

[84] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subspace regression in reproducing kernel hilbert space. In *Proc. of the 13th System Identification Symposium (SYSID2003)*, pages 839–842, Rotterdam, Nederland, August 2003.

[85] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. A comparison of pruning algorithms for sparse least squares support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing (ICONIP 2004)*, Calcutta, India, November 2004.

[86] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Primal space sparse kernel partial least squares regression for large scale problems. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2004)*, Budapest, Hungary, July 2004.

[87] L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subset based least squares subspace regression in RKHS. *Neurocomputing*, 63:293–323, January 2005.

[88] A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.

[89] A. Hoskuldsson. Pls regression methods. *Journal of Chemometrics*, 2(3):211–228, June 1988.

[90] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[91] H. Hotelling. The most predictable criterion. *Journal of Educational Psychology*, 26:139–142, 1935.

[92] H. Hotelling. Relation between two sets of variates. *Biometrika*, 28:321–377, 1936.

[93] A. S. Householder. *Principles of Numerical Analysis*. McGraw-Hill, Dover, New York, 1974.

[94] I.C.F. Ipsen. Computing an eigenvector with inverse iteration. *IAM rev.*, 39(2):254–291, 1997.

[95] V.V. Ivanov. On linear problems which are not well-posed. *Soviet. Math. Dokl.*, 3(4):981–983, 1962.

[96] Ma J., J. Theiler, and S. Perkins. Accurate on-line support vector regression. *Neural Comput.*, 15(11):2683–2703, 2003.

[97] A. M. Jade, B. Srikanth, V.K. Jayaraman, B.D. Kulkarni, J.P. Jog, and L. Priya. Feature extraction and denoising using kernel PCA. *Chemical Engineering Science*, 58(19):4441–4448, October 2003.

[98] Burges C. J.C. A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2(2):121–167, 1998.

[99] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[100] B.J. Kim, I.K. Kim, and K.B. Kim. Feature extraction and classification system for nonlinear and online data. In *Adv. in knowl. disc. and data mining, proceedings Lecture notes in Art. Int.*, pages 171–180. Springer-Verlag, Berlin, Germany, 2004.

[101] K. In Kim, M.O. Franz, and B. Schölkopf. Kernel Hebbian algorithm for iterative kernel principal component analysis. Technical Report 109, Max-Planck-Institut für biologische Kybernetik, Tübingen, June 2003.

[102] G.S. Kimeldorf and G. Wahba. Tchebycheffian spline functions. *J. Math. Ana. Applic.*, 33:82–95, 1971.

[103] J. Kivinen, A.J. Smola, and R.C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, August 2004.

[104] A. Kuh. Adaptive kernel methods for CDMA systems. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2001)*, pages 1404–1409, Washington DC, 2001.

[105] P.L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(5):365–377, 2000.

[106] W.E. Larimore. Canonical variate analysis in identification, filtering and adaptive control. In *Proceedings of the 26th Conference on Decision and Control (CDC90), Hawaii, US*, pages 594–604, 1990.

[107] Y. Lee and O. L. Mangasarian. RSVM: Reduced support vector machines. First SIAM International Conference on Data Mining, Chicago, April 2001. Technical Report 00-07, July, 2000.

[108] A. Levy and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction. *IEEE Transactions on Image processing*, 9(8):1371–1374, 2000.

[109] K.-M. Lin and C.-J. Lin. A study on reduced support vector machines. *Neural Networks, IEEE Transactions on*, 14(6):1449 – 1459, Nov 2003.

[110] J.H. Liu, J.P. Chan, S. Jiang, and J.S. Cheng. Online ls-svm for function estimation and classification. *Journal of the university of science and technology Beijing*, 10(5):73–77, October 2003.

[111] L. Ljung. *System Identification, Theory for the User*. Prentice Hall, 1987.

[112] A. Lorber, L. Wangen, and B. Kowalski. A theoretical foundation for the pls algorithm. *Journal of Chemometrics*, 1:19–31, 1987.

[113] Bernardo J. M. and Smith A. F.M. *Bayesian Theory*. John Wiley & Sons, New York., 1994.

[114] D.J.C. Mackay. Probable networks and plausible predictions, a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6:469–505, 1995.

[115] C.L. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

[116] O.L. Mangasarian and D.R. Musicant. Active set support vector machine classification. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 577–583. MIT Press, Cambridge, MA, 2001.

[117] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, London, San Diego, 1979.

[118] H. Martens, S. Wold, and M. Martens. *Food Research and Data Analysis*, chapter Laymans guide to multivariate data analysis, pages 473–492. Applied Science Publications, London, 1983.

[119] W.F. Massy. Principal component regression in exploratory statistical research. *Journal of the American Statistical Association*, 60:234–246, 1965.

[120] S. Mika, B. Schölkopf, A. Smola, K. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. *Advances in Neural Information Processing Systems*, 11:536 – 542, 1999.

[121] A. J. Miller. *Subset Selection in Regression*. Chapman & Hall, 1990.

[122] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[123] T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

[124] R. R. Mohler. *Nonlinear systems, Volume II: Applications to Bilinear Control*. Englewood-Cliffs, New Jersey: Prentice-Hall, 1991.

[125] M. Momma and K. P. Bennett. Sparse kernel partial least squares regression. In *Proceedings of Conference on Learning Theory (COLT 2003)*, pages 216–230, 2003.

[126] M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle. On- and off-line identification of linear state-space models. *International Journal of Control*, 49:219–232, Jan. 1989.

[127] M. Moonen, P. Van Dooren, and J. Vandewalle. An SVD updating algorithm for subspace tracking. *SIAM J. Matrix Anal. Appl.*, 13(4):1015–1038, 1992.

[128] C.L. Muntz. Solution direct de l'equation seculaire et de quelques problems analogues. *Comptus Rendus de l'Academie des Sciences*, 156:43–46, 1913.

[129] R.M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, Graduate Dept. of Computer Science, Univ. of Toronto, March 1995.

[130] E. J. Nyström. Uber die praktische auflosung von linearen integralgleichungen mit anwendungen auf randwertaufgaben der potentialtheorie. *Commentationes PhysicoMathematica*, 4(15):1–52, 1928.

[131] J.M. Papy, L. De Lathauwer, and S. Van Huffel. Exponential data fitting using multilinear algebra, the single-channel and multi-channel case. *Numerical Linear Algebra with Applications (In press.)*, 2005.

[132] B.N. Parlett. *The symmetric eigenvalue problem*, volume 20. SIAM Classics Appl. Math., Philadelphia, 1998.

[133] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine and journal of science*, 2(11):559–572, November 1901.

[134] K. Pelckmans. *Primal-dual Kernel Machines*. PhD thesis, KULeuven, Applied Sciences, Electrical Engineering, Belgium, Leuven, May 2005.

[135] K. Pelckmans, I. Goethals, J. De Brabanter, J.A.K. Suykens, and B. De Moor. Componentwise least squares support vector machines. *Support Vector Machines: Theory and Applications,* L. Wang (Ed.), Springer, 2005, *In press*.

[136] D.L. Philips. A technique for the numerical solution of certain integral equations of the first kind. *J. Assoc. Comp. Mach.*, 9:84–97, 1962.

[137] J. Platt. Flat training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.

[138] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78(9):1481–1497, September 1990.

[139] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 1st edition, 1986.

[140] F. Ramsey and D. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis*. Wadsworth Publishing Company, Belmont, CA, 1997.

[141] W. S. Rayens. The art of maximizing covariance. Technical Report 383, Kentucky university, dept. of statistics, June 2000.

[142] R. Reed. Pruning algorithms – a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

[143] R. Rifkin, G. Yeo, and T. Poggio. *Regularized Least Squares Classification. Advances in Learning Theory: Methods, Models and Applications, Eds. Suykens, Horvath, Basu, Micchelli, and Vandewalle*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*. IOS Press, Amsterdam, 2003.

[144] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[145] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, New York, 1962.

[146] R. Rosipal and M. Girolami. An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, 13(3):505–510, 2001.

[147] R. Rosipal, M. Girolami, L.J. Trejo, and A. Cichocki. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10(3):231–243, 2001.

[148] R. Rosipal and L. J. Trejo. Kernel partial least squares regression in reproducing kernel Hilbert space. *Journal of Machine Learning Research*, 2:97–123, Dec 2001.

[149] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by backpropagation errors. *Nature*, 323:533–536, 1986.

[150] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine learning (ICML'98)*, pages 515–521. Morgan Kaufmann, 1998.

[151] B. Schölkopf, R. Herbrich, A. Smola, and R. Williamson. A generalized representer theorem. Technical report, Neurocolt2 technical report series, May 2000.

[152] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Mller, G. Rätsch, and A.J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions On Neural Networks*, 10(5):1000–1017, September 1999.

[153] B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, and K.-R. Mller. Kernel PCA pattern reconstruction via approximate pre-images. In M. Boden L. Niklasson and T. Ziemke, editors, *Perspectives in Neural Computing*, pages 147–152, Berlin, Germany, 1998. Springer.

[154] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[155] B. Schölkopf, A.J. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[156] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classiers. *IEEE Trans. on Signal Processing*, 45(11):2758–2765, 1997.

[157] B. Scholkopf, R.C. Williamson, A.J. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for method for novelty detection. In *Proceedings of the 1999 Conference on Advances in Neural Information Processing Systems*, volume 12, 2000.

[158] J. Schoukens, J. G. Nemeth, P. Crama, Y. Rolain, and R. Pintelon. Fast approximate identification of nonlinear systems. *Automatica*, 39:1267–1274, 2003.

[159] G. Schwartz. Estimating the dimension of a model. *Ann. of Statist.*, 6:461–464, 1979.

[160] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

[161] M. Stone. Cross-validatory choice and assessment of statistical predictions. *J. Royal Statist. Soc.*, B(36):111–147, 1974.

[162] M. Stone and R. Brooks. Continuum regression: cross-validated sequentially constructed pre- diction embracing ordinary least squares, partial least squares and principal components regression. *Journal of the Royal Statistical Society*, B(52):237–269, 1990.

[163] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines : robustness and sparse approximation. *Neurocomputing, Special issue on fundamental and information processing aspects of neurocomputing*, 48(1-4):85–105, Oct 2002.

[164] J. A. K. Suykens, L. Lukas, and J. Vandewalle. Sparse approximation using least squares support vector machines. In *IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, pages II757–II760, Geneva, Switzerland, May 2000.

[165] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[166] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor. A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, 14(2):447–450, March 2003.

[167] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, june 1999.

[168] J. A. K. Suykens and J. Vandewalle. Multiclass least squares support vector machines. In *Proc. International Joint Conference on Neural Networks (IJCNN'99)*, Washington DC, 1999.

[169] N. Syed, H. Liu, and K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-1999)*, San Diego, CA, USA, 1999. ACM Press NY.

[170] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet. Math. Dokl.*, 5:1035–1038, 1963.

[171] A.N. Tikhonov and V.Y. Arsenin. *Solution of Ill-Posed Problems*. Winston, Washington DC, 1977.

[172] V. Tresp. Scaling kernel-based systems to large data sets. *Data Mining and Knowledge Discovery*, 5(3):197–211, 2001.

[173] L.R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[174] C. J. Twining and C. J. Taylor. The use of kernel principal component analysis to model data distributions. *Pattern Recognition*, 36(1):217–227, January 2003.

[175] T. Van Gestel. *From Linear to Kernel Based Methods in Classification, Modelling and Prediction.* PhD thesis, K.U. Leuven Department of Electrical Engineering, 2002.

[176] T. Van Gestel, J. A. K. Suykens, J. De Brabanter, B. De Moor, and J. Vandewalle. Kernel canonical correlation analysis and least squares support vector machines. In *Proc. of the International Conference on Artificial Neureal Networks (ICANN 2001)*, pages 381–386, Vienna, Austria, August 2001.

[177] T. Van Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Multiclass ls-svms: Moderated outputs and coding-decoding schemes. *Neural Process. Lett.*, 15(1):45–58, 2002.

[178] P. Van Overschee and B. De Moor. Choice of state-space basis in combined deterministic-stochastic subspace identification. *Automatica,* Special Issue on Trends in System Identification, 31(12):1877–1883, 1995.

[179] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications.* Kluwer Academic Publishers, 1996.

[180] V. Vapnik and A. Chervonenkis. Theory of pattern recognition. *Nauka*, 1974. German translation: Theorie der Zeichenerkennung, Akademie-Verlag, 1974, Berlin, 1979.

[181] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[182] V.N. Vapnik. Estimation of dependences based on empirical data. *Nauka*, 1979. English translation, Springer-Verlag, 1982.

[183] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Berlin: Springer-Verlag, 1995.

[184] V.N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[185] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–281, 1971.

[186] V. Verdult, J.A.K. Suykens, J. Boets, I. Goethals, and B. De Moor. Least squares support vector machines for kernel cca in nonlinear state-space identification. In *Proc. of the 16th international symposium on Mathematical Theory of Networks and Systems (MTNS2004), Leuven, Belgium*, July 2004.

[187] V. Verdult and M. Verhaegen. Identification of multivariable bilinear state space systems based on subspace techniques and separable least squares optimization. *International Journal of Control*, 74(18):1824–1836, 2001.

[188] M. Verhaegen and P. Dewilde. Subspace model identification, part II: analysis of the elementary output-error state space model identification algorithm. *International Journal of Control*, 56:1211–1241, 1992.

[189] M. Verhaegen and D. Westwick. Identifying MIMO Hammerstein systems in the context of subspace model identification methods. *International Journal of Control*, 63:331–349, 1996.

[190] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1990.

[191] D.S. Watkins. Understanding the QR algorithm. *SIAM review*, 24:427–440, 1982.

[192] D. Westwick and M. Verhaegen. Identifying MIMO wiener systems using subspace model identification methods. *Signal Processing*, 52(2):235–258, 1996.

[193] J.H. Wilkinson. *The algebraic eigenvalue problem*. Clarendon Press, Oxford, 1965.

[194] C.K.I. Williams, C.E. Rasmussen, A. Schwaighofer, and V. Tresp. Observations on the Nyström method for Gaussian processes. Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, University of Edinburgh, 2002.

[195] C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems, eds. T. K. Leen, T. G. Diettrich, V. Tresp*, volume 13, pages 682–688. MIT Press, 2001.

[196] H. Wold. Estimation of principal components and related models by iterative least squares. In *Multivariate Analysis, ed. P.R. Krishnaiah*, pages 391–420. Academic Press, New York, 1966.

[197] H. Wold. Path models with latent variables: the nipals approach. In H. M. Balock, editor, *Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building*, pages 307 – 357. Academic Press, NY, 1975.

[198] H. Wold. Soft modelling: Intermediate between traditional model building and data analysis. *Mathematical Statistics*, 6:333–346, 1980.

[199] A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Trans. Signal Processing*, 50:1545–1553, 2002.

[200] T. Zhang and G. H. Golub. Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications. Society for Industrial and Applied Mathematics*, 23(2):534–550, 2001.

[201] Y. Zhu. Estimation of an N-L-N Hammerstein-Wiener model. *Automatica*, 38:1607–1614, 2002.

Curriculum Vitae

- Luc Hoegaerts was born January 23, 1975, in Bonheiden, Belgium.
- In 1993 he completed secondary school (mathematics/modern languages) at Herentals College.
- In 1998 he received the Master of Science degree in Physics (magna cum laude) from the KULeuven. His specialization was Nuclear Physics and his master thesis was devoted to the study of *Magnetism in Fe/Cr multilayers by Perturbed Angular Correlation Spectroscopy*, at the Institute of Nuclear and Radiation Physics (IKS) under the supervision of Prof. dr. Michel Rots.
- In 1998 he additionally obtained a Teacher's Degree in Science at KULeuven.
- In 1999 he obtained a Postacademic Degree in Informatics (cum laude) at KULeuven.
- In 2000 he worked as a System Engineer at software firm Ockham Technology NV (a spin-off of the Bekaert Group) in Mechelen.
- From 2001 he received 4 years funding from the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT) to investigate on 'Data Mining and Knowledge Extraction with Neural Networks'. This PhD research was conducted under supervision of his promotors Prof. dr. ir. Joos Vandewalle and Prof. dr. ir. Johan Suykens, at the KULeuven in the faculty of Appplied Sciences, Department of Electrical Engineering, in the *Laboratory for Signals, Identification, System theory and Automation* (SISTA/SCD) in the Data Mining group. He was also co-associated in 2 industrial projects with BASF NV and ELIA NV.
- From 2005 he works at Fortis Bank as a Modeller in Central Risk Management, in Brussels.

Publications

This research was supported by the Institute for the promotion of Innovation by Science and Technology in Flanders (IWT-Flanders), Bischoffsheimlaan 25, B-1000 Brussels.

- B. De Moor, P. Van Overschee, L. Hoegaerts, K. Pelckmans, and O. Barrero. Softsensors for the chemical process industry. *Technical report.* Research project: KULeuven Electrical Engineering (ESAT-SCD/SISTA), IPCOS-ISMC NV, BASF Antwerpen NV, Leuven, Belgium, 2001.

- B. De Moor, T. Van Gestel, L. Hoegaerts, K. Pelckmans, and I. Goethals. Modelling, scenario and risk analysis of peak loads in the belgian high-voltage electricity network. *Technical report.* Research Project: KULeuven Electrical Engineering (ESAT-SCD/SISTA), ELIA NV (Belgian System Operator), Leuven, Belgium, 2002.

- L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Kernel PLS variants for regression. In *Proceedings of the 11th European Symposium on Artificial Neural Networks (ESANN2003)*, pp. 203–208, Bruges, Belgium, April 2003.

- L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subspace regression in reproducing kernel Hilbert space. In *Proceedings of the 13th System Identification Symposium (SYSID2003)*, pp. 839–842, Rotterdam, Nederland, August 2003.

- L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Subset based least squares subspace regression in RKHS. *Neurocomputing, Special issue*, vol. 63, January 2005, pp. 293-323.

- L. Hoegaerts, L. De Lathauwer, J.A.K. Suykens, and J. Vandewalle. Efficiently updating and tracking the dominant kernel eigenspace. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2004)*, Leuven, Belgium, July 2004.

- L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor. Primal space sparse kernel partial least squares regression for large scale problems. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2004)*, Budapest, Hungary, July 2004.

- L. Hoegaerts, J.A.K. Suykens, J. Vandewalle, and B. De Moor. A comparison of pruning algorithms for sparse least squares support vector machines. In *Proceedings of the 11th International Conference on Neural Information Processing (ICONIP 2004)*, Calcutta, India, November 2004.

- M. Espinoza, K. Pelckmans, L. Hoegaerts, J.A.K. Suykens, and B. De Moor. A comparative study of LS-SVMs applied to the Silver Box identification problem. In *Proceedings of the 6th IFAC Symposium on Nonlinear Control Systems (NOLCOS 2004)*, Stuttgart, Germany, September 2004.

- L. De Lathauwer, L. Hoegaerts, and J. Vandewalle. A Grassmann-Rayleigh quotient iteration for dimensionality reduction in ICA. In *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, Granada, Spain, September 2004.

- L. Hoegaerts, L. De Lathauwer, I. Goethals, J.A.K. Suykens, J. Vandewalle, and B. De Moor. Efficiently Updating and Tracking the Dominant Kernel Principal Components. *Submitted for publication*, 2005.

- I. Goethals, L. Hoegaerts, V. Verdult, J.A.K. Suykens, and B. De Moor. Subspace Identification of Hammerstein-Wiener Systems using Kernel Canonical Correlation Analysis. *Submitted for publication*, 2005.