

KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculteit ingenieurswetenschappen
Department elektrotechniek

Sparse kernel-based models for speech recognition

Promotoren:
Prof. dr. ir. J.A.K. Suykens
Prof. dr. ir. H. Van hamme

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door
Peter KARSMAKERS

May 2010

Sparse kernel-based models for speech recognition

Jury:
Prof. dr. ir. P. Van Houtte, voorzitter
Prof. dr. ir. J.A.K. Suykens, promotor
Prof. dr. ir. H. Van hamme, promotor
Prof. dr. ir. D. Van Compernolle
Prof. dr. ir. S. Dupont
(Faculté Polytechnique de Mons)
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. B. Vanrumste
(Katholieke Hogeschool Kempen)
Prof. dr. ir. V. Wertz
(Université Catholique de Louvain)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door
Peter KARSMAKERS

May 2010

© Katholieke Universiteit Leuven – Faculteit Ingenieurswetenschappen
Kasteelpark Arenberg 10, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden 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.

Legal depot number D/2010/7515/40
ISBN 978-94-6018-201-3

Voorwoord

Een aantal jaren intensief werken heeft uiteindelijk geleid tot het huidige proefschrift. Al vanaf het begin van mijn carrière als onderzoeker kon ik onderzoek combineren met een educatieve taak als assistent aan de KHKempen. Hoewel de balans tussen onderzoek en onderwijs niet altijd gemakkelijk te maken viel, ben ik door beide zowel persoonlijk als professioneel geëvolueerd.

Vooreerst zou ik een aantal mensen willen bedanken die mij vooral in de opstartfase hebben geholpen. In de eerste plaats wil ik Jürgen Van Gorp bedanken: Jürgen, dankzij jouw expertise en enthousiasme heb ik gekozen voor onderzoek in het wetenschappelijk domein van het automatisch leren wat me nog altijd fascineert. Jij wist me uiteindelijk ook te overtuigen om een doctoraat te gaan doen en stelde me voor aan de juiste mensen binnen de academische wereld. Graag wil ik ook ons toenmalige departementshoofd van de KHKempen Jef Vanroye bedanken. Hoewel dit niet evident was, heeft hij me van in het begin gesteund om een doctoraat op te starten. Verder wil ik ook prof. Joos Vandewalle bedanken die me toen als onbekende heeft ontvangen om me te helpen bij het uitstippelen van het pad van industrieel ingenieur naar een doctoraat.

Eenmaal de juiste kwalificaties behaald waren ben ik in contact gekomen met mijn huidige promotoren prof. Johan Suykens en prof. Hugo Van hamme. De kruisbestuiving van hun expertises heeft geleid tot het voorgestelde onderzoek. Graag zou ik hen willen bedanken: Johan, bedankt dat jij, als mijn superviserende promotor, me de kans gaf om onderzoek te doen binnen de SCD afdeling, me regelmatig opvolgde en me bijstuurde waar nodig. Jouw expertise, continue ondersteuning en aanmoedigingen zijn zeer belangrijk geweest voor mij en dit werk. Hugo, bedankt dat je altijd bereid was om mij te helpen bij mijn vragen, hoe onregelmatig dat ik ze ook stelde. Snel even een extra spraakexperimentje ontwerpen en/of programmeren was geen probleem. Ik bedank jullie beide voor de wetenschappelijke inzichten en opbouwende kritische analyses die me heel veel

geholpen hebben in de afgelopen jaren. Verder wil ik ook de assessoren bedanken voor hun constructieve bijdrage voor het verbeteren van deze tekst.

Graag zou ik mijn dankbaarheid willen betuigen aan mijn vele collega's van op verschillende fronten. Hierbij denk ik in de eerste plaats aan Kristiaan Pelckmans die als post-doc mijn werk dagelijks opvolgde. Kristiaan, bedankt voor je ongelofelijke creativiteit, brede expertise en bereidwilligheid om al mijn vragen nauwkeurig te beantwoorden. Het was soms frustrerend om te zien hoe snel jij een probleem kan oplossen. Verder mag ik mijn directe SCD collega's Jos, Kris, Marco, Tillmann, Carlos, Fabian niet vergeten te bedanken. *Thanks for the cooperation.* Niels bedankt voor de babbels in de wagen. Ook Kris H., Kris D., Jacques en Tingyao wil ik bedanken voor de samenwerking. Daarnaast mag ik de ondersteuning die ik gekregen heb van mijn directe (ex-)collega's en het huidig departementshoofd in Geel ook niet vergeten: Jan, Rob, Patrick, Vic, Staf, Guy, Herman, Jürgen, Paul, Joan, Bart, Peter S., Tom en Wim bedankt voor de collegialiteit en de mogelijkheden die jullie me gaven om dit onderzoek te kunnen voltooien.

Een groot woord van dank wil ik richten tot mijn ouders, zussen en broer en mijn familie voor al hun steun en toeverlaat de voorbije jaren. Bedankt allemaal!

Ik besef maar al te goed dat dit werk er niet was kunnen komen zonder de steun van mijn vrouw. Leen, bedankt dat je er altijd bent voor mij en dat je me altijd blijft steunen ook tijdens de moeilijke momenten. Leen, Eline en Rune, dikwijls heb ik jullie niet de aandacht gegeven die jullie verdienden. Hoewel het niet gemakkelijk zal zijn, ga ik er werk van maken om in de toekomst een juist evenwicht tussen gezin en werk te vinden. Graag zou ik dit proefschrift aan jullie willen opdragen.

Korte inhoud

Deze thesis bestudeert de integratie van een specifieke familie van leermethodes, die modellen leren op basis van een set van voorbeelden, in een Automatisch SpraakHerkennings (ASH) raamwerk. Meer exact gaat het over een groep van kernel gebaseerde leermethodes. Kernel gebaseerde methoden integreren technieken van convex optimaliseren, functionaal analyse en statistische leertheorie in een krachtige raamwerk. Bovendien blijkt deze set van methoden heel goed te werken voor een brede waaier van toepassingen zoals in bioinformatica, financiële ingenieurstechnieken, of tijdreeks predictie. In de literatuur van ASH worden deze technieken echter niet vaak gebruikt. Mogelijk is dit grotendeels te wijten aan hun typisch hoge rekenintensiteit.

Huidige ASH standaarden zijn vaak gebaseerd op het modelleren van de onderliggende mechanismen die de data genereren. De resulterende modellen worden dan gebruikt om een classificatie te doen. Bijvoorbeeld, voor het geval dat spraakeenheden (bijv. fonemen) dienen herkent te worden, wordt er afzonderlijk per foneem een model geschat. Nadien worden deze modellen gecombineerd om voor een nieuw stuk spraaksignaal een foneemclassificatie uit te voeren. In contrast hiermee proberen kernel gebaseerde methodes om rechtstreeks geschikte classificatieregels te leren die goed generaliseren, wat betekent dat ze ook goede resultaten geven op data die nog niet gezien werd tijdens het modelleren. Veel voorkomende implementaties van dergelijke methodes zijn Support Vector Machines (SVM) en Least Squares Support Vector Machines (LS-SVM).

De volgende 4 aandachtspunten zijn cruciaal in het ontwerp van kernel gebaseerde methoden in een context van ASH: (i) de grootte van de data set waarmee de leermethode overweg moet kunnen; (ii) de nood om voorbeelden met een variabele lengte te verwerken; (iii) de wens om modelresultaten met probabilistische interpretatie te hebben; (iv) de nood om meer dan 2 klassen van elkaar te onderscheiden.

Ten eerste, omdat het leren van kernel gebaseerde modellen een algoritmische complexiteit heeft die typisch kwadratisch schaalt met het aantal trainingsvoorbeelden, is een benaderende methode vereist. Deze thesis maakt gebruik van een primair-duaal context, dewelke afkomstig is van de theorie van convexe optimalisatie, om benaderende oplossingen te bekomen. Meer specifiek, beschrijft deze thesis hoe de Nyström benadering in combinatie met de fixed-size LS-SVM methode een duidelijk raamwerk geeft om een goede benaderende oplossing te bekomen. Een belangrijke consequentie van deze oplossing is dat het aantal parameters verschillend van nul wordt gereduceerd (of met andere woorden dat de spaarsheid van het model verhoogd wordt). Dit resulteert in snellere model evaluaties en kan eventueel leiden tot een betere generalisatie. Ten tweede, gezien het feit dat spraakeenheden kunnen voorgesteld worden als een akoestisch signaal met een variabele lengte, zijn in het algemeen de grenzen van de spraakeenheden niet gemakkelijk te onderscheiden van elkaar. Hierdoor moet de classificatiemodel, dewelke signalen van variabele lengte vertaalt naar verschillende spraakeenheden, overweg kunnen met de variabele lengtes van de spraaksignalen. Ten derde, een typische ASH omgeving is opgebouwd uit meerdere modules die communiceren met elkaar. De informatie van elk van de modules wordt typisch samengebracht door waarschijnlijkheden met elkaar te combineren. Vandaar dat een nieuwe methode, waarvan de resultaten een probabilistische interpretatie hebben, makkelijker te integreren is. Ten vierde, het ASH probleem is een probleem waarbij meer dan 2 klassen van elkaar moeten onderscheiden worden, het komt er simpelweg op neer om een set van spraakeenheden van elkaar te onderscheiden (die nadien worden gecombineerd tot zinnen).

Dit manuscript werkt 2 methoden in detail uit die aan bovengestelde eisen voldoen. De eerste methode vertrekt van een reeds bestaande methode, genaamd Kernel Logistic Regression (KLR), die voor de laatste 2 aandachtspunten een elegante oplossing biedt en breidt deze uit zodanig dat het kan gebruikt worden in een ASH context. De 2de methode mikt op een zo compact (spaars) mogelijke oplossingen om korte evaluatietijden te verkrijgen. Hiervoor werd een generische methode ontwikkeld die benaderende spaarse oplossingen vindt van een lineair systeem, met als naam *Sparse Conjugate Directions Pursuit*. Toegepast in de context van LS-SVMs worden spaarse kernel modellen bekomen. Experimenten tonen aan dat state-of-the-art resultaten behaald worden in een ASH testopzet.

Abstract

This thesis studies the integration of a particular family of machine learning methods into an Automatic Speech Recognition (ASR) framework. More precisely, the class of kernel-based machine learning methods is considered. Kernel-based methods integrate techniques from convex optimization, functional analysis and statistical learning theory into a powerful framework, and are found to work well for a wide range of application tasks such as in bioinformatics, financial engineering, or time series prediction. Those techniques are however not found often within the domain of speech recognition, mainly due to their typical computational demand.

Current ASR standards are often based on modeling the underlying mechanisms which generate the data. These models are then applied to perform classification. Kernel-based methods however simply aim for learning appropriate classification rules that generalize well, i.e. have good results on unseen examples. Common implementations of such methods are Support Vector Machines (SVMs), and Least Squares SVMs (LS-SVM).

There are four main concerns for designing kernel-based methods applicable in the context of ASR which are (i) the size of the data sets which the learner is processing (ii) the need for dealing with input examples of variable length, (iii) the desire to have probabilistic outcomes, and (iv) the need to perform multi-class classification. Firstly, since training kernel models implies a computational complexity which typically scales up quadratically in terms of the number of examples, an approximation is required. This thesis uses a primal-dual context which originates from the theory of convex optimization to accomplish approximative solutions. Specifically, we describe how the method of Nyström approximation and fixed-size LS-SVMs provides a clear framework to do so. A valuable consequence of the proposed method is that the number of nonzero variables is reduced (or that 'sparsity' is increased) which leads potentially to better generalization performance and allows faster model evaluations. Secondly, given

that a certain word might be represented as an acoustic signal of different length, the word boundaries are not well-defined in general. This requires the classifier, which maps speech signals to different words, to be able to deal with the duration variability of the considered speech signals. Thirdly, probabilitistic outcomes are needed in order to combine results from a new method with information from other ASR modules. Fourthly, ASR is a multi-class problem since it basically amounts to classifying a larger set of words, or phones in a vocabulary.

This manuscript describes two methods in detail which satisfy the above criteria. The first method starts from an already existing formalism, called Kernel Logistic Regression (KLR) which provides an elegant solution to the latter two issues, and extends it such that it can be used in an ASR context. The second method aims at a *sparse* solutions to obtain short evaluation times. For this purpose a generic method was developed that searches for approximate sparse solutions to linear systems, it is called *Sparse Conjugate Directions Pursuit*. Applied in a context of LS-SVMs, sparse kernel models are obtained. Experiments indicate that state-of-the-art results are obtained in an ASR test case.

Notation

Variables and Symbols

x	Vector in lowercase letter
X	Matrix in capital letter
$X^\dagger$	Pseudoinverse of X
$\mathbf{X}$	Random variable in boldface
x^T	Transpose of the vector x
Ω^T	Transpose of the matrix Ω
$(x)_i$	i-th component of vector x
Ω_{ij}	The ij-th element of matrix Ω
$\mathcal{S}$	Set of objects (e.g. set of vectors, indices)
$X(\mathcal{B})$	Subset of rows from a matrix or vector where $\mathcal{B} \in \mathbb{N}^l$ is as set of indices
$\lvert\mathcal{S}\rvert$	Number of elements of set $\mathcal{S}$
$w^{(k)}$	Marks differences of w between algorithmic iterations
$e_{\pi(i)} \in \mathbb{R}^D$	The i-th unit vector of size D
$I \in \mathbb{R}^D$	$D \times D$ identity matrix
$\mathcal{I}(y_i = j)$	Indicator function which equals to 1 if $y_i = j$ otherwise 0
1_D	D dimensional vector of all ones
0_D	D dimensional vector of all zeros
$\{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^D \times \{-1, 1\}$	Training set of N data points
$\varphi(\cdot)$	Feature map

$K(x, x')$	Mercer kernel evaluated on data points x and x'
$\min_x f(x)$	Minimization of cost function $f(x)$ over x, minimal function value returned
$\arg\min_x f(x)$	Minimization over x, optimal value of x returned
$w_{ci} \in \mathbb{R}^C D$	Double vector indices indicate a component ordering such as $w = (w_{11}, \ldots, w_{1D}, \ldots, w_{C1}, \ldots, w_{CD})^T$
$w_{c.}$	Selects all elements from the double indexed vector with first index equal to c
$\mathrm{diag}(X)$	Diagonal of matrix X $((\mathrm{diag}(X))_i = X_{ii})$
$\mathrm{diag}(x)$	Diagonal matrix with i-th diagonal elements equal to $(x)_i$
$\mathrm{blockdiag}(X_1, ..., X_C)$	Block diagonal matrix with the blocks on the diagonal equal to $X_1, \ldots, X_C$
$\underline{\mathrm{U}} = (u_1, ..., u_{N_u})$	Ordered sequences of units
$\det(X)$	Determinant of X
$\mathcal{N}(x; \mu', \sigma)$	univariate normal distribution with mean μ' and variance σ^2
$\mathcal{N}(x; \mu, \Sigma)$	multivariate normal distribution with mean vector μ and covariance matrix Σ

Acronyms

ANN	Artificial Neural Network
ARD	Automatic Relevance Determination
ASR	Automatic Speech Recognition
BP	Basis Pursuit
BSR	Backward Stepwise Regression
CG	Conjugate Gradient
CL	Conditional maximum Likelihood
CPU	Central Processing Unit
CSA	Coupled Simulated Annealing
CV	Cross-Validation
CDBoost	Conjugate Direction Boosting
DTAK	Dynamic Time Alignment Kernel
DTW	Dynamic Time Warping
EM	Expectation Maximization
EVD	EigenValue Decomposition
FS-LSSVM	Fixed size Least Squares Support Vector Machine
FS-MKLR	Fixed size Multi-class Kernel Logistic Regression
FSR	Forward Stepwise Regression
FStR	Forward Stagewise Regression
GMM	Gaussian Mixture Model
GP	Gaussian Process
HMM	Hidden Markov Model
IRLS	Iteratively Re-weighted Least Squares
IVM	Import Vector Machine
KLR	Kernel Logistic Regression
KNN	K Nearest Neigbors
KMP	Kernel Matching Pursuit
LARS	Least Angle RegreSsion
LDA	Linear Discriminant Analysis
LOO	Leave-One-Out
LS-SVM	Least Squares Support Vector Machine
MAP	Maximum A-Posteriori
MARK	Multiple Additive Regression Kernel
MFCC	Mel Frequency Cepstral Components
MKLR	Multi-class Kernel Logistic Regression
ML	Maximum Likelihood

MLP	Multi-Layer Perceptron
MMI	Maximum Mutual Information
MP	Matching Pursuit
MSE	Mean Squared Error
OLS	Orthogonal Least Squares
OMP	Orthogonal Matching Pursuit
PDF	Probability Density Function
PNLL	Penalized Negative Log Likelihood
PV	Prototype Vector
QP	Quadratic Program
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SCDP	Sparse Conjugate Directions Pursuit
SCDPP	SCDP Probabilistic
SCDP-FSLSSVM	Sparse Conjugate Directions Pursuit FS-LSSVM
SMO	Sequential Minimal Optimization
SGGP	Sparse Greedy Gaussian Process
SV	Support Vector
SVM	Support Vector Machine
TDNN	Time Delay Neural Network

Contents

Contents xi

1 Introduction 1

1.1 General Background 1

1.1.1 Speech unit 3

1.1.2 Building blocks for automatic speech recognition 4

1.2 Machine learning 5

1.2.1 Learning 7

1.2.2 Parametric versus non-parametric models 8

1.2.3 Model inference 8

1.2.4 Conceptual classification approaches 10

1.2.5 Model Selection and Bias-Variance Trade-off 12

1.2.6 Kernel-based classifiers 14

1.3 Motivation 17

1.4 Challenges and objectives 17

1.5 Chapter by Chapter Overview 19

1.6 Contributions of This Thesis 21

2 Automatic Speech Recognition 25

2.1 The Bayesian Recognition Paradigm 25
2.2 Frame-synchronous HMM Speech Recognition 28
2.2.1 The acoustical model as an HMM 28
2.2.2 Context-dependent phone models 30
2.2.3 HMM training and decoding 31
2.2.4 HMM assumptions and problems 32
2.3 Segment-based Speech Recognition 34
2.4 Summary 37

3 Kernel-based Methods **39**

3.1 Kernel-based learning 40
3.2 Support Vector Machines 40
3.2.1 Large-scale implementation 44
3.2.2 Sparseness 44
3.3 Least-Squares Support Vector Machines 46
3.3.1 Large-scale implementations 47
3.3.2 Sparseness 48
3.3.3 Fixed-size Least Squares Support Vector Machines 49
3.4 Kernel Logistic Regression 53
3.4.1 Large-scale implementation 56
3.4.2 Sparseness 56
3.5 Cross-validation based model selection 57
3.6 Estimating A-Posteriori Probabilities 58
3.7 Multi-class Coding Schemes 60
3.8 Learning with a universum 62

4 Fixed-size Kernel Logistic Regression **65**

4.1 Introduction 66
4.2 Modeling with multi-class kernel logistic regression 67
4.2.1 Linear multi-class logistic regression 67
4.2.2 Multi-class kernel logistic regression 70
4.2.3 Correction for unbalanced data sets 71
4.2.4 Inference with a universum 72
4.2.5 Multi-class coding schemes 72
4.3 Large-scale implementation 73
4.3.1 Fixed-size approach: estimation in the primal weight space 73
4.3.2 Solving the problem in primal space: Newton Trust Region approach 76
4.3.3 Very large data sets 81
4.3.4 Computational complexity 82
4.4 Experiments 83
4.4.1 Training time 84
4.4.2 Negative log likelihood in terms of number of classes 84
4.4.3 Prototype vector selection 86
4.4.4 Balancing and tuning multiple hyper-parameters 93
4.4.5 Include universum data 97
4.5 Conclusion 97

5 Sparse Conjugate Directions Pursuit with application to FS-LSSVM 101

5.1 Introduction 102
5.2 Heuristics to obtain a sparse approximation 104
5.3 Greedy heuristic: Sparse Conjugate Directions 109
5.3.1 Conjugate Gradient Method 110
5.3.2 Modified CG using Sparse Conjugate Directions 112

5.3.3 Computational Complexity 118
5.3.4 Computational issues 118
5.4 Application: using SCDP on FS-LSSVM formalism 120
5.4.1 Determining the final model size 123
5.4.2 Algorithmic parameters 125
5.4.3 Fast v-fold cross-validation 125
5.4.4 Relation to other methods 126
5.4.5 Comparison of computational complexity 128
5.5 Experiments 129
5.5.1 Binary classification problems 129
5.5.2 Multi-class classification problems 136
5.6 Conclusion 137

6 Segment-based Phone Recognition using Kernel Methods 139

6.1 Introduction 140
6.2 Modeling the Phone Probability Factor 141
6.3 Modeling the Segmentation Probability Factor 143
6.4 Phone Recognition 145
6.5 Related work 146
6.6 Experiments 149
6.6.1 Phone classification 149
6.6.2 Phone recognition 153
6.7 Conclusion 157

7 Conclusion 159

7.1 Concluding Remarks 159
7.2 Future Research 161

A Primal-dual formulation of MKLR **165**

A.0.1 Gradient and Hessian 165

A.0.2 Iteratively re-weighted least squares 166

A.0.3 Iteratively re-weighted least-squares support vector machines 167

B Benchmark data sets **169**

B.1 Data sets only for classification experiments 169

B.2 `timit` benchmark data set 170

Bibliography **173**

Curriculum vitae **191**

Publications by the author **193**

Chapter 1

Introduction

Here we describe the general setting of this thesis. First some background regarding the speech recognition process is discussed in Section 1.1. After briefly explaining some important choices which influence the complexity of the automatic speech recognizer, in Section 1.1 it is identified which task this thesis aims at. Then a sketch of a current state-of-the-art speech recognizer architecture is given. A core element, called acoustic model, within this task is the inference of a statistical relation between human generated acoustic signals and speech units. This core module is a specific type of problem which can be tackled using a wide range of machine learning methods. This concerns the process of induction of mathematical models from a finite set of observations. Section 1.2 gives an overview of a number of basic principles. Then, it is motivated why in this thesis a specific family of machine learning methods for integration into a speech recognizer was chosen. The challenges and objectives of the presented work are given in Section 1.4. Next, in Section 1.5 the structure of the manuscript is outlined. Finally, the main contributions of the conducted research are enumerated in Section 1.6.

1.1 General Background

The general setting of this thesis can be found in the context of Automatic Speech Recognition (ASR). With ASR this work refers to the process and the related technology for converting an acoustic speech signal into a sequence of words (or other linguistic units) by means of an algorithm implemented as a computer program. Automatic speech recognition has many applications. Over the last few

years several examples emerged including voice dialing, call routing, interactive voice response, voice search, data entry and dictation, command and control (voice user interface with the computer), structured document creation (e.g. medical and legal transcriptions), appliance control by voice, computer-aided language learning, content-based spoken audio search, and robotics (see e.g. [79]).

ASR systems are usually designed for a specific purpose. Incorporating knowledge of the task at hand is likely to perform better in terms of successful recognition of the users' words. Each application has a different difficulty (complexity) level. Recognizing a set of digits, which are spoken clearly separate from each other, might lead to an ASR with higher accuracy than recognizing fluently spoken speech.

In order to define the precise ASR process this thesis aims at, a list of criteria which have a substantial influence on the complexity of the problem, are:

- **Speech vocabulary:** A speech vocabulary is the list of words a speech recognition program is able to distinguish. The use of a larger vocabulary might be harder for a program since the number of words to choose from is increased. Put in other words, a larger search space increases the overall complexity of the speech recognition task. Besides vocabulary size other measures such as perplexity and word length have an impact on the complexity of the vocabulary. Firstly, perplexity indicates the mean number of possible speech units given the context. A higher perplexity results in a more difficult task. Secondly, longer words carry more acoustic information (which can be used for discrimination) than shorter ones which makes the ASR task easier.
- **Isolated or Continuous mode:** An isolated-word recognizer requires the user to clearly pause between each spoken word. A Continuous Speech Recognition (CSR) system is designed to recognize fluently spoken speech. Compared to CSR the isolated-word recognizer task is simpler since the word boundaries in the acoustic speech signal are easier to find and the pronunciation of a word tends not to affect others. In case of CSR, additionally a segmentation of the speech signal into parts, hypothesized as related to a word, is required and the start and end of words are affected by the preceding and following words. This is known as "co-articulation" [83].
- **Speaker dependency:** A speaker-dependent system is developed to operate for a single speaker. These systems are usually easier to develop and more accurate, but not as flexible as speaker-independent systems. A speaker-independent system is developed to operate for any speaker of a particular language (e.g. Dutch). These systems are harder to develop and accuracy is likely to be lower than speaker-dependent systems due to large

acoustic variabilities between different speakers. However, they are more flexible.

- **Adverse environments:** The definition of the term "adverse environments" as in [150] implies unknown, mismatched, and often severe differences in environment and other effects between development (training) and "actual recognition" (testing). Recognition in adverse environments is likely to be more complex than assuming equal recording conditions for both designing and recognizing. These differences are inevitable in real-world applications. Examples of such varying conditions are unknown noise (e.g. radio is playing while speaking), speech signal distortion (e.g. different microphones are used for design and recognition), articulatory effects (e.g Lombard effect [1]).
- **Adaptivity:** Adaptivity it is the ability of the ASR to adjust to operating conditions. This has the advantage that recognizers can adapt themselves to conditions not foreseen during the development (training) process. Example include a speaker-adaptive system which is developed to adapt its operation to the characteristics of new speakers or adaptive techniques can be used to normalize the mismatch across different environment conditions [83].

The speech recognition framework in this thesis aims at a *continuous non-adaptive speaker-independent speech recognition* task where it is assumed that speech is recorded in *equal clean conditions* for both design and operation (recognition) times.

1.1.1 Speech unit

In case the intention is to recognize a very large vocabulary (tens of thousands of words) it is difficult to relate the acoustic signals produced by a person and corresponding sequence of words directly [83]:

- Every new task contains novel words without any available training data, such as proper nouns and newly invented jargons.
- There are simply too many words, and these different words may have different acoustic realizations. And as will become clear later, each word needs a sufficient amount of repetitions to be described well by a model. This can be unrealistic in case of a large vocabulary.

[1] The Lombard effect is the involuntary tendency of speakers to increase the intensity of their voice when speaking in loud noise to enhance its audibility. This change includes not only loudness but also other acoustic features such as pitch, rate and duration of sound syllables. [92]

Therefore recognition on large vocabularies is usually decomposed into recognition of smaller speech units which are combined afterwards to form words. Examples of such units are syllables, phonemes, or phones. The selection of the most basic units to represent salient acoustic and phonetic information for a certain natural language is an important issue in designing a workable system [83].

Remark 1. Remark that there is a subtle difference between a phone and phoneme. A phone is the smallest segmental unit of sound employed to form meaningful contrasts between utterances without regards to a specific language. A phoneme is the smallest unit in speech that carries linguistic meaning [83]. It can be thought of as a set of phones which carry the same meaning. An example of a phoneme is the /k/ sound in the words **k**it and s**k**ill (phonemes are placed between slashes). Even though most native speakers do not notice this, in most dialects, the k sounds in each of these words are actually pronounced differently: they are different speech sounds, or phones (which are placed in square brackets). In our example, the /k/ in **k**it is aspirated, [k^h], while the /k/ in s**k**ill is not, [k].

This dissertation chooses to use units from a broad phonetic class which is defined as a mapping of phones into broad (or clustered) phonetic categories [120] (but the theory also holds for other units).

1.1.2 Building blocks for automatic speech recognition

In order to sketch its operation we now review briefly the basic building blocks appearing in our ASR system. The review is based on the textbooks [83], [149]. A prototypical setup is given in Fig. 1.1[2]. The separate blocks are:

- **Acoustic preprocessing:** Acoustic preprocessing aims at transforming an acoustic signal produced by human speakers to a compatible representation for the speech recognizer software. The final representation should reduce as much as possible the variability of speech in order to cover a wide range of purposes. For example, in case of a speaker-independent system the accuracy should ideally be independent of the speaking rate or pitch[3]. In order to accomplish this the acoustic speech signal acquired from a microphone transducer is passed first through an analog-to-digital converter. In order to reduce as much as possible the variability of speech the digital signal is preprocessed using digital signal processing techniques. As a result a sequence

[2]Note that since this thesis does not consider an adaptive setup to adjust speech recognizers to varying usage conditions, no module regarding this subject was added as building block.

[3]Pitch is the perceived property of a sound measured by its frequency.

of so called *observation* vectors (or *frames*) is obtained which characterize acoustic properties of overlapping analysis windows of speech.

- **Acoustic model:** An acoustic model is defined as a statistical relation between a sequence of observation vectors and a sequence of words. As mentioned before, since it is difficult to define one acoustic model to account for every possible word sequence, the process is broken up in recognizing smaller units. A recognized sequence of units can then later on be processed at a higher level which e.g. can then result into a sentence hypothesis.
- **Language model:** A language model is used to assign probabilities to sequences of recognized words depending on how well this sequence fits into a language. Surely, not all sequences of units are sentences. The units have to form a sequence of words, and that sequence has to obey the syntax of a language. In this thesis N-grams are used as language models. An N-gram model is a type of probabilistic model for predicting the next item in a sequence. An N-gram is a subsequence of N items from a given sequence.
- **Knowledge base:** A knowledge base contains additional information to improve recognition performance. Examples include a phonetic lexicon which is a list of all the words in the language and their phonetic transcriptions, and semantic models.
- **Decoder:** The decoding (search) process of a speech recognizer finds a sequence of words (the most likely sentence hypothesis) whose corresponding acoustic and language models most likely match the input signal. The complexity of a search algorithm is highly correlated with the search space, which is amongst others determined by the constraints imposed by the language models.

We give a formal description of this process in Section 2.1.

1.2 Machine learning

Machine learning concerns the process of learning mathematical models from a finite set of observations. As indicated earlier an acoustic model maps acoustic speech signals to units. Learning the devised ASR task is a specific type of problem which can be tackled using a wide range of machine learning methods. This approach seems appropriate since the complex nature of speech recognition prohibits to construct a set of physical rules to accomplish a statistical relation.

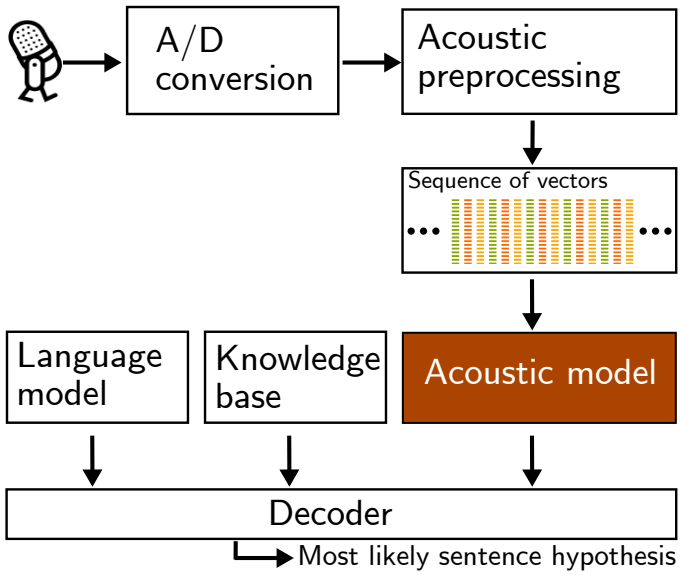


Figure 1.1: Principal state-of-the-art automatic speech recognizer architecture for continuous speech recognition.

Instead, acoustic models are learned based on a (preferably large) set of real-life example speech utterances.

Within the machine learning domain one distinguishes different learning tasks. This thesis focusses on supervised learning which attempts to learn a model by examples both containing a representation (”input”) as well as a corresponding label (”output”). In ASR this corresponds to examples consisting of a ”speech signal” (input) and corresponding ”unit” (output).

Note that the considered ASR setup has a training step which concerns building up the acoustical models. These models then remain unchanged when employed by the ASR system for recognition (testing) purposes. This must be contrasted to learning methods which try to adapt to changing environmental conditions during the recognition (testing) step. These are not considered in this thesis.

1.2.1 Learning

Following the notation such as in [142], or [15] the machine learning concept can be described as

$$\text{Alg} : \mathcal{D} \times \mathcal{A} \rightarrow \mathcal{F}, \tag{1.1}$$

where Alg defines the mapping from a set of observations $\mathcal{D}$ and a collection of assumptions and prior knowledge $\mathcal{A}$ to the estimation class $\mathcal{F}$.

- **Estimation class $\mathcal{F}$:** The estimation class is the set of all estimators which are a possible outcome of the considered learning algorithm Alg. One typically restricts the members of the estimation class (in our context called "models") to a certain representation. Examples include rule-based models used to perform lexical analysis on e.g. natural language processing, models with probabilistic interpretation, or models which output discrete class labels. For the latter two main categories can be distinguished: parametric and non-parametric techniques (see next section). Depending on the output type one refers to the models with different names.

- **Assumptions $\mathcal{A}$:** Incorporating exact and inexact prior knowledge (assumption) can provide additional information to the learning process. Prior knowledge about the structure of the problem at hand can affect learning performance. This knowledge can be embedded in the learner by restricting the set of models in the estimation class to a certain representation. An example of prior knowledge is the restriction to some type of distribution, from which the data is known to be generated from.

- **Data $\mathcal{D}$:** An algorithm learns a model from the data which is a set of observations with corresponding outputs or labels. Consider the set of N given observations

  $$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{1.2}$$

  of input samples $x_i \in \mathcal{X}$ and the corresponding observed output values $y_i \in \mathcal{Y}$. Different types of domains $\mathbb{D}$ are imaginable for the observed variables. Typical examples include: the continuous $\mathbb{R}$, the binary (e.g. $\{0, 1\}$), the categorical (e.g. $\{A, B, C, D\}$), the ordered $\{Low, Equal, High\}$. In case of speech recognition the observations consist of sequences of vectors and unit labels (e.g. phones) $\{(\underline{\mathrm{X}}_i, y_i)\}_{i=1}^{N}$ where $\underline{\mathrm{X}}_i$ denotes a sequence of feature vectors (e.g. from a phone segment of the full speech utterance) and y_i is the corresponding target phone label. Note that the learning task which

involves an output restricted to a set of labels $f_c : \mathbb{D}^D \rightarrow \{1, \ldots, C\}$ (with D the number of dimensions) is denoted as classification. The learning machine itself is called a classification method.

- **Learning algorithm** Alg**:** The learning algorithm is considered to be a uniquely defined mapping which fits the training data set and the set of assumptions onto one model estimate which is the "best" in some sense among alternatives. In this thesis the specific objective of the learning algorithm is to obtain models with adequate generalization abilities, i.e. which perform well on predicting class labels for unseen examples.

1.2.2 Parametric versus non-parametric models

The classical approach requires the designer to postulate a statistical model beforehand. Inference is then concerned with fitting the parameters in the predefined model. This can be written for example as

$$\mathcal{F}_w = \{f : \mathbb{R}^D \rightarrow \mathbb{R} | f(x) = w^T x, y = f(x) + e\}, \tag{1.3}$$

where e is assumed to be generated from some distribution function (e.g $e \sim \mathcal{N}(x; 0, 1)$).

In contrast non-parametric techniques do not explicitly formulate a family of statistical models, but instead define the estimator class by putting a proper set of constraints. As an example opposed to that of $\mathcal{F}_w$ the following could be stated

$$\mathcal{F}_\nu = \{f : \mathbb{R}^D \rightarrow \mathbb{R} | \ \|f(x)\|_1 < \nu\}, \tag{1.4}$$

no explicit definition of $f(x)$ (which is different from the previous definition of f) is given. Instead a restriction, by bounding the 1-norm $\| \cdot \|_1$, is posed.

Parametric models include the potential risk that the family of pre-specified models is not appropriate for the problem at hand. By considering a much broader range of class estimators this issue is solved by the the family of non-parametric models. For further details we refer to [52], [78], [172].

1.2.3 Model inference

The process of learning a model is called inference. This thesis uses two variants:

Maximum likelihood

Maximum Likelihood (ML) as a framework is a common general technique to infer model parameters. Consider the random variable $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ and a finite number of data samples i.i.d. from $\mathbf{Z}$ denoted as $\mathcal{D} = \{x_i, y_i\}_{i=1}^N \subset \mathbb{R}^D \times \{-1, 1\}$. The maximum of the likelihood $\Pr(\mathbf{Z}|\theta)$ of a parameter θ characterizing an element from a finite set of probabilistic rules, given the observations is denoted as

$$\theta^{\mathrm{ML}} = \arg\max_{\theta} \Pr(\mathbf{Z}|\theta) = \arg\min_{\theta} -\sum_{i=1}^{N} \log \Pr(\mathbf{Z} = z_i|\theta). \tag{1.5}$$

Structural risk minimization

Structural risk minimization is an inductive principle which investigates under which conditions empirical risk minimization results into consistent estimates minimizing the theoretical risk. The risk is a scalar which measures the deviation from the outputs of a specific model to the real output values. In order to have adequate generalization abilities the structural risk minimization principle balances the model complexity against its success at fitting the training data. Support Vector Machines (SVMs) [184], [20] are probably the most widely known implementation in this category.

The main theory describes the case of binary classifications. Let $\mathbf{X} \in \mathbb{R}^D$ and $\mathbf{Y} \in \{-1, 1\}$ be random variables with fixed but unknown probability density function $p(\mathbf{X}, \mathbf{Y})$. Let the theoretical risk R of any mapping $f : \mathbb{R}^D \to \{-1, 1\}$ be defined as follows

$$\mathrm{R}(f) = \int \mathcal{I}\left(f(x)y \leq 0\right) p(\mathbf{X}, \mathbf{Y}) d\mathbf{X} d\mathbf{Y}, \tag{1.6}$$

where the indicator function $I(x \leq 0)$ equals 1 if $x \leq 0$ and zero otherwise. The sample estimate, called empirical risk, of this integral based on a finite number of data samples of $\mathbf{X}$ denoted as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^D \times \{-1, 1\}$ becomes

$$\hat{\mathrm{R}}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}\left(f(x_i)y_i \leq 0\right). \tag{1.7}$$

Now, statistical learning theory [15] aims at defining theoretical insights under which circumstances the empirical risk will converge to the theoretical risk. Often, this convergence is phrased as a generalization bound, discussed in e.g. [159].

Remark 2. Another popular learning paradigm is called Bayesian learning [78], [12]. In a ML setting, θ is considered to be a fixed and error bars on this estimate are

obtained by considering multiple possible data sets from $\mathcal{D}$ (e.g. bootstrap [78]). This must be contrasted to the Bayesian viewpoint where there is only a single data set (namely the one that is actually observed), and the uncertainty in the parameters is expressed through a probability distribution over θ which is called the a-posteriori probability distribution $\Pr(\theta|\mathcal{D})$. In case a mode of this a-posteriori distribution is selected, the Maximum A-Posteriori (MAP) estimate is obtained

$$\theta^{MAP} = \arg\max_{\theta} \Pr(\theta|\mathcal{D}) = \arg\max_{\theta} \frac{\Pr(\mathcal{D}|\theta)\Pr(\theta)}{\Pr(\mathcal{D})}. \tag{1.8}$$

Factor $\Pr(\mathcal{D})$ can be discarded since it is independent of θ in the above maximization. Note that if $\Pr(\theta)$ is uniform then the MAP estimate would coincidence with the ML estimate.

In the remainder of this text we do not consider probability distributions over model parameters θ but rather give single estimates as outcomes of learning algorithms. In case a-posteriori probability is mentioned in the remainder of the text this does not refer to a Bayesian way of inference but rather indicates a type of conditional probability used in Bayesian decision theory.

1.2.4 Conceptual classification approaches

According to [12] one might divide classification methods into 3 distinct categories. These are given, in decreasing order of complexity, by:

Generative classifiers

Approaches that model the distribution of stochastic variables are known as generative models. Generative models can be used for the purpose of classification as follows. Suppose a C class problem. In case class conditional density estimates $\Pr(X = x|Y = c)$, $c = 1, \ldots, C$ and prior estimates $\Pr(Y = c)$ are obtained for each of the classes separately then by applying Bayes' rule we can write,

$$\Pr(\mathbf{Y} = i|\mathbf{X} = x) = \frac{\Pr(X = x|Y = i)\Pr(Y = i)}{\sum_{c=1}^{C} \Pr(X = x|Y = c)\Pr(Y = c)}. \tag{1.9}$$

In this manner (non-)parametric density estimation can be used for classification. Equivalently, the joint distribution $\Pr(X = x, Y = c)$ can be modeled directly and normalized afterwards to obtain the a-posteriori probabilities. Having found the a-posteriori probability estimates, the class membership can be decided for each new x. This type of modeling is used in e.g. the state-of-the-art ASR systems (see Chapter 2).

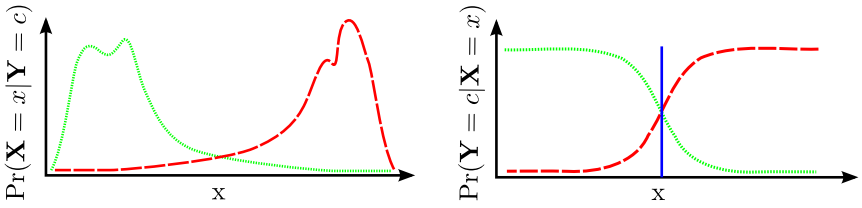


Figure 1.2: The left figure shows the individual class densities which show interesting structure which disappears in the figure on the right were the a-posteriori probabilities are plotted. The vertical blue line in the right plot indicates the decision boundary in X that gives the minimum misclassification rate.

Discriminative classifiers

Indirect probabilistic approach Approaches that model the a-posteriori class probabilities directly are called discriminative classifiers. Examples which utilize this type of modeling are logistic regression, its non-linear variant Kernel Logistic Regression (KLR) (see Section 3.4) and the discriminative training methods for Gaussian Mixture Models (GMMs) described in (Section 2.2.3) [4].

Direct approach In this case a discriminant function $f(x)$ maps each input x directly onto a class label. For instance, in the case of two-class problems, $f(\cdot)$ might be valued such that $f(\cdot) = -1$ represents class 1 and $f(\cdot) = 1$ represents class 2. In this case, probabilities play no role. The most popular example within this category is SVM.

Merits of each approach The generative approach is the most demanding because it involves finding the multivariate distribution over both X and Y. For many applications, X will have high dimensionality, and consequently a large training set might be needed in order to be able to determine the class-conditional densities to a reasonable accuracy. Note that the class priors $\Pr(Y = c)$ can often be estimated simply from the fractions of the training set data points in each of the classes. An advantage of the generative approach, however, is that it also allows the marginal density of data $\Pr(X = x)$ to be determined from $\sum_{c=1}^{C} \Pr(X = x|Y = c) \Pr(Y = c)$. This can be useful for detecting new data points that have low probability under the model and for which the predictions may be of low accuracy, which is known as outlier detection or novelty detection [12].

[4] Note that KLR can be catalogued under the non-parametric methods while the discriminative GMM techniques are parametric formulations.

However, if classification is the ultimate goal, then learning the separate class densities might not be needed and estimating the a-posteriori probabilities well near the decision boundary is sufficient [78] [12]. Fig. 1.2 presents a 2-class example where 2 multi-modal densities are given. From this figure it is clear that for classification purposes it is not relevant to learn the fine details of each of the densities (left part of Fig. 1.2). Instead, modeling the smooth a-posteriori probabilities (right part of Fig. 1.2) directly might be easier. Since no attempt is made to model all probability rules generating the data, less assumptions about the classes are made. This makes discriminative learning potentially more robust in classifying data opposed to the generative alternative.

The direct discriminative approach is even simpler where a discriminant function $f(\cdot)$ maps each input directly onto a class label. In Fig. 1.2 this would correspond to finding the vertical blue line. This line corresponds to the decision boundary giving the minimum probability of misclassification.

With the latter option, however, no a-posteriori probabilities $\Pr(Y = c|X = x)$ are available. As explained, the problem of speech recognition is usually decomposed into smaller tasks. As long as each of the modules give a-posteriori probabilities, one can combine the outputs systematically using the rules of probability.

1.2.5 Model Selection and Bias-Variance Trade-off

Designing a model for a particular application requires choosing a specific member from the estimation class $\mathcal{F}$. Model selection deals with the question which member is to be chosen. Before formalizing this process, first an example is given to illustrate the purpose of model selection. Consider the following ingredients:

- **Learning algorithm:** ridge regression, with the following optimization objective

$$\hat{w} = \arg\min_{w} \frac{1}{2}\|Xw - y\|_2^2 + \nu\|w\|_2^2, \tag{1.10}$$

  where w, y are vectors which respectively contain model weights, (to be predicted) output values (possibly class labels), and $X = (x_1, \ldots, x_N)^T$ the matrix with observation vectors. Besides minimizing the training error using the first term, additionally a second term is added which aims at a small norm of the solution vector.

- **Data:** Suppose a 1-D classification task with $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ where $x_i \in \mathbb{R}$ and $y_i \in \{-1, 1\}$.

- **Assumptions:** Assume that the true model is amongst the set of polynomial models $\mathrm{P}^M(x,w) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$. In order to search for the weights w using ridge regression the X matrix is constructed as follows $x_i' = (1, x_i, x_i^2, \ldots, x_i^M)^T,\ \forall i$, and $X = (x_1', \ldots, x_N')^T$.

In this example there are 2 tuning-parameters (hyper-parameters): the degree M of the polynomial and the regularization constant ν, which both have impact on the final estimate. Consider the data in Fig. 1.3a. Fitting a polynomial of degree 1 (and $\nu = 10^{-6}$) gives unsatisfactory results since it is clearly seen from the data that a model giving a non-linear decision boundary is required to separate the two classes. Note that the lines in Fig. 1.3 represents the class estimates obtained by signing the model outputs ($\hat{y} = \mathrm{sign}\,(Xw)$). This gives the full lines in the figures.If for example the degree is increased to $M = 15$ then a more complex non-linear decision boundary can be composed. However, allowing too much flexibility by the model might result in an oscillating and over-fitted solution as is seen Fig. 1.3b ($\nu = 10^{-6}$). This is because the polynomial interpolates the given training data but fails to generalize well in between the given training data points. A less oscillating result is given in Fig. 1.3c where the regularization parameter was set to $\nu = 0.1$. Fig. 1.3b and Fig. 1.3c conceptually indicate the bias-variance trade-off in terms of the regularization constant ν. A large value ν decreases the variance (complexity of the model) but leads to a larger bias (average deviation from the true underlying model).

In a practical application these tuning-parameters need to be determined based on a metric which can be formally written as

$$\mathcal{M}_{\mathrm{Modsel}} : \mathcal{A}(\Theta_1) \times \mathrm{Alg}(\Theta_2) \times \mathcal{D} \to \mathbb{R}, \tag{1.11}$$

in the present example the parameter vector Θ_1 would contain the degree parameter M and vector Θ_2 the regularization parameter ν.

Having a metric allows to choose the individual estimator by optimizing the following problem

$$\hat{\Theta} = \underset{\Theta_1, \Theta_2}{\arg\min}\, \mathcal{M}_{\mathrm{Modsel}}\left(\Theta_1, \Theta_2\right), \tag{1.12}$$

where $\hat{\Theta} = (\hat{\Theta}_1^T, \hat{\Theta}_2^T)^T$.

In case of classification, popular choices for $\mathcal{M}_{\mathrm{Modsel}}$ are (i) minimize the model error on a separate validation set; (ii) instead of working with a single validation set one can work with a v-fold cross-validation [21] procedure and choose the tuning-parameters in such a way that the sum of the errors on the validation sets that are

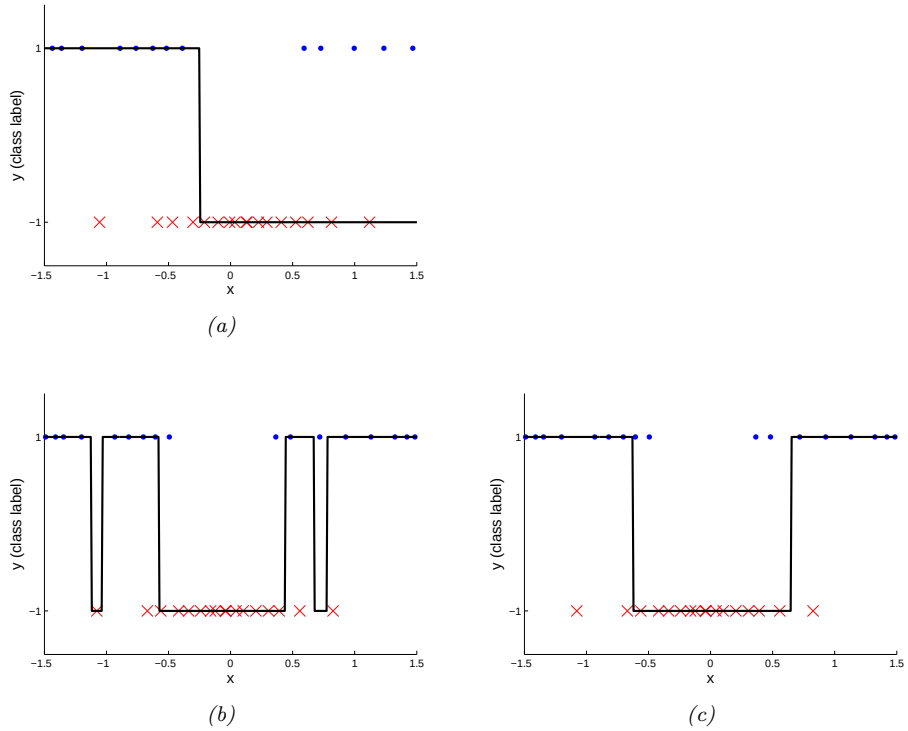


Figure 1.3: Estimation of a 1-D binary classifier using ridge regression with: a) a polynomial model of degree $M = 1$ and regularization parameter $\nu = 10^{-6}$, b) a polynomial model of degree $M = 15$ and $\nu = 10^{-6}$, c) $\nu = 0.1$. The model outputs are signed ($\hat{y} = \text{sign}(Xw)$) to obtain the class estimates. This gives the lines in the figures.

left out in several runs is minimal; (iii) applying methods of Bayesian inference which enable to do automatic selection of the tuning-parameters (e.g. [122]).

1.2.6 Kernel-based classifiers

In this thesis we aim at integrating a family of kernel-based classifiers in an ASR framework. This approach to pattern analysis first embeds the data in a suitable feature space, and then uses a linear method to discover patterns in the embedded data. The SVM is a popular example of such methods. Any kernel method solution consists of two main parts [159], [165]:

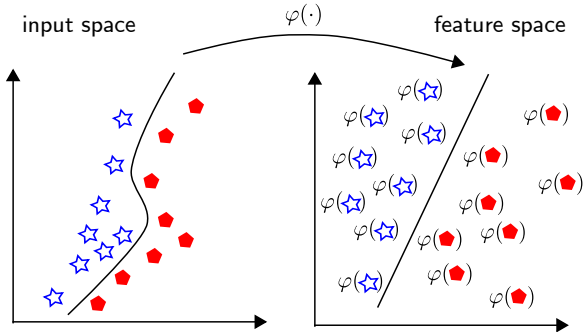


Figure 1.4: The function $\varphi(\cdot)$ embeds the data into a feature space where the non-linear decision line now appears linear. The kernel computes inner products in the feature space directly from the inputs.

- a mapping function that performs a transformation into a feature space and a learning algorithm designed to discover linear patterns in that space. There are two main reasons why this approach works well. Detecting linear statistical relations in data has been the focus of much research in statistics for decades, and the resulting algorithms are both well understood and efficient.
- As will be seen in Chapter 3 there is a convenient trick which makes it possible to represent linear patterns efficiently in high-dimensional spaces to ensure adequate representational power. The shortcut is what we call a kernel function $K(x, x') = \varphi(x)^T \varphi(x')$.

In Fig. 1.4 it is visualized what is meant by a feature map. Data samples x are embedded into a vector space, called the feature space, as $\varphi(x)$. Then linear relations are sought among the images of the data in the feature space.

The considered kernel methods additionally provide a well-defined mechanism to control the bias-variance trade-off, even in high dimensional feature spaces. In Chapter 3 each of the explained kernel methods is presented in its primal-dual formulation, resulting from the theory of convex optimization. This has the advantage that it can give insights in how to construct approximate schemes. The primal representation is used to formulate the desired optimality principle in combination with prior knowledge in the form of a constrained optimization problem. The dual representation refers to a problem in terms of Lagrange multipliers enabling the application of a positive semi-definite[5] kernel function.

[5]We use here the linear algebra terminology of *positive definite* and *positive semi-definite*. The corresponding terminology in functional analysis is often *strictly positive definite* and *positive definite*, respectively.

Plugging in kernels results in a non-parametric form in the dual formulation.

Scalability

For kernel models the computational complexity scales (worst case) quadratically with the number of data points N (denoted as $\mathcal{O}(N^2)$). Automatic speech recognition typically involves very large training set sizes. Attaining as much useful data as possible will result in models with higher confidence on the outcomes.

Remark 3. In this thesis large-scale data sets are considered to have more than $50,000$ training examples.

For large-scale data sets solving the exact problem (with complexity $\mathcal{O}(N^2)$) becomes infeasible on a standard computer[6]. In [75] an overview of existing approximate solutions in the literature is given. These are briefly summarized as follows:

- **Low-rank approximations:** By defining low-rank matrix (involved in training) approximations memory and processing requirements can be reduced. Classical results here are the recursive Cholesky and Nyström low-rank approximations (see e.g. [172], [75] for its application to LS-SVMs).
- **Sampling:** Using an appropriate sampling scheme, selecting a subset of the training set which represents the full set as accurately as possible, might still result in accurate modeling while reducing the computational load. In [172], and [18] an efficient algorithm called fixed-size LS-SVM is described which combines a Renyi entropy based sampling mechanism with the Nyström low-rank approximation with estimation in the primal.
- **Recursive Estimation:** Early stopping combined with recursively solving the kernel problems with growing model size give approximate results for the full problem without ever solving it.
- **Ensemble Learning:** Instead of using one model on the whole data set, a committee of sub-models each based on a subsample of the data could be employed. Examples include committee networks, bagging and boosting [78].

[6] We assume a standard computer to have 2 gigabytes of memory and a dual core 2.2 gigahertz processor.

1.3 Motivation

The current state-of-the-art in speech recognition is based on Hidden Markov Models (HMM) as will be discussed in Chapter 2. Such systems are typically trained using a generative approach. In an attempt to improve upon the discriminative ability of such speech recognizers Artificial Neural Network (ANN) based systems were used. Two commonly used neural network architectures are the Time Delay Neural Network (TDNN) [174] and the Recurrent Neural Network (RNN) [152], especially the RNN (i.e. partially RNNs which are written as static Multi-Layer Perceptrons (MLPs) for relatively short context windows) has been used in several successful connectionist hybrid speech recognition systems (see e.g. [14]). However, despite some achievements, none of these approaches could really outperform the results obtained with standard Hidden Markov Model (HMM) based ASR systems.

Potential advantages when using kernel-based methods are:

- **Convexity:** A drawback in using ANNs and HMMs is that their classical methodology results in non-convex optimization of parameters resulting in possibly suboptimal results. This must be contrasted to non-linear kernel-based methods such as SVM and LS-SVM which are based on well-defined convex optimization objectives with nice convergence (both on numerical as on generalization performance) and benefit from a vast amount of theoretical background.
- **High dimensional spaces:** Kernel-based methods are known to generalize well even in high dimensional spaces.
- **Customized kernels:** Kernel-based methods elegantly allow different types of representations through the use of a wide range of positive definite kernels.

However, kernel-based methods are not found often within the domain of automatic speech recognition. This thesis takes on the challenge to develop kernel-based methods for integration in ASR systems.

1.4 Challenges and objectives

In the literature kernel-based methods are known to give state-of-the-art results in several applications such as in computational biology (e.g. [157]), genome sequence classification (e.g. [164]), time series prediction (e.g. [56]), financial engineering

(e.g. [181]), text categorization (e.g. [112]), handwritten digit recognition (e.g. [110]), clustering profiles (e.g. [2]), face recognition (e.g. [144]), see also [74] for a benchmarking competition on several applications.

However, in order to allow for a smooth integration into a speech recognizer the following requirements are desired:

- **Scalability:** As already discussed in previous section in case of large scale problems, such as in ASR, even very efficient algorithms are prohibitive which suggest the need of an approximative scheme. A main topic of this thesis is to search for such approximate schemes while preserve as much as possible the characteristics of the original theoretically well-defined kernel method. In order to do this properly we use a primal-dual formulation which gives insights about the parametric and non-parametric relations in a certain context.
- **Sparseness:** Often closely related with the approximation scheme used to enable a scalable algorithm is the sparseness (the number of non-zero parameters) of the final model estimate. Since recognizing speech utterances involves evaluations of a huge amount of vector sequences it is desired to have models which allow fast computation. Sparser (more compact) models straightforwardly give faster evaluation times.
- **Multi-class classification:** Kernel-based methods are typically developed for binary classification. Several, mostly ad-hoc, approaches are available to extend those methods to the multi-class setting. The different strategies need to be compared.
- **Probabilistic outcomes:** As should be clear from the principal ASR architecture in Fig. 1.1 it is seen that a complete recognizer is composed out of different modules which need proper interaction. As a common interface a probabilistic language is normally used. In this sense it is important to develop methods which output results with probabilistic interpretation.
- **Variable length sequences:** Speech recognition consists of recognizing objects composed out of variable length vector sequences while classical kernel methods operate on objects with fixed-length vectors. This requires some sort of mapping from a variable length sequence to a fixed representation.

Throughout the thesis all these issues will be kept in mind when developing new algorithms to tackle speech recognition. Note that all algorithms are designed such that they can be generally used with any positive definite kernel function.

1.5 Chapter by Chapter Overview

This dissertation is organized in five chapters. The first and second chapter respectively introduce the main concepts of the current state-of-the-art of automatic speech recognition and supervised kernel-based learning. The latter three chapters present our contributions.

- **Chapter 2** discusses the current state-of-the-art in speech recognition. A core element in this process is the statistical relation between the symbolic speech units and the recorded digital signal. The current technology to accomplish this is based on Hidden Markov Models (HMMs). With a well-formed mathematical foundation, and efficient, automated training procedures which can process the ever increasing amounts of speech data, impressive HMM-based recognizers have been created for a wide-variety of increasingly difficult speech recognition tasks. Still, there are a number of assumptions that might ultimately limit the performance of HMM based systems. We therefore resort to a different framework which uses segmental features and has the potential to overcome the earlier described weaknesses and then indicate in which way kernel-based methods can be integrated in this segment-based ASR framework.
- **Chapter 3** gives a brief overview of the kernel methods which are devised to be used in a speech recognizer architecture. While having superior performances in terms of classification accuracies on unseen data on a wide range of applications, such methods are not yet frequently used in speech recognition mainly because they are computationally demanding. Although efficient training algorithms exist, in case of large-scale problems, as are common in speech recognition, solving the full optimization problem can even become untractable on a standard computer. We therefore additionally describe some typical techniques to approximate the standard formalisms in order to achieve realistic model training and evaluation times. In order to facilitate the integration into a speech architecture some general extensions to the original frameworks are given.
- **Chapter 4** puts particular emphasis on a non-linear "kernelized" variant of logistic regression called Kernel Logistic Regression (KLR). Unlike the empirical risk minimization principle utilized by SVM, (K)LR yields a-posteriori probabilities of membership in each of the classes based on a maximum likelihood argument. Thus, besides predicting class labels, (K)LR additionally provides a probabilistic measure about this labeling. (K)LR has

an additional advantage that the extension to the multi-class case is well described, which must be contrasted to the commonly used coding approach. We propose a fast, accurate and stable approximate implementation of KLR for large-scale data sets. This proposed algorithm is verified both for computational complexity as generalization performance using a set of public benchmark data sets. Besides the usual hyper-parameters controlling the regularization and kernel bandwidth, an additional hyper-parameter M controlling the trade-off between accuracy and training speed (and corresponding model size) is introduced. As such, the training and final model complexity is easily manageable. The parameter puts more or less restrictions on the class of estimators beforehand. Intuitively stating more restrictions will lead to faster training. We can for instance preset M to a value such that the algorithm will finish in a reasonable amount of time. On the other hand it might be the case that this M is overestimated meaning that computer resources are wasted and evaluation speed of the resulting model is not optimal. This side-effect can be reduced by the method discussed next.

- **Chapter 5** studies an optimization scheme resulting in a sparse approximate solution of an over-determined linear system. Sparse Conjugate Direction Pursuit (SCDP) aims to construct a solution as good as possible using only a small number of non-zero (i.e. non-sparse) coefficients. The principal idea is to build up iteratively a conjugate set of vectors of increasing cardinality, in each iteration solving a small linear subsystem. By exploiting the structure of this conjugate basis, an algorithm is found (i) converging in at most D iterations for D-dimensional systems, (ii) with computational complexity close to the classical conjugate gradient, and (iii) which is especially efficient when a few iterations suffice to produce a good approximation. As an example, the application of SCDP to Fixed-Size LS-SVM (FS-LSSVM) is discussed resulting in a scheme which efficiently finds the optimal (up to an appropriate heuristic) model size (M) for the FS-LSSVM setting. While the original FS-LSSVM selects a set of Prototype Vectors (PVs) only prior to training, SCDP selects PVs during the training process. The same ideas can be applied to FS-MKLR but it was opted to first use the simpler setting of FS-LSSVM as a test case. Given a tuning-parameter set, the SCDP algorithm can efficiently compute FS-LSSVM models for an increasing range of model sizes. The final training procedure is scalable to large-scale data sets and provides sparse multi-class models with probabilistic interpretation. At the end the final algorithm is validated on several public available data sets.

- **Chapter 6** describes a framework for ASR suitable for integrating the

previously explained kernel-based methods. Instead of using frame-based features which are common in the current state-of-the-art speech recognition systems, features on the phone level are used. Our motivation to use this type of framework is two-fold: (i) a framework which can handle duration and trajectory information on the phone level has the potential to overcome limitations from the standard HMM (ii) including additional features increases the dimensionality of the feature space which can decrease performance of the traditional HMM systems while kernel-based classifiers methods have a robust mechanism which generalizes well even in high dimensional spaces. The devised segment-based framework approach divides the recognition process into two main tasks namely (i) a segmentation model which generates measures of confidence on different segmentations of the full utterance and (ii) a phone model which outputs a phone sequence hypothesis given a segmentation. Our main focus is on the latter which is particulary suited for the described kernel-based algorithms in this thesis. Although, the judgement is left to the segment model we indicate that a simple modification to the learning process of the phone model, by including information from non-speech segments (wrongly segmented speech) improves recognition performance. Phone classification as well as recognition experiments on the TIMIT benchmark corpus are presented.

1.6 Contributions of This Thesis

The main contributions of this thesis are summarized as follows:

- **Large-scale Fixed-size Multi-class Kernel Logistic Regression (FS-MKLR)**
  We presented a practical, stable and scalable implementation for kernel logistic regression which can cope with all objectives given in Section 1.4 such that this makes it a valuable candidate to be used in speech recognition architectures. By introducing an extra hyper-parameter, stating more or less restrictions on the estimation class, the final model complexity and corresponding training time is easily manageable. A spectrum of design choices was thoroughly tested and compared with state-of-the-art alternatives.

  – P. Karsmakers, K. Pelckmans, H. Van hamme, J.A.K Suykens, "Large-Scale Kernel Logistic Regression for Segment-Based Phoneme Recognition",*Internal Report 09-174*, ESAT-SISTA, K.U.Leuven, Leuven, 2009, submitted for publication.

  - P. Karsmakers, K. Pelckmans, J.A.K. Suykens, "Multi-class kernel logistic regression: a fixed-size implementation," *In Proc. of the international joint conference in neural networks (IJCNN)*, Orlando, Florida, U.S.A, pp. 1756-1761, 2007.

- **Conjugate Directions Pursuit Fixed-size Least Squares Support Vector Machines (SCDP-FSLSSVM)** We investigated a general heuristical method called Conjugate Direction Pursuit (SCDP) to find sparse approximations to symmetric positive definite linear systems and positioned this method on a conceptual level in the literature of statistics, signal processing and machine learning. As an application SCDP is applied to the LS-SVM framework resulting in a sparse LS-SVM solution which efficiently computes a range (starting from 1) of model complexities. We carefully examined the proposed algorithm in terms of scalability, type of order selection criterion (which data points to include in the model), influence of additional tuning-parameters, determination of a suitable stopping criterion and performance in the multi-class setting.

  - P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van hamme, J.A.K Suykens, "Sparse Conjugate Directions Pursuit with Application to Fixed-size Kernel Models",*Internal Report 10-63*, ESAT-SISTA, K.U.Leuven, Leuven, 2010, submitted for publication.

- **Kernel-based phone classification and recognition using segmental features**
  We successfully validated our proposed kernel-based algorithms for its use on the `timit` speech data and compared it to the state-of-the-art. We succeeded in fitting an all-at-once multi-class kernel model with probabilistic interpretation in a reasonable amount of processing time to speech data. While having comparable phone recognition accuracies the resulting model is much sparser than that obtained using the state-of-the-art SVM model. As a consequence unseen speech utterances are evaluated much faster. We introduced a computationally attractive method (without adding an additional so called garbage class) to incorporate information from non-speech data (wrongly segmented data) into final phone model. This improved the final phone recognition rate without increasing the number of parameters of the phone model. Without any language model our segment-based approach has recognition scores comparable to those of the frame-synchronous state-of-the-art HMM recognizer using a 1-gram. Since we mainly focussed on the development of this single component (the acoustical phone model) this leaves

room for improvement concerning the interactions with the other modules in the recognizer.

- P. Karsmakers, K. Pelckmans, H. Van hamme, J.A.K Suykens, "Large-Scale Kernel Logistic Regression for Segment-Based Phoneme Recognition",*Submitted for publication, Internal Report 09-174*, ESAT-SISTA, K.U.Leuven, Leuven, 2009.
- P. Karsmakers, K. Pelckmans, J. A. K. Suykens, H. Van hamme, "Fixed-Size Kernel Logistic Regression for Phoneme Classification,"*In Proc. of INTERSPEECH*, Antwerpen, Belgium, pp. 78-81, 2007.

Chapter 2

Automatic Speech Recognition

This chapter describes the state-of-the-art in automatic speech recognition. A core element in this process is the model giving a statistical relation between the symbolic speech units and the recorded acoustical signal. The current technology to accomplish this is based on Hidden Markov Models (HMMs). Impressive HMM-based recognizers have been created for a wide-variety of increasingly difficult speech recognition tasks. Still, there are a number of questionable assumptions, such as that of the first-order Markov chain, that might ultimately limit the performance of HMM based systems. We therefore resort to a different framework which uses segmental features instead. It is then indicated in which way kernel-based methods can be integrated in this segment-based ASR framework. At first, the automatic speech recognition process is formalized in Section 2.1, then the theory and potential problems of the current state-of-the-art frame-synchronous HMMs are briefly discussed in Section 2.2. Finally, in Section 2.3 a framework which can deal with segment-based features is described.

2.1 The Bayesian Recognition Paradigm

After introducing some notation we will review the formal statistical description of an automatic speech recognition process as also presented in e.g. [149], [89], [83] in this section.

Speech signals are first transformed into a sequence of vectors (called frames) $\underline{\mathrm{X}} = (x_1, \ldots, x_{N_x})$. Each frame contains features extracted from a time-window of

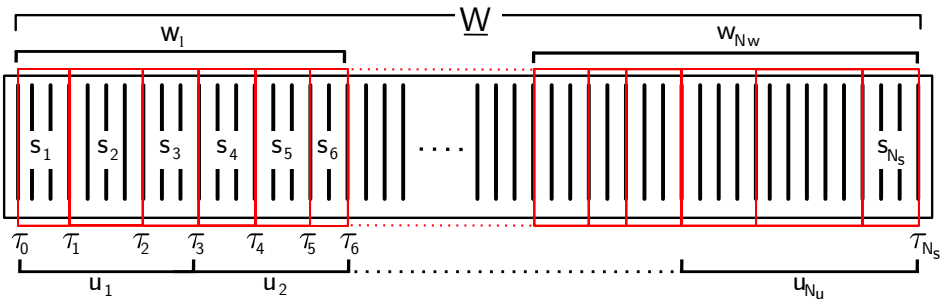


Figure 2.1: Time domain speech signals are first transformed into a sequence of vectors (called frames) $\underline{\mathrm{X}} = (x_1, \ldots, x_{N_x})$. These are presented in the figure by the black vertical lines. Each frame contains features extracted from a time-window of speech (e.g. 30 ms long). Now, instead of recognizing the whole speech utterance at once (which is untractable in case of a large vocabulary) the process is decomposed into smaller problems. Typically, a recognizer identifies a sequence of phones (smallest distinguishable sounds) $\underline{\mathrm{U}} = (u_1, \ldots, u_{N_u})$ which are later combined to words, and sentences denoted as $\underline{\mathrm{W}} = (w_1, \ldots, w_{N_w})$. A segment s_i is defined as a group of frames. A sequence of segments is denoted as $\underline{\mathrm{S}} = (s_1, \ldots, s_{N_s})$ identifying a segmentation of the full frame sequence $\underline{\mathrm{X}}$. The i-th segment respectively starts and ends in a frame with an index τ_{i-1} and $\tau_i - 1$. Depending on the approach a unit is linked to one or more segments. A certain sequence of units can be identified by several different segmentations.

speech (e.g. 30 ms). Now, instead of recognizing the whole speech utterance at once (which is untractable in case of a large vocabulary) the process is decomposed into smaller problems. Typically, a recognizer identifies a sequence of speech units which are later combined to words, and sentences denoted as $\underline{\mathrm{W}} = (w_1, \ldots, w_{N_w})$. These sub-words are taken from an alphabet of basic units represented as $\mathcal{U}' = \{u'_1, \ldots, u'_C\}$, a sequence of basic units is noted as $\underline{\mathrm{U}} \in \mathcal{U}'^{N_u}, \underline{\mathrm{U}} = (u_1, \ldots, u_{N_u})$. A segment s_i is defined as a group of frames. A sequence of segments is denoted as $\underline{\mathrm{S}} \in \mathcal{S}, \underline{\mathrm{S}} = (s_1, \ldots, s_{N_s})$ (with $\mathcal{S}$ the set of all possible segmentations of $\underline{\mathrm{X}}$) identifying a segmentation of the full frame sequence $\underline{\mathrm{X}}$. The i-th segment respectively starts and ends in a frame with an index τ_{i-1} and $\tau_i - 1$. Depending on the approach a unit is linked to one or more segments. A certain sequence of units can be identified by several different segmentations. For notational convenience we skip the random variables in the notation for probability. Our notation is schematically summarized in Fig. 2.1.

The acoustic preprocessing module (see Fig. 1.1) transforms the speech utterance

into a sequence of frames $\underline{\mathrm{X}}$. The goal is then to search for the most likely word sequence hypothesis $\hat{\underline{\mathrm{W}}}$ given this vector sequence

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}|\underline{\mathrm{X}}). \tag{2.1}$$

In case of a large dictionary, recognition is typically performed on sub-words. In order to express this formally an additional random variable $\underline{\mathrm{U}}$ is introduced. Here, $\underline{\mathrm{U}}$ represents the sequence of basic speech units. The ASR system then calculates the probability of all word, and unit sequences given a set of frames. The optimal estimate $\hat{\underline{\mathrm{W}}}$ is chosen as the recognition result

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}|\underline{\mathrm{X}}), \tag{2.2}$$

where the sum over $\underline{\mathrm{U}}$ is only necessary in case of using multiple word pronunciations. Most often this sum is replaced by a maximum, hence using a single best pronunciation variant for each sentence hypothesis. By applying Bayes' rule equation (2.2) can be decomposed as

$$\begin{aligned}\hat{\underline{\mathrm{W}}} &= \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \frac{\Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}, \underline{\mathrm{X}})}{\Pr(\underline{\mathrm{X}})} \\ &\propto \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}) \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}}) \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}})\end{aligned} \tag{2.3}$$

where $\Pr(\underline{\mathrm{X}})$ can be ignored in this maximization since it does not depend on $\underline{\mathrm{W}}$, nor $\underline{\mathrm{U}}$. It is also assumed that the basic units used are detailed enough such that all the information in $\underline{\mathrm{W}}$ which is relevant in determining a probability score on $\underline{\mathrm{X}}$ is covered in $\underline{\mathrm{U}}$ ($\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{W}}) = \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}})$).

Equation (2.3) clearly decomposes speech recognition into acoustic modeling $\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}})$ and language modeling $\Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}})$ and $\Pr(\underline{\mathrm{W}})$. In Bayesian terminology, $\Pr(\underline{\mathrm{W}})$ is the prior probability of $\underline{\mathrm{W}}$, $\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}})$ is the likelihood of the data given the unit string $\underline{\mathrm{U}}$, $\Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}})$ describes the lexicon, and (2.3) maximizes the a-posteriori probability of $\underline{\mathrm{W}}$.

Remark 4. Due to the decomposition of the full learning problem into smaller tasks, the acoustic models for the basic units u_i' can be used in many different sequences $\underline{\mathrm{U}}$. Therefore the acoustic modeling can only use local information regarding the unit. This explains why longer units (e.g., syllables) can exploit more context information than shorter models.

2.2 Frame-synchronous HMM Speech Recognition

A common speech recognizer decodes speech frame-synchronous. This means that the ASR model evaluates the frames which individually describe very short time-windows of speech (e.g 30 ms each) separately. This evaluation occurs in the acoustic modeling part of the recognizer. Hidden Markov Models (HMMs) have been the de facto standard for acoustical modeling in ASR since the mid eighties [150]. In this section we give a short overview mainly based on the notes in textbook [83] of the main properties of these state-of-the-art frame-synchronous HMM systems.

2.2.1 The acoustical model as an HMM

HMMs describe a speech utterance by a sequence of internal states $\underline{\mathrm{Q}} = (q_1, \ldots, q_{N_q})$ which cannot be observed (*hidden*) directly, only an output dependent on the state q_i is visible. The first internal state is defined as q_1. We already defined a sequence of input vectors $\underline{\mathrm{X}}$. In HMMs the distribution of $x_i \in \underline{\mathrm{X}}$ depends on q_i. However, the initial state q_1 has no corresponding observation. The different states are linked to each other using some predefined topology. In most cases acoustic models are built using a first order Markov chain which means that for all $t = 1, \ldots, N_q$ one has

$$\Pr(q_{t+1}|q_1, \ldots, q_t) = \Pr(q_{t+1}|q_t). \tag{2.4}$$

Only the present state determines the next, this means that no context information from the past is used. The probabilities $\Pr(q_{t+1}|q_t)$ in (2.4) are called transition probabilities which can be stored in a matrix $A \in \mathbb{R}^{N_q \times N_q}$ defined as $A_{ij} = \Pr(q_{t+1} = j|q_t = i), \forall i, j$. When setting some of the A_{ij} to zero changes the topology of the Markov chain. The initial state distribution at the beginning of the Markov chain is defined for all $i = 1, \ldots, N_q$ as

$$\pi_i = \Pr(q_1 = i). \tag{2.5}$$

As previously mentioned a frame-synchronous setup is used. This means that the distribution of $\underline{\mathrm{X}}$ only depends on the current internal state q_t and is called the observation (emission) probability distribution for state i and is defined for all $t = 1, \ldots, N_x$, $i = 1, \ldots, N_q$ as

$$b_i(x_t) = p(x_t|q_t = i), \tag{2.6}$$

where p is a Probability Density Function (PDF) and x_t a frame of $\underline{\mathrm{X}}$ at time t. Different types of observation probabilities are used for this purpose which can be divided into two categories:

- Discrete densities based on vector quantization (e.g. [124], [130]).
- Continuous density functions which are mostly based on a mixture of parametric PDFs.

Although discrete HMMs offer excellent computational speed they are rarely used when the goal is to obtain extremely high recognition performance because of the quantization noise introduced by the vector quantization process. For that reason, more accurate Gaussian Mixture Models (GMM) which fit into the class of continuous density functions is favored in most current state-of-the-art systems. In a Gaussian mixture, the probability density is obtained as a weighted sum of M_i Gaussian distributions, i.e.,

$$b_i(x) = \sum_{c=1}^{M_i} w_{ci}\mathcal{N}(x;\mu_{ci},\Sigma_{ci}) \text{ such that } \sum_{c=1}^{M_i} w_{ci} = 1, w_{ci} \geq 0, \tag{2.7}$$

where M_i is the number of mixture components for i-th emission probability function, w_{ci} the weight for the c-th Gaussian and i-th emission probability function and the multivariate normal distribution

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{2.8}$$

described by a mean vector μ and covariance matrix Σ. The parameter set of the above state-dependent distribution is $\eta_i = \{w_{ci},\mu_{ci},\Sigma_{ci}\}_{c=1}^{M_i}$. In order to limit the number of parameters that need to be estimated, diagonal covariance matrices are often used. GMMs are expressive but parameter inference is non-convex and cannot be written analytically. Instead, they are often trained iteratively using an Expectation-Maximization (EM) approach [40].

Remark 5. In this thesis, new approaches will be compared to the state-of-the-art large vocabulary HMM speech recognition system developed at the ESAT/PSI/SPRAAK speech laboratory of the university [45], [46], [41], [42]. This system uses semi-continuous [82] HMMs. The difference with standard GMM/HMMs is that the individual Gaussians are shared by all HMM states, or for all $i = 1,\ldots,N_q$,

$$b_i(x) = \sum_{c=1}^{M_{\text{tot}}} w_{ci}\mathcal{N}(x;\mu_c,\Sigma_c) \text{ such that } \sum_{c=1}^{M_{\text{tot}}} w_{ci} = 1, w_{ci} \geq 0, \tag{2.9}$$

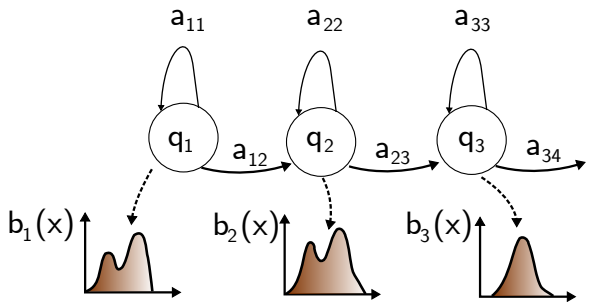


Figure 2.2: A 3-state hidden Markov model with left-to-right structure.

where M_{tot} is the total number of Gaussians and w_{ci} shows that each of the states has a different weight vector for all Gaussians. Such semi-continuous HMMs require in general a smaller number of Gaussian basis functions in order to achieve similar performances.

In speech recognition one typically uses the HMM model setup as given in Fig. 2.2. Here a 3-state, left-to-right HMM [83] with continuous density functions is shown. For this topology the matrix A is sparse, $A_{ij} \neq 0$ for $j = i$ and $j = i + 1$. Such a left-to-right HMM gets a natural interpretation as representing successive parts of the model. The self-loop probability A_{ii} then takes the role of duration model. The resulting duration model is due to the first order Markov property memory-less which means that the model cannot express a certain HMM state is likely to last for a number of frames, and that both shorter and longer periods are less likely.

2.2.2 Context-dependent phone models

As seen in previous sections the observation probability distributions from (2.6) depend only on the current HMM state which means that HMMs model speech as a sequence of stationary segments. Very often, acoustic phone models are 3-state left-to-right HMMs, such as the one in Fig. 2.2. In such a model, the assumption is made that the phone consists of three stationary parts. However, in continuous speech recognition dynamic effects occur frequently. The mechanic inertia of the articulators causes a strong interdependence of neighboring sounds, known as coarticulation. Consequently, state-of-the-art HMM recognizers condition their individual phone models on the neighboring sounds. This is known as context dependent phone models, e.g. a 3-phone [115] is a phone model where the HMM parameters depend both on the previous and the next phone.

2.2.3 HMM training and decoding

Training

Given the HMM specifications the parameters involved need to be trained from the available data. Two main training methods emerge

- **Generative training (maximum likelihood):** By maximizing the (log) likelihood of a signal observation $\underline{\mathrm{X}}$ given the model structure, an estimate of the HMM parameters $\theta = (\pi, A, \eta)$ can be inferred. Formally this can be written as

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \log(\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \theta)) \\ &= \arg\max_{\theta} \log \sum_{\underline{\mathrm{Q}}} (\Pr(\underline{\mathrm{X}}, \underline{\mathrm{Q}}|\underline{\mathrm{U}}, \theta)). \end{aligned} \tag{2.10}$$

  To solve (2.10), one typically resort to the simple and elegant iterative procedure for the estimation of the HMM parameters known as the Expectation Maximization (EM) algorithm [40]. Direct application of EM to HMMs results in the Baum-Welch training algorithm. The details are given in [6]. In practical implementations one typically uses the Viterbi approximation. Viterbi training is similar to the Baum-Welch algorithm, but instead of considering all alignment paths, only the most likely alignment given the HMM parameters from a previous iteration is used to compute a new set of parameters.

- **Discriminative training:** HMMs assume that a frame is independent of the previous frame given the current state. This is obviously a false assumption since successive frames are extracted from windowed speech segments that are either overlapping or at most some milliseconds apart. Due to this incorrect model assumption and because the training material is always sparse, even in large data sets, the class-conditional probability distributions estimated using a generative model may be far from the true unknown distributions. Better recognition performance may be achieved by using different criteria which optimize all the models jointly and is more related to the ultimate goal of automatic speech recognition, namely the classification rate (see also discussion in Section 1.2.3). Such methods are called discriminative training methods and include both the Maximum Mutual Information (MMI) method [7] and the Minimum Classification Error (MCE)

method [91]. The textbook [79] is a good reference work. There the authors demonstrate that the MCE training approach significantly improves upon the maximum likelihood approach and can effectively applied not only to small-scale tasks but as well to large-scale ASR tasks. However, the training objective is non-convex which can lead to global convergence difficulties.

Decoding

During recognition the goal is to search for the most likely word sequence $\hat{\underline{\mathrm{W}}}$ or state sequence $\hat{\underline{\mathrm{Q}}}$ given a sequence of input vectors $\underline{\mathrm{X}}$. Both the state alignment (which additionally defines a segmentation of the utterance as in Fig. 2.1) and unit labeling has to be performed. A hypothesis is judged by the following criterion

$$\Pr(\underline{\mathrm{X}}, \underline{\mathrm{Q}}|\underline{\mathrm{U}}) = \pi_{q_1} \prod_{i=2}^{N_q} A_{q_{i-1}q_i} b_{q_i}(x_{i-1}). \tag{2.11}$$

Marginalizing out q gives

$$\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}) = \sum_{\underline{\mathrm{Q}}} \Pr(\underline{\mathrm{X}}, \underline{\mathrm{Q}}|\underline{\mathrm{U}}), \tag{2.12}$$

the forward pass algorithm. The likelihood of the input sequence given the HMM models is the sum of all likelihoods of all possible state alignments of the input data. Using the first order Markov chain assumption a dynamic programming algorithm can be defined to compute the utterance score. Still, the forward pass algorithm can be a computational burden. It is therefore that in most current state-of-the-art ASR systems the Viterbi algorithm is used which approximates (2.12) by

$$\Pr(\underline{\mathrm{X}}|\underline{\mathrm{W}}) = \max_{\underline{\mathrm{Q}}} \Pr(\underline{\mathrm{X}}, \underline{\mathrm{Q}}|\underline{\mathrm{U}}). \tag{2.13}$$

It has been shown that this approximation has no significant effect on the recognition accuracy [43]. To further improve the efficiency of the algorithm, pruning techniques are available to reduce the search space during calculation, see [44] for more details.

2.2.4 HMM assumptions and problems

The first order Markov chain assumption allows the use of efficient training and evaluation algorithms. Still, there are a number of questionable assumptions that

might ultimately limit the performance of HMM based systems. In [44], [83] three limitations of the standard HMM formalism are enumerated which we will briefly review here:

- **Duration modeling:** The Markov assumption combined with the use of self-loops gives a memoryless duration modeling, the model cannot express that the state is likely to last for a number of frames, and that both shorter and longer periods are less likely. Several extensions to the original HMM were proposed to address this problem such as: methods which use Bakis topology HMMs (i.e., left-to-right models with both self-loops and skips) with the number of states proportional to the average duration of the phone or methods which break the first order Markov assumption in order to model the duration.
- **Trajectory modeling:** Trajectory modeling tracks the dynamics of a speech signal. In standard HMMs, the probability of successive observations is assumed independent given the state sequence, i.e., the emission probability distribution in (2.6) depends only on the current state, and not on other values of $\underline{X}$. Consequently, the temporal (inter-frame) correlations are poorly modeled. Potential modeling limitations are:
  - In Fig. 2.3 two equally likely paths are shown given a three-state HMM denoted by three sets of concentric ellipsoids representing the observation distributions. Each concentric circle denotes a certain probability. It is clear that the path represented by the triangles is more natural, but the HMM is unable to make a distinction.
  - In addition the HMM is unable to judge whether or not the path follows a trajectory that resembles the ones it was trained on. Fig. 2.4 shows a path and two HMMs (dashed and dotted) that assign an equal likelihood to the shown path. Suppose that the dotted HMM was trained on a number of approximately straight (horizontal in the figure) trajectories, while the dashed HMM was trained on a number of U-shaped trajectories. That information is lost in the process of estimating the three states' distributions, and hence the path gets an equal likelihood from both models, while it clearly resembles the training data of the dashed HMM.
- **Coarticulation modeling:** Context dependent model variations (e.g. combination of phones) need to be predefined during training. On the one hand, since all possibilities are needed, the context span is best kept short. On the other hand, to model long-term influences longer context spans are required.

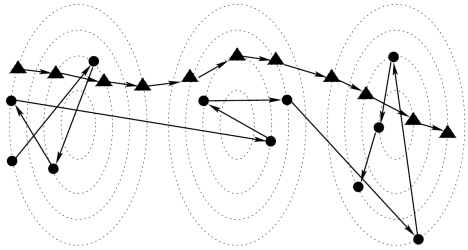


Figure 2.3: Two equally likely paths according to an HMM with 3 states. The figure shows a two-dimensional input space, in which the observation distributions of the HMM states are drawn as concentric ellipsoids. Each concentric circle denotes a certain probability. Figure was taken from [44]

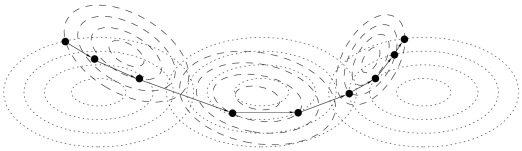


Figure 2.4: Two equally likely HMMs for a path. The figure shows a two-dimensional input space. The two sets (dashed and dotted) of concentric ellipsoids indicate the observation distributions of two different three-state HMMs. Each concentric circle denotes a certain probability. Figure was taken from [44]

2.3 Segment-based Speech Recognition

In this thesis we aim at integrating kernel-based methods in an ASR framework. Kernel-based alternatives are known to generalize well even in high dimensional spaces [159]. Besides the usual purely acoustical features also trajectory and duration related information might be incorporated. However, as explained in Section 2.2, features which describe dependencies between different frames such as the latter two, cannot be used in the standard frame-synchronous HMM setup. We therefore resort to a segment-based setup which has the potential to overcome standard HMM limitations. In the previous section the set $\underline{\mathrm{Q}}$ expressed a sequence of HMM states, defining a segmentation on $\underline{\mathrm{X}}$. In that case a unit might be linked to more than one segment. In this section a certain segment s_i from $\underline{\mathrm{S}}$ with start- and end-time $(\tau_{i-1}, \tau_i - 1)$ is associated to a unit u_i from $\underline{\mathrm{U}}$.

Segment-based speech recognizers assign an observation likelihood based on features extracted from a subset of frames from the full sequence $\underline{\mathrm{X}}$. Such features are denoted as $f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})$ with $\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1} = (x_{\tau_{i-1}}, \ldots, x_{\tau_i-1})$. Although this type of

recognizer is not used as frequently as its frame-synchronous alternative, it has the potential advantage that it can include information of the correlation that exists among the frames in the segment including its surroundings (trajectory modeling) and can perform duration modeling. Since the segment scores are dependent on the segmentation, segment-based recognizers have to examine all possible segmentations and corresponding units of a speech utterance which can be a challenge. The idea of segment-based modeling seems to have arisen around the early nineties in several different research groups [4], [114], [139]. Although theoretically promising, up till now only minor improvements over the frame-synchronous HMM system are obtained, while training is computational more demanding.

In order to formally define the segment-based speech recognition process the random variable $\underline{\mathrm{S}}$ is introduced in (2.2) which gives

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}, \underline{\mathrm{S}}|\underline{\mathrm{X}}), \tag{2.14}$$

Note again that these sums can be replaced by a maximum, hence using a single pronunciation variant with a single best segmentation for each sentence hypothesis. By applying Bayes' rule equation (2.14) can be decomposed as

$$\hat{\underline{\mathrm{W}}} = \arg\max_{\underline{\mathrm{W}}} \sum_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \frac{\Pr(\underline{\mathrm{W}}, \underline{\mathrm{U}}, \underline{\mathrm{S}}, \underline{\mathrm{X}})}{\Pr(\underline{\mathrm{X}})} \tag{2.15}$$

$$\propto \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}) \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}}) \sum_{\underline{\mathrm{S}}} \Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}) \tag{2.16}$$

where the constant factor $\Pr(\underline{\mathrm{X}})$ can again be ignored in this maximization since it does not depend on $\underline{\mathrm{W}}, \underline{\mathrm{U}}$ nor $\underline{\mathrm{S}}$. It is also assumed that the basic units used are detailed enough such that all the information in $\underline{\mathrm{W}}$ which is relevant in determining a probability score on $\underline{\mathrm{S}}$ and $\underline{\mathrm{X}}$ is covered in $\underline{\mathrm{U}}$ ($\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{W}}) = \Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}})$).

By considering factor $\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}})$ in (2.16) segment-based recognizers can be divided into two broad classes [139]:

- Those using the conditional segment modeling approach, which use the decomposition $\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}) = \Pr(\underline{\mathrm{S}}|\underline{\mathrm{U}}) \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}})$. The main factor in this decomposition is the likelihood $\Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}})$. Examples which implement such an approach are found in [206], [138], [69].
- Those using the a-posteriori segment modeling approach, which use the decomposition

$$\Pr(\underline{\mathrm{S}}, \underline{\mathrm{X}}|\underline{\mathrm{U}}) = \frac{\Pr(\underline{\mathrm{X}}) \Pr(\underline{\mathrm{U}}, \underline{\mathrm{S}}|\underline{\mathrm{X}})}{\Pr(\underline{\mathrm{U}})}, \tag{2.17}$$

where we again can ignore the constant factor $\Pr(\underline{X})$. The main factor in this decomposition is the joint a-posteriori probability $\Pr(\underline{U}, \underline{S}|\underline{X})$. Examples which utilize this formalism are [114], [4], [126].

Since the kernel-based classifiers which we devise in this thesis are based on a discriminative approach, they might be used to define a model for the a-posteriori probabilities used in the latter class. Substituting (2.17) into (2.16) results in

$$\hat{\underline{W}} \propto \arg\max_{\underline{W}} \Pr(\underline{W}) \sum_{\underline{U}} \Pr(\underline{U}|\underline{W}) \sum_{\underline{S}} \frac{\Pr(\underline{S}, \underline{U}|\underline{X})}{\Pr(\underline{U})}. \tag{2.18}$$

The joint probability term $\Pr(\underline{S}, \underline{U}|\underline{X})$ which can be found in (2.18) can be decomposed into a unit and segmentation probability factor,

$$\Pr(\underline{S}, \underline{U}|\underline{X}) = \Pr(\underline{U}|\underline{S}, \underline{X}) \Pr(\underline{S}|\underline{X}). \tag{2.19}$$

This factorization has some advantages:

- Dedicated statistical models can be trained for estimating the segmentation and the unit probabilities. Only the segmentation model has to be trained on all possible segmentations which can yield a reduction in training time.
- The computational requirements of the recognition process can be reduced considerably by calculating the unit probabilities only for those candidate segmentations having a segmentation probability exceeding a predefined threshold.

Instead of considering features on the full utterance level, the modeling assumption is made to compute features based on smaller units. Adjacent frames, especially those within the same phonetic segment, often exhibit smooth transitions and are highly correlated, violating the conditional independence assumption as explained earlier. However, the relationship between features computed in different phonetic segments is weaker. We therefore extract features on the phone segment level. By applying this to (2.19) we have,

$$\Pr(\underline{S}, \underline{U}|\underline{X}) \equiv \Pr(\underline{U}|f_{\mathrm{s}}(\underline{X}, \underline{S})) \Pr(\underline{S}|\underline{X}). \tag{2.20}$$

Note that using segmental features, one has to address the conditioning event mismatch problem that is mentioned in [139] [185] which we briefly review here. Considering (2.20) it is seen that the overall conditioning event $f_{\mathrm{s}}(\underline{X}, \underline{S})$ depends on the segmentation (different segmentations give different features). However,

many practical recognition systems use the Viterbi search algorithm. In this case the probability estimate of a word sequence is conditioned on the sequence of segmental feature vectors $f_s(\underline{X}, \underline{S})$ corresponding with a single segmentation $\underline{S}$. The recognition process will thus compare probabilities of phone sequences with different segmentations and thus a different conditioning part $f_s(\underline{X}, \underline{S})$. Consequently, the set of segmentations that is used for calculating the probability of a certain word sequence does not necessarily coincidence with the set that is used for another word sequence. However, one can expect that there is a large overlap in the sets of segmental feature vectors that are used for calculating the probability estimates for the top competing candidate word sequences. In [185] it was experimentally verified that using the full forward search only has a minor improvement on the final recognition rate compared to a computationally more efficient Viterbi search.

In Chapter 6 we implement the segmental-based approach in combination with the kernel-based methods devised in this thesis and compare to a state-of-the-art frame-synchronous HMM system.

2.4 Summary

In this chapter we briefly reviewed the HMM technology which is the core element in current state-of-the-art ASR systems. Although HMMs have shown impressive results both in training speed and recognition accuracy still some modeling problems such as modeling duration, trajectory and coarticulation occur. As an attempt to overcome the first two problems segment-based models which operate on segmental features, extracted from a larger (than the usual 30ms window) parts of the speech utterance (e.g. phone), were discussed. For now these approaches only led to minor improvements while being computational more demanding. However, the kernel-based algorithms we devise in this thesis have the potential to increase performance of such systems since kernel-based methods are based on well-defined training algorithms and are known to generalize well even in high dimensional spaces. The latter allows one to incorporate additional features to better describe each of the segments such as those which describe the segmental duration and trajectory information. Since the kernel methods we apply are discriminative, and therefore are naturally more suited to either directly or indirectly generate a-posteriori probability outcomes, we opt to use the a-posteriori segment-based speech recognition paradigm. Later on, in Chapter 6 we apply on this segment-based framework in combination with the kernel-based methods.

Chapter 3

Kernel-based Methods

In this section a brief overview is given of the kernel methods which are used in this thesis for learning acoustical models in an ASR architecture. While having superior performances in terms of generalization performances on a wide range of applications, such methods are not yet frequently used in speech recognition mainly because they are often too computational demanding. Although efficient training methods exist, in case of large-scale problems, as are common in speech recognition, solving the full optimization problem can even become untractable. We therefore describe some typical techniques to approximate the optimization problems in order to achieve realistic model training and evaluation processing times. In order to facilitate the integration into a speech architecture some general extensions to the original frameworks are given. In Section 3.1 we briefly explain what a kernel method is. We then give details about Support Vector Machines, Least Squares Support Vector Machines and Kernel Logistic Regression in Sections 3.2, 3.3, 3.4 respectively. Training a kernel-based model involves additional hyper-parameters. Our specific approach is given in Section 3.5. Not all kernel-based methods give model outcomes with probabilistic interpretation. In Section 3.6 a simple approach is given to overcome this limitation. In Section 3.7 we describe in which manner binary classifiers can be extended to the multi-class case. Finally in Section 3.8 a relative cheap technique to incorporate prior knowledge learned from universum (background signals) data into the model is presented.

3.1 Kernel-based learning

The machine learning algorithms we focus on in this thesis belong to the kernel-based classifier category. They all rely on optimizing a convex objective function which learns linear decision functions in a feature space. In order to enable non-linear decision surfaces in the original input space, a kernel function is used which maps the input vectors $x \in \mathbb{R}^D$ via a feature map $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ to a higher dimensional space. In this feature space the classes are then linearly separated. As will be shown next, the corresponding optimization objective and evaluation function of the resulting model can be written in terms of dot-products on the transformed input vectors. This implies that instead of defining the feature map explicitly, a kernel function $K(x, x')$, which represents a dot-product $K(x, x') = \varphi(x)^T \varphi(x')$ in a corresponding feature space, can be used instead. Hence, complex non-linear classifiers are obtained in the input space using well-defined learning algorithms designed to discover linear patterns in the feature space. Any valid kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ corresponds with an inner product in a corresponding feature space as long as the function K is positive semi-definite. This classical result was introduced in [5]. Although, many kernels are available which satisfy this property, and a specific choice of kernel function can improve performance for the problem at hand, popular choices are the following static [1] kernels

- linear kernel: $K(x, x') = x^T x'$
- polynomial kernel: $K(x, x') = (c + x^T x')^d$, $c \geq 0, d > 0$
- Radial Basis Function (RBF) kernel with $\sigma > 0$ termed the bandwidth: $K(x, x') = \exp\left(-||x - x'||_2^2/\sigma^2\right)$

3.2 Support Vector Machines

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be samples from the random vector $(\mathbf{X}, \mathbf{Y})$ such that $x_i \in \mathcal{X} \subset \mathbb{R}^D$ and $y_i \in \{-1, 1\}$. Let us consider the hyperplane described as

$$\text{Hp}(w) = \{f(x) = w^T \varphi(x) + b = 0 | \varphi(x) \in \mathbb{R}^{D_\varphi}\}, \tag{3.1}$$

[1] Keyword "static" refers to kernels which operate on fixed-length vectors. This must be contrasted to e.g. sequence kernels which operate on varying-length vector sequences.

where $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ is a fixed but unknown mapping. Let $\mathcal{F} = \{\text{Hp}(w), w \in \mathbb{R}^{D_\varphi}\}$ be the class of hyperplanes which give the same labeling on the training data. The prediction of the label corresponding to a new point x^* with respect to the hyperplane $\text{Hp}(w)$ can be determined as follows

$$\hat{y}^* = \text{sign}\left(f(x^*)\right) = \text{sign}\left(w^T\varphi(x^*) + b\right). \tag{3.2}$$

The distance of any point $\varphi(x^*)$ to the hyperplane (the margin) $\text{Hp}(w) \in \mathcal{F}$ is given as

$$d\left(\varphi(x^*), \text{Hp}(w)\right) = \frac{|f(x^*)|}{||f'(x^*)||_2} \geq \frac{y^*(w^T\varphi(x^*) + b)}{\sqrt{w^Tw}}. \tag{3.3}$$

In [184] the authors propose to choose the separating hyperplane that maximizes the margin, that is to say the hyperplane that maximizes the distance of the hyperplane and the closest example. This can be written as follows

$$\max_{w,b,m} m \quad \text{such that } d(\varphi(x_i), \text{Hp}(w)) \geq m, \forall i = 1, \ldots, N. \tag{3.4}$$

Without loss of generality one can scale variables such that $\sqrt{w^Tw} = \frac{1}{m}$. As such, one can rewrite optimization problem (3.4) to

$$\min_{w,b} \sqrt{w^Tw} \quad \text{such that } y_i\left(w^T\varphi(x_i) + b\right) \geq 1, \forall i = 1, \ldots, N.$$

which is equivalent to minimizing

$$\min_{w,b} \frac{1}{2}w^Tw \quad \text{such that } y_i\left(w^T\varphi(x_i) + b\right) \geq 1, \forall i = 1, \ldots, N. \tag{3.5}$$

Minimizing (3.5) corresponds to maximizing the margin subject to the constraint that the labels predicted for the samples equal the observed labels. This is denoted as the separable case.

A relaxation was formulated to the case where the data of the different classes is not strictly separable by an hyperplane from the class $\mathcal{F}$. After introducing slack variables $\xi = (\xi_1, \ldots, \xi_N)^T$, one can write

$$\min_{w,b,\xi} \frac{1}{2}w^Tw + C\sum_{i=1}^{N}\xi_i$$

$$\text{such that } \begin{cases} y_i\left(w^T\varphi(x_i) + b\right) \geq 1 - \xi_i, \forall i = 1, \ldots, N \\ \xi_i \geq 0, \forall i = 1, \ldots, N. \end{cases} \tag{3.6}$$

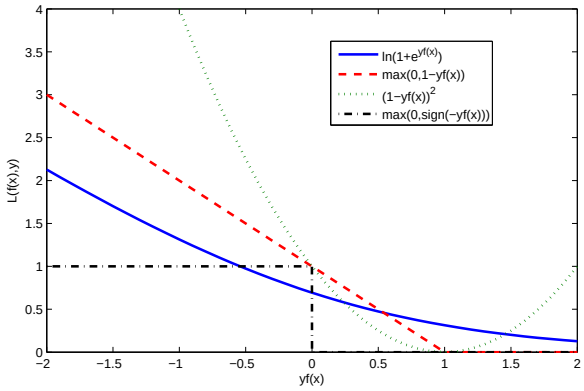


Figure 3.1: Comparison of different loss functions for binary classification with true outputs $y = \pm 1$ and predictions $f(x)$.

Remark 6. The different algorithms that we will explore in this section follow the general objective function:

$$\min_{w,b} \frac{1}{2} w^T w + \varrho \sum_{i=1}^{N} L(f(x_i), y_i), \tag{3.7}$$

where $L(f(x), y)$ is the loss function and ϱ is a positive real constant. In case the constrained problem in (3.6) is rewritten in the general form of the unconstrained optimization problem of (3.7), the loss function is defined as $L(f(x), y) = \max(0, 1 - yf(x))$ and $\varrho = C$. This is the so called hinge loss function which is plotted in Fig. 3.1.

The dual formulation is summarized by the following lemma.

Lemma 1. *[184] Given a positive definite kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ with $K(x, x') = \varphi(x)^T \varphi(x')$ and a regularization constant $C \in \mathbb{R}^+$, the solution to (3.6) is given as the following dual problem:*

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{N} y_i y_j K(x_i, x_j) \alpha_i \alpha_j + \sum_{i=1}^{N} \alpha_i$$

$$\text{such that } \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, N. \end{cases} \tag{3.8}$$

Proof. The Lagrangian of the problem (3.6) is

$$\mathcal{L}(w,b,\xi;\alpha,\nu) = \frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i(w^T\varphi(x_i)+b) - 1 + \xi_i\right) - \sum_{i=1}^{N}\nu_i\xi_i. \tag{3.9}$$

where the $\alpha_i \geq 0$ and $\nu_i \geq 0$ values are the Lagrange multipliers. The solution is given by the saddle point of the lagrangian:

$$\max_{\alpha,\nu}\min_{w,b,\xi}\mathcal{L}(w,b,\xi;\alpha,\nu). \tag{3.10}$$

One obtains

$$\begin{cases} \frac{\partial\mathcal{L}}{\partial w} = 0 & \to w = \sum_{i=1}^{N}\alpha_i y_i\varphi(x_i) \\ \frac{\partial\mathcal{L}}{\partial b} = 0 & \to \sum_{i=1}^{N}\alpha_i y_i = 0 \\ \frac{\partial\mathcal{L}}{\partial\xi_i} = 0 & \to 0 \leq \alpha_i \leq C;\ \forall i = 1,\ldots,N. \end{cases} \tag{3.11}$$

Writing the problem in terms of α_i and applying the so called kernel trick $K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j)$ leads to the convex Quadratic Program (QP) problem in (3.8). □

Due to the truncation property of the hinge loss only the data points lying on the wrong side of the classification boundary, and those on the right side but near the boundary have an influence in determining the position of the boundary, and hence have corresponding non-zero α_i's. The data points (which are represented as vectors) linked to these non-zero α_i's are called Support Vectors (SVs). In the resulting classifier the sum should be taken only over the non-zero α_i values instead of all training data points

$$\hat{y}^* = \text{sign}\left(\sum_{i=1}^{N_{\text{sv}}}\alpha_i y_i K(x,x_i) + b\right), \tag{3.12}$$

where index i runs now over the number of SVs denoted as N_{sv}.

In statistical learning theory a relation between the empirical and theoretical risk has been established. The following summarizes a main result [184]

Theorem 2. *[184] Let $0 < \eta \ll 1$ be fixed and $\mathcal{D} = \{(x_i,y_i)\}_{i=1}^N \subset \mathbb{R}^D \times \{-1,1\}^N$ be sampled i.i.d. from a fixed but unknown density function $p(\boldsymbol{X},\boldsymbol{Y})$. Let the theoretical risk or generalization performance of a classifier be defined as*

$$\text{R}(f) = \int I(yf(x) < 0)p(\boldsymbol{X},\boldsymbol{Y})d\boldsymbol{X}d\boldsymbol{Y}, \tag{3.13}$$

where $I(yf(x) < 0)$ equals one if $yf(x) < 0$ and zero otherwise. The empirical risk is defined as

$$\hat{\mathrm{R}}(f) = \frac{1}{N}\sum_{i=1}^{N} I\left(y_i f(x_i) \leq 0\right). \tag{3.14}$$

Let $\mathcal{F}$ be a set of different hyperplanes with VC-dimension h'

$$\mathrm{R}(f) \leq \hat{\mathrm{R}}(f) + \sqrt{\frac{h'\,\log(2N/h'+1) - \log(\eta/4)}{N}} \tag{3.15}$$

holds with probability $1-\eta$.

3.2.1 Large-scale implementation

Training a SVM model typically involves solving a convex QP problem. Since large-scale problems are associated to a kernel matrix which is too large to be stored in memory, customized decomposition methods were designed to handle such cases. A particular case of such method is iterative chunking where the overall problem is decomposed into small problems restricted to a small subset of training examples called the active set. An extreme form of chunking, where only two elements are selected per iteration, is Sequential Minimal Optimization (SMO) in [145]. This choice is the current most popular solver for medium to large-scale SVM problems. To perform SVM experiments we utilize the state-of-the-art LIBSVM toolbox [117] which uses the SMO solver described in [57]. The computation cost of SVM is $\mathcal{O}(NN_{\mathrm{sv}} + N_{\mathrm{sv}}^3)$ [30]. However, for large-scale data set sizes ($N > 50,000$) approximative techniques are needed. These are discussed next.

3.2.2 Sparseness

Approximating the original problem can lead to faster training and even sparser models. Many approximative methods exists which further reduce the number of non-zero α_i's in the SVM solution, while attaining similar accuracies. Increasing the sparsity of the solution has the advantage of faster evaluation. A first attempt would be to perform a postprocessing step to further increase the sparsity pattern of the SVM solution (e.g. [19], [50]) but since an expensive full SVM training is still required we do not consider this approach here.

Other algorithms focus on approximatively solving the primal SVM objective (3.6). In [30] it is seen that even in the non-linear setting, it is feasible to solve the

primal problem (3.6) and it is claimed that when the goal is to find an approximate SVM solution, the primal optimization is superior since it better describes the actual intended learning goal. We therefore restrict ourselves to methods which approximate the primal SVM objective. From the conditions for optimality it can be seen that w can be expressed as a linear combination of the training samples $w = \sum_{i=1}^{N} \beta_i \varphi(x_i)$. If we fill this in in (3.6) and solve for β_i (assuming uniqueness of the solution), then $\beta_i = \alpha_i y_i$. Note that the latter equation is only satisfied in case of exact optimality. In case a suboptimal primal solution is aimed for, the connection between the β_i and α_i is lost. Some examples include:

- In [137] sparsity on the β_i's is achieved by posing a L_1 regularizer on the β_i parameters ($||\beta||_1 = \sum_{i=1} |\beta_i|$). No algorithm for large-scale data was given.
- In [190] the authors optimize the standard SVM criterion with the additional constraint that

$$w = \sum_{x \in \mathcal{X}_{\mathrm{PV}}} \beta_i \varphi(x_i), \tag{3.16}$$

  where $x_i \in \mathcal{X}_{\mathrm{PV}}$ is a set of $M \ll N$ Prototype Vectors (PVs)[2] selected from the training set. The resulting non-convex optimization problem both select a number of M PVs and corresponding β_i at the same time but for large-scale data no algorithm is provided.
- In [116] first a random subset of the training is chosen as PVs and then the β_i's are optimized which resulted in a computationally more realistic approach.
- In [101] two practical greedy approaches are proposed to select the PVs. The methods are strongly related to the pre-fitting and back-fitting procedures which are explained in [187] in the context Kernel Matching Pursuit which will be discussed later in section Section 3.3 and Chapter 5. Those are mainly developed for least-squares problems. In [101] the back-fitting procedure is preferred since both are similarly in terms of generalization and sparsity of the model while the back-fitting selection method is computationally less expensive. In most cases this method reaches the same accuracy as a standard SVM while using less PVs. This means that from a time complexity point of view, this approach can mostly reach the same accuracy as a SVM in less time. The algorithm roughly scales as $\mathcal{O}(NM^2)$. For more references and relations see [101].

[2] We make a distinction between SVs, which follow directly from the convex objective of SVM, and PVs which are selected in a non-convex manner. PVs might not coincidence with training examples while SVs always do.

Remark 7. Since we are interested in fast and accurate classifier algorithms (with fast evaluation) using non-linear decision functions, and since our main focus is not on variable (feature) selection, we do not consider sparseness in the primal weights w_i. For completeness we refer to some examples which do, such as: (i) recursive feature elimination [73] which uses SVM in combination with external feature selection methods, or (ii) those using a L_1-norm penalty on the weights in (3.6) instead of the original L_2 (e.g. [203]).

3.3 Least-Squares Support Vector Machines

The Least Squares Support Vector Machine (LS-SVM) framework is discussed in [172]. Although the LS-SVM framework is studied in many other contexts as well, we will only consider it in case of classification problems which is directly related to the regression case. As will be shown next, the framework is based on solving a system of linear equations. Given a training set, the convex primal problem of the LS-SVM classifier can be formulated as

$$\min_{w,e_i',b} \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{N} e_i'^2 \quad \text{such that } y_i f(x_i) = 1 - e_i', i = 1, ..., N, \tag{3.17}$$

where a quadratic loss is used instead of the hinge loss function typically employed in SVMs where $f(x) = w^T \varphi(x) + b$. Another difference is the use of equality constraints instead of inequality constraints. The e_i' are slack variables allowing deviation form the target labels y_i. The formulation can be considered as a regression on the labels y_i and is equivalent to

$$\min_{w,e_i,b} \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{i=1}^{N} e_i^2 \quad \text{such that } f(x_i) = y_i - e_i, i = 1, ..., N, \tag{3.18}$$

where $e_i' = y_i e_i$.

Remark 8. Objective function (3.18) can be rewritten in the form as defined in (3.7) where the loss function is now defined as $L(f(x), y) = (1 - yf(x))^2$ (this quadratic loss is often referred to as L_2 in the literature) and $\varrho = \frac{\gamma}{2}$. In Fig. 3.1 this loss function is plotted.

The dual problem is given in the following Lemma.

Lemma 3. *[168] Given a positive definite kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ with $K(x, x') = \varphi(x)^T \varphi(x')$ and a regularization constant $\gamma \in \mathbb{R}^+$, the solution to*

(3.18) is given by the following dual problem:

$$\left(\begin{array}{c|c} \Omega + \frac{1}{\gamma} I & 1_N \\ \hline 1_N^T & 0 \end{array}\right) \left(\begin{array}{c} \alpha \\ \hline b \end{array}\right) = \left(\begin{array}{c} y \\ \hline 0 \end{array}\right), \tag{3.19}$$

where $1_N = (1, \ldots, 1)^T, 1_N \in \mathbb{R}^N$, $y = (y_1, ..., y_N)^T$, $\alpha = (\alpha_1, ..., \alpha_N)^T$, $\Omega_{ij} = K(x_i, x_j)$ *and* $I = \text{diag}(1, \ldots, 1), I \in \mathbb{R}^{N \times N}$ *the identity matrix.*

Proof. The Lagrangian of the problem (3.18) is

$$\mathcal{L}(w, e_i, b; \alpha_i) = \frac{\gamma}{2} \sum_{i=1}^{N} e_i^2 + \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \left(w^T \varphi(x_i) + b - y_i + e_i \right), \tag{3.20}$$

where the α_i values are the Lagrange multipliers. The conditions for optimality yield

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 & \rightarrow w = \sum_{i=1}^{N} \alpha_i \varphi(x_i) \\ \frac{\partial \mathcal{L}}{\partial e_i} = 0 & \rightarrow \alpha_i = \gamma e_i, i = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \rightarrow \sum_{i=1}^{N} \alpha_i = 0 \\ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 & \rightarrow w^T \varphi(x_i) + b - y_i + e_i = 0, i = 1, \ldots, N. \end{cases} \tag{3.21}$$

Eliminating w and e_i leads to

$$\begin{cases} \frac{\alpha_i}{\gamma} + \sum_{j=1}^{N} \alpha_j \varphi(x_j)^T \varphi(x_i) + b = y_i, i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i = 0, \end{cases} \tag{3.22}$$

using $K(x, x') = \varphi(x)^T \varphi(x')$ and writing the above equations in matrix form gives the linear system in (3.19). □

The classifier model can be evaluated in the dual representation as:

$$\hat{f}(x) = \text{sign} \left(\sum_{i=1}^{N} \alpha_i K(x, x_i) + b \right). \tag{3.23}$$

3.3.1 Large-scale implementations

By simplifying the SVM formulation the optimization problem is transformed into solving a set of linear equations instead of solving a QP problem. In [169] large-scale problems with LS-SVM are solved using a Conjugate Gradient (CG)

algorithm [85]. An improved conjugate gradient algorithm was given in [35]. An alternative large-scale algorithm, with similarities to the SMO approach in SVM implementations, was proposed by [99]. On current standard computer systems [3] LS-SVM problems having up to about $5,000$ points are easily solved using standard solvers.

3.3.2 Sparseness

The use of equality constraints and the L_2 loss function simplifies the LS-SVM training problem compared to that of SVM. Due to the equality constraints the model is typically expressed in terms of dense kernel expansions where each training data point has a contribution (which can be seen from the optimality condition $\alpha_i = \gamma e_i$).

In order to obtain faster model evaluation in [170], and [173] the authors proposed a pruning mechanism based on sorting the α values, i.e. the absolute value of the solution of LS-SVM. A more sophisticated pruning scheme is explained in [108] for function estimation and in [200] a SMO-based pruning method is proposed. However, in case of large-scale problems these methods are not feasible since they all solve a large linear system which is iteratively shrunken. We therefore do not consider these type of algorithms and concentrate on other methods within this family, which are scalable to large-scale problems and produce sparse models.

As was stated before, in case of creating approximations to SVM models (which produce sparser models than the original formulation), on the one hand a representative set of PVs has to be chosen and on the other hand the corresponding model parameters have to be defined. It is desired to optimize both at the same time. However, in order to keep the computational costs limited several heuristics are applied:

- In Fixed-Size LS-SVM (FS-LSSVM) an active selection of PVs prior to inferring the parameters is determined. This means that the training set is split up in two steps: a first step consists of selecting M PVs of the complete training set which are used to approximate the feature map φ such that $\sum_{i=1}^{N}\sum_{j=1}^{N}(\hat{\varphi}(x_i)^T\hat{\varphi}(x_j) - K(x_i,x_j))^2$ is as small as possible. Using the approximate feature map the primal weight space problem is then solved.

[3] We assume a standard computer to have 2 gigabytes of memory and a dual core 2.2 gigahertz processor.

- Previous attempt decomposes the problem into: (i) the selection of PVs, and (ii) the actual model inference. Other heuristics exist that try to solve both in one process. In Chapter 5 we present a different view on how to implement a sparse LS-SVM solution. This is achieved by using an altered CG optimization algorithm which uses sparse conjugate directions on an approximate LS-SVM objective. The final model solution is very much related to other techniques such as Kernel Matching Pursuit (KMP) [187], Sparse Greedy Gaussian Process (SGGP) [163], other sparse LS-SVM versions [25], [90] and MARK [9].

Remark 9. For completeness we note that in case of using the parametric primal form, similar to SVMs, also in this context the L_2-norm on the weights w in (3.18) can be replaced by or combined with a L_1 norm which will lead to sparseness in the parameters w. Solving L_1 regularized least-squares problems is, however, more challenging than solving its L_2 regularized counterpart, since the regularization term is non-differentiable. The optimization problem in this case can be reformulated to a QP opposed to a linear system. This principle is applied in the estimation of sparse parametric models known as basis pursuit [32] and LASSO [175].

3.3.3 Fixed-size Least Squares Support Vector Machines

Since the algorithms we discuss in the next chapters use the approach of Fixed-Size LS-SVMs (FS-LSSVM) we give some more details about this technique. We now proceed by reviewing how to estimate an approximation to the feature map $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ using the Nyström method [189] and then discuss how to solve the problem in primal weight space using the estimated eigenfunctions and eigenvalues.

Approximation to the feature map

Suppose one takes a finite dimensional feature map (e.g. a linear kernel). Then one can equally well solve the primal (3.18) as the dual (3.19) problem. In fact solving the primal problem is more advantageous for larger data sets common in speech recognition because the dimension of the parameters $w \in \mathbb{R}^D$ is smaller compared to that of $\alpha \in \mathbb{R}^N$. In order to work in the primal space using a kernel function other than the linear one, it is required to compute an explicit approximation of the nonlinear mapping $\hat{\varphi} : \mathbb{R}^D \to \mathbb{R}^M$, such that $\Omega_{ij} \approx \hat{\varphi}(x_i)^T \hat{\varphi}(x_j)$ with $M \ll N$. This leads to a sparse representation of the model when estimating in primal space.

Explicit expressions for φ can be obtained by means of an eigenvalue decomposition of the kernel matrix Ω with entries $K(x, x')$. Consider the eigenfunction expansion

of a kernel function

$$K(x, x') = \varphi(x)^T \varphi(x') = \sum_i \lambda_i \phi_i(x)^T \phi_i(x'), \tag{3.24}$$

where λ_i and ϕ_i are respectively the eigenvalues and eigenfunctions, defined by the integral equation $\int K(x, x')\phi_i(x)p(x)dx = \lambda_i \phi_i(x')$ for a variable x with a probability density $p(x)$. Given (3.24) we can write

$$\varphi(x') = \left(\sqrt{\lambda}_1 \phi_1(x'), \sqrt{\lambda}_2 \phi_2(x'), \ldots, \sqrt{\lambda}_{D_\varphi} \phi_{D_\varphi}(x')\right)^T. \tag{3.25}$$

Note that D_φ can be infinite when for instance an RBF kernel is used. The Nyström method [189] approximates the integral by means of a sample average

$$\lambda_i \phi_i(a) \approx \frac{1}{N} \sum_{l=1}^{N} K(x_l, a)\phi_i(x_l), \forall a \in E \subseteq \mathbb{R}^D. \tag{3.26}$$

Let $a = x_j$, in matrix notation one obtains the eigenvalue problem,

$$\Omega U = U\Lambda, \tag{3.27}$$

where $U = (u_1, \ldots, u_N)$ is the square matrix of eigenvectors of Ω, $u_i = ((u_i)_1, \ldots, (u_i)_M)^T$, $\Lambda = \text{diag}(\lambda_1^s, \ldots, \lambda_N^s)$ is a diagonal matrix of nonnegative eigenvalues in decreasing order, and $\Omega_{ij} = K(x_i, x_j)$ are the elements of the kernel matrix. The eigenvalues λ_i and eigenfunctions ϕ_i from the continuous problem in expression (3.25) can be approximated by the sample eigenvalues λ_i^s and eigenvectors $u_i \in \mathbb{R}^N$ as

$$\lambda_i \approx \frac{1}{N}\lambda_i^s, \phi_i \approx \sqrt{N} u_i. \tag{3.28}$$

Substituting (3.28) in (3.26) results in an approximation of an eigenfunction evaluated in point $a \in E \subseteq \mathbb{R}^D$,

$$\hat{\phi}_i(a) = \frac{\sqrt{N}}{\lambda_i^s} \sum_{j=1}^{N} (u_i)_j K(x_j, a). \tag{3.29}$$

Based on this approximation, an expression for the entries of the approximated feature map $\hat{\varphi}_i : \mathbb{R}^D \to \mathbb{R}, i = 1, \ldots, N$ for a training point, or for any new point x^*, with $\hat{\varphi} = (\hat{\varphi}_1, \ldots, \hat{\varphi}_N)^T$, is given as

$$\hat{\varphi}_i(x^*) = \sqrt{\lambda_i}\hat{\phi}_i(x^*) \tag{3.30}$$

$$= \frac{1}{\sqrt{\lambda_i^s}} \sum_{j=1}^{N} (u_i)_j K(x_j, x^*). \tag{3.31}$$

Until now the entire training set of size N is used to compute the approximate feature map. Therefore $\hat{\varphi}(x)$ will yield at most N components, each one of which can be computed by (3.31) for all $x_i, i = 1, \ldots, N$. However, if we have a large scale problem, it has been motivated [172] to use a subsample of $M \ll N$ data points to compute the $\hat{\varphi}(x)$. In this case, up to M Prototype Vectors (PVs) will be computed. The function $f(x)$ will be approximated as follows

$$\hat{f}(x) = \sum_{i=1}^{M} w_i \left(\frac{1}{\sqrt{\lambda_i^s}} \sum_{k=1}^{M} (u_k)_i K(x_k, x) \right) + b. \tag{3.32}$$

Remark 10. Note that the computation of the features in matrix notation is

$$\hat{\Phi} = \Omega' U S, \tag{3.33}$$

with matrix $\hat{\Phi} \in \mathbb{R}^{N \times M}$ defined as

$$\hat{\Phi} = \begin{pmatrix} \hat{\varphi}_1(x_1) & \ldots & \hat{\varphi}_M(x_1) \\ \vdots & \ddots & \vdots \\ \hat{\varphi}_1(x_N) & \ldots & \hat{\varphi}_M(x_N) \end{pmatrix}, \tag{3.34}$$

and $\Omega' \in \mathbb{R}^{N \times M}, \Omega'_{ij} = K(x_i, x_j), x_i \in \mathcal{X}_{\text{tr}}, x_j \in \mathcal{X}_{\text{PV}}$ with $\mathcal{X}_{\text{PV}}$ the set of selected PVs and $\mathcal{X}_{\text{tr}}$ the set of training vectors and $S = \Lambda^{-\frac{1}{2}}$.

Solving the problem in primal space

Given the finite dimensional approximation to the feature map $\hat{\varphi}$, the following ridge regression problem can be solved in the primal weight space with parameters $\tilde{w} \in \mathbb{R}^M, b \in \mathbb{R}$ and M the number of PVs

$$\min_{\tilde{w}, b} \frac{1}{2} \tilde{w}^T \tilde{w} + \frac{\gamma}{2} \sum_{j=1}^{N} \left(y_j - \tilde{w}^T \hat{\varphi}(x_j) + b \right)^2. \tag{3.35}$$

The solution to (3.35) is then given by

$$\left(\begin{array}{c|c} \hat{\Phi}^T \hat{\Phi} + \frac{1}{\gamma} I & \hat{\Phi}^T 1_N \\ \hline 1_N^T \hat{\Phi} & 1_N^T 1_N \end{array} \right) \left(\begin{array}{c} \tilde{w} \\ \hline b \end{array} \right) = \left(\begin{array}{c} \hat{\Phi}^T y \\ \hline 1_N^T y \end{array} \right). \tag{3.36}$$

In order to have a positive definite matrix in (3.34), additionally a penalization using an L_2 norm is issued on the the intercept term b. Alternatively, one could opt to use a separate predefined small regularization parameter different from γ to bound the intercept term. In case of classification the prediction of a class label is then performed using $\hat{f}(x^*) = \text{sign}(\tilde{w}^T \hat{\varphi}(x^*) + b)$.

Active selection of prototype vectors

Instead of using a purely random selection of PVs, a Rényi entropy based selection method can be utilized as is proposed in [172] and advanced in [18]. The selected subsamples should represent the main characteristics of the whole training data set. In fact the choice of PVs directly affects the final FS-LSSVM model accuracy. More formally, in selecting $M \ll N$ PVs one aims at minimizing the mean squared error $\sum_{i=1}^{N}\sum_{j=1}^{N}(\hat{\varphi}(x_i)^T\hat{\varphi}(x_j) - K(x_i,x_j))^2$. Or in terms of information theory the subsample should contain as much information or entropy as possible to describe the entire set of data examples. The quadratic Rényi entropy $H_R^M(x)$ is defined as

$$H_R^M(x) = -\log \int p(x)^2 dx. \tag{3.37}$$

In order to compute this entropy, an estimate of the density $p(x)$ is required.

In [18] the authors consider the general multivariate kernel estimator to estimate $p(x)$ which is given by

$$\hat{p}(x') = \frac{1}{M\det(D)} \sum_{i=1}^{M}\sum_{i=1}^{M} \kappa(D^{-1}(x' - x_i)), \tag{3.38}$$

where the kernel $\kappa : \mathbb{R}^D \to \mathbb{R}$ satisfies $\int_{\mathbb{R}^D} \kappa(u)du = 1$, $\det(\cdot)$ denotes the determinant, D is a non-singular matrix of the form $D = \mathrm{diag}(h_1, \ldots, h_D)$ with h_i the kernel bandwidths per dimension. Combining (3.38) and (3.37), Rényi's quadratic entropy estimator becomes

$$H_R^M(X) = -\log \frac{1}{M^2\det(D)^2} \sum_{i=1}^{M}\sum_{j=1}^{M} \kappa\left((D\sqrt{2})^{-1}(x_i - x_j)\right), \tag{3.39}$$

where in this case $X = (x_1, \ldots, x_M)^T$.

Using this entropy one can evaluate the amount of information a subsample of a certain size M is expressing. The goal is now to maximize this entropy by evaluating different sets of subsample. In [172] an algorithm given in Algorithm 1 is defined for this purpose. As a start a randomly selected working set with a predefined size M is created. In this working set a point is randomly selected and replaced by a randomly selected point from the training input sample if the new point improves the Rényi's quadratic entropy criterion.

In [18] a fast plug-in method is discussed to determine the kernel bandwidth parameters h_i for a RBF kernel.

Algorithm 1 Active support vector selection [172]

Input: The input data matrix $X \in \mathbb{R}^{N\times D}$ containing N data samples described by D attributes, M number of PVs.
Output: M prototype vectors.
Define: Index sets $\mathcal{S} = \{1, .., N\}$, $\mathcal{A} = \emptyset$, $\mathcal{A}' = \emptyset$, $X(\mathcal{S})$ the rows of X selected by the index set $\mathcal{S}$, $H_R^M(X)$ is defined as in (3.39) .
Randomly select M indices out of $\mathcal{S}$ and store in $\mathcal{A}$.
$\mathcal{S} := \mathcal{S}\backslash\mathcal{A}$
repeat
  Randomly select an index i out of $\mathcal{A}$ and j out of $\mathcal{S}$.
  $\mathcal{A}' = (\mathcal{A}\backslash\{i\})\bigcup\{j\}$.
  if $H_R^M(X(\mathcal{A}')) > H_R^M(X(\mathcal{A}))$ **then**
    $\mathcal{A} := \mathcal{A}'$, $\mathcal{S} := (\mathcal{S}\backslash\{j\})\bigcup\{i\}$.
  end if
until Stopping criterion is met

Remark 11. Note that FS-LSSVM is related to RBF networks which can be seen from the explanation in [172] (section 1.6, p. 22). However, compared to RBF networks the FS-LSSVM method can generally be used with any positive semi-definite kernel.

3.4 Kernel Logistic Regression

In this section we briefly recall the principles of binary kernel logistic regression (KLR). We restrict ourselves to the binary case in this section in order to compare with (LS-)SVM. In Chapter 4 we will further elaborate on the natural multi-class extension. Instead of using a risk minimization principle as is used in case of SVMs, KLR is based on conditional maximum likelihood inference which results in estimates of a-posteriori class probabilities via logit stochastic models

$$\begin{cases} \Pr(\mathbf{Y} = -1 \mid \mathbf{X} = x; f) = \frac{\exp(f(x))}{1+\exp(f(x))} \\ \Pr(\mathbf{Y} = 1 \mid \mathbf{X} = x; f) = \frac{1}{1+\exp(f(x))}, \end{cases} \tag{3.40}$$

where $f(x) = w^T\varphi(x) + b$. The class membership of a new point x_* can be given by the classification rule which is

$$\underset{c\in\{-1,1\}}{\arg\max}\ \Pr(\mathbf{Y} = c|\mathbf{X} = x_*; f). \tag{3.41}$$

For notational simplicity we use shorthand notation

$$p_{ci} = \Pr(\mathbf{Y} = c|\mathbf{X} = x_i; w). \tag{3.42}$$

The common method to infer the parameters of the different models is via the use of a penalized negative log likelihood (PNLL) criterion resulting in the following primal weight space objective function

$$\min_{w,b} \frac{1}{2} w^T w + \frac{\nu}{2} \sum_{i=1}^{N} \log\left(1 + \exp(-y_i f(x_i))\right). \tag{3.43}$$

Remark 12. The objective function (3.43) can be rewritten in the form as defined in (3.7) where the loss function is now defined as $L(f(x), y) = -\log\left(1 + \exp(-yf(x))\right)$ and $\varrho = \frac{\nu}{2}$. In Fig. 3.1 this logistic loss is plotted. It is seen that its shape is very similar to that of the hinge loss used by SVM. Both increase linearly as $yf(x)$ gets very small (negative) and both encourage y and $f(x)$ to have the same sign. In fact, in [202] it is proven that KLR, similar to SVM, can also be considered as a margin maximizer.

In [153], and [202] it is seen that (3.43) can be rewritten as an iteratively re-weighted least squares (IRLS) problem. Note that in [143] it is shown that the SVM solution can also be obtained by a IRLS scheme. Starting from initial parameter vectors $w^{(0)}$ the solution can be found by iterating

$$w^{(k)} = w^{(k-1)} + s^{(k)}, \tag{3.44}$$

where step $s^{(k)}$ is determined in each iteration k by the following minimization problem

$$q^{(k)} = \min_{s^{(k)}} \frac{1}{2}(s^{(k)} + w^{(k-1)})^T (s^{(k)} + w^{(k-1)}) + \frac{\nu}{2} \sum_{i=1}^{N} g_i^{(k)} (e_i^{(k)})^2$$

$$\text{such that } s^{(k)T} \varphi(x_i) + b = z_i^{(k)} - e_i^{(k)}, i = 1, \ldots, N. \tag{3.45}$$

where

$$z_i^{(k)} = \frac{(p_{y_i i}^{(k)} - 1) y_i}{g_i^{(k)}}, \tag{3.46}$$

and $g_i^{(k)} = p_{1i}^{(k)}(1 - p_{1i}^{(k)})$.

Lemma 4. *[93] Given a positive definite kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ with $K(x, x') = \varphi(x)^T \varphi(x')$ and a regularization constant $\nu \in \mathbb{R}^+$, the solution to (3.45) at iteration k is given by the following dual problem:*

$$\left(\begin{array}{c|c} \Omega + \frac{1}{\nu}(W^{(k)})^{-1} & 1_N \\ \hline 1_N^T & 0 \end{array}\right) \left(\begin{array}{c} \alpha^{(k)} \\ \hline b^{(k)} \end{array}\right) = \left(\begin{array}{c} z^{(k)} + \Omega\alpha^{(k-1)} \\ \hline 0 \end{array}\right), \tag{3.47}$$

where $1_N = (1, \ldots, 1)^T, 1_N \in \mathbb{R}^N$, $z^{(k)} = (z_1^{(k)}, \ldots, z_N^{(k)})^T$, $\alpha^{(k)} = (\alpha_1^{(k)}, \ldots, \alpha_N^{(k)})^T$, $\Omega_{ij} = K(x_i, x_j)$, $W^{(k)} = \mathrm{diag}(g_1^{(k)}, \ldots, g_N^{(k)})^T$.

Proof. The Lagrangian of the constrained problem as stated in (3.45) becomes

$$\begin{aligned} \mathcal{L}(w^{(k)}, e_i^{(k)}, b^{(k)}; \alpha_i^{(k)}) =& \frac{\nu}{2} \sum_{i=1}^{N} e_i^{(k)2} g_i + \frac{1}{2} \left(s^{(k)} + w^{(k-1)}\right)^T \left(s^{(k)} + w^{(k-1)}\right) \\ & - \sum_{i=1}^{N} \alpha_i^{(k)} \left(s^{(k)T} \varphi(x_i) + b^{(k)} + e_i^{(k)} - z_i^{(k)}\right), \end{aligned} \tag{3.48}$$

where the α_i values again are the Lagrange multipliers. The conditions for optimality yield

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial s^{(k)}} = 0 & \to s^{(k)} = \sum_{i=1}^{N} \alpha_i^{(k)} \varphi(x_i) - w^{(k-1)} \\ \frac{\partial \mathcal{L}}{\partial e_i^{(k)}} = 0 & \to \alpha_i^{(k)} = \nu g_i^{(k)} e_i^{(k)}, i = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial b^{(k)}} = 0 & \to \sum_{i=1}^{N} \alpha_i^{(k)} = 0 \\ \frac{\partial \mathcal{L}}{\partial \alpha_i^{(k)}} = 0 & \to s^{(k)T} \varphi(x_i) + b^{(k)} - z_i^{(k)} + e_i^{(k)} = 0, i = 1, \ldots, N. \end{cases} \tag{3.49}$$

Eliminating $s^{(k)}$ and $e^{(k)}$ leads to,

$$\begin{cases} \frac{\alpha_i^{(k)}}{g_i^{(k)} \nu} + \sum_{j=1}^{N} \alpha_j^{(k)} \varphi(x_j)^T \varphi(x_i) + b^{(k)} = z_i^{(k)} + \sum_{i=1}^{N} w^{(k-1)T} \varphi(x_i), i = 1, \ldots, N \\ \sum_{i=1}^{N} \alpha_i^{(k)} = 0. \end{cases} \tag{3.50}$$

The solution $w^{(k_L)}$ can be expressed in terms of $\alpha^{(k_L)}$ computed in the last iteration. This can be seen when combining (3.44) and the formula for $s^{(k)}$ in (3.49) which gives

$$w^{(k_L)} = \sum_{i=1}^{N} \alpha_i^{(k_L)} \varphi(x_i). \tag{3.51}$$

using (3.51) in (3.50) and $K(x, x') = \varphi(x)^T \varphi(x')$ we can write the linear system in matrix form as given in (3.47). □

From (3.51) it is seen that in the dual representation the function values $f(x)$ in the KLR logit models (3.40) can be computed as follows

$$f(x) = \sum_{i=1}^{N} \alpha_i^{k_L} K(x, x_i) + b^{k_L}. \tag{3.52}$$

3.4.1 Large-scale implementation

For small sized learning problems KLR is usually solved using the IRLS method described in previous section. For large-scale problems this method becomes computationally expensive since the linear system in (3.47) must be solved exactly in each Newton iteration. For that reason the authors in [100] developed a dual formulation similar to the dual arising in SVMs that allows an efficient optimization of the dual variables. In [158] a truncated Newton method is used, where approximate Newton directions are searched very efficiently using the model structure and linear conjugate gradients in order to reduce the computation to standard linear algebra primitives. But still these KLR algorithms do not scale to very large problems which we aim at. Moreover, the final model in the dual parameters is, similar to that of LS-SVMs, not sparse. In order to obtain large-scale algorithms again heuristics can be used to approximate the classical KLR formulation which, as a byproduct, can lead to sparse models in the dual variables.

3.4.2 Sparseness

As with the other discussed kernel approaches approximating the standard formulation can lead to sparse results. Examples are:

- An approximate and sparse KLR version is known as the Import Vector Machine (IVM) [202]. It is based on a greedy approach which aims at minimizing the negative log likelihood criterion iteratively by adding training samples to a subset of selected prototype vectors until the algorithm converges to some predefined value.
- In [154] a sparse KLR solution is achieved by replacing the penalization term in (3.43) by the L_1 norm on the dual unknowns α.

- In [84] an adaptation of maximum likelihood estimation, instantiated on logistic regression is proposed. The model outputs proper conditional probabilities into a user-defined interval and is less precise elsewhere.

These algorithms are still computationally demanding for very large data sets. In [98] the authors conclude by stating that the main problem of IVM is the computationally expensive estimation of the model parameters such that it is not applicable for large data sets. Since KLR can be expressed as an iterative re-weighted LS-SVM, the approximation techniques known in the context of LS-SVMs can potentially be used within this context. From that point of view we implemented a fixed-size variant of the standard KLR formulation (FS-KLR) (see Chapter 4) which does easily scale to very large data sets.

Note the existence of related sparse Bayesian methods, such as the relevance vector machine [177] using an automatic relevance determination framework [123] and within the context of Gaussian Processes (GPs) the informative vector machine [109].

Remark 13. As with the other kernel-methods several linear logistic regression algorithms estimate sparse weight vectors. This is typically done using the L_1-norm. Examples can be found in for instance [26], [106].

3.5 Cross-validation based model selection

In order to assess the model hyper-parameters such as the regularization parameter ϱ in (3.7) and the RBF-kernel bandwidth parameter σ a cross-validation (CV) based method is used. A hyper-parameter pair is given a score according to some model selection criterion. A typical criterion expresses the number of misclassifications on unseen validation data

$$\text{MISCLASS}(y,z) = \sum_{i=1}^{N} \max\left(0, \text{sign}(-y_i z_i)\right), \tag{3.53}$$

where $y = (y_1, ..., y_N)^T$, and the estimated latent variables $z_i = f(x_i)$ are stacked as $z = (z_1, \ldots, z_N)^T$. The above criterion is not entirely suitable for model selection purposes since a continuous function is preferred that is more amenable to numerical optimization routines. Possible continuous functions are the press statistic [1]

$$\text{MSE}(y,z) = \frac{1}{N}\sum_{i=1}^{N}(y_i - z_i)^2, \tag{3.54}$$

(for classification problems its use was motivated in [26]) or, in case probabilistic outcomes are available, the negative log likelihood criterion

$$\mathrm{NLL}(y,z) = -\sum_{i=1}^{N} \ln \Pr(\mathbf{Y} = y_i | \mathbf{X} = x_i; \theta), \tag{3.55}$$

where $(\mathbf{Y}, \mathbf{X}) \in \mathbb{R}^D \times \{1, ..., C\}$ are random variables with C the number of classes and θ the ensemble of model parameters.

The actual hyper-parameter search process is a non-convex task which typically is tackled via a simple grid-search based procedure, in which the model selection score is evaluated at a set of points, forming a regular grid with even logarithmic spacing. An alternative strategy is using Nelder-Mead simplex optimization [131]. This procedure can be used as long as the number of hyper-parameters is relatively small. In order to determine good initial start values for this simplex method we use the method of Coupled Simulated Annealing with variance control (CSA) [195], [196] and [197] whose working principle was inspired by the effect of coupling in Coupled Local Minimizers (CLM) [171] compared to the uncoupled case i.e. multi-start based methods. In cases where a relative high number of hyper-parameters need to be tuned at once, gradient-based methods are likely to be more efficient.

Note that to avoid the effects of the randomness in creating different CV partitions on the determination of the hyper-parameters first a fixed random partition is chosen, then the model selection criterion is minimized.

3.6 Estimating A-Posteriori Probabilities

Binary classifiers such as SVM and LS-SVM compute a decision function $f(x)$ such that $\mathrm{sign}\,(f(x))$ can be used to predict the label of any test example x. Some applications, such as speech recognition, require a-posteriori probabilities $\Pr(\mathbf{Y} = c | \mathbf{X} = x)$ instead of predictions of the labels. Several methods exist to approximate the posterior probabilities from the classifier function outputs (e.g. [184], [77], [182], [198], [146]). In the LIBSVM toolbox, which we will use to report SVM results in the remainder, an improved version of the method described in [146] to approximate the posterior probabilities is employed. Since we would like to compare the outcomes using this toolbox to those obtained with other methods we will use the same approximation for all other kernel-based alternatives.

In [146] the authors proposed to approximate the a-posteriori probabilities by mapping a sigmoid function on the latent function values $f(x_i), i = 1, \ldots, N$.

This sigmoid parametric model was inspired by empirically looking at the model outcomes and is defined as follows

$$\Pr(\mathbf{Y} = c|\mathbf{X} = x) \approx P^{AB}(x) \equiv \frac{1}{1 + \exp(Af(x) + B)}. \tag{3.56}$$

Supposing that the final outcomes are a-posteriori probabilities of class membership, then provided the data represent an i.i.d. (independent and identically distributed) sample from a Bernoulli distribution conditioned on the input variables, the likelihood of the training data is given by

$$\ell = \prod_{i=1}^{N} p_i^{t_i} \left(1 - p_i\right)^{1-t_i}, \tag{3.57}$$

where $t_i = (y_i + 1)/2$ and

$$p_i = \frac{1}{1 + \exp(Af(x_i) + B)}. \tag{3.58}$$

In order to avoid over-fitting a prior model on out-of-sample data to regularize the objective function, is used. As explained in [146] this bowls down to use different target values (different than $t_i \in \{0, 1\}$), defined as

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & y_i = 1 \\ \frac{1}{N_- + 2} & y_i = -1, \end{cases} \tag{3.59}$$

where N_+ and N_- respectively indicate the number of observed positive and negative examples. These targets are used instead of $\{0, 1\}$ for all of the data in the sigmoid fit.

The authors in [146] use a Levenberg-Marquardt algorithm to optimize the objective in (3.57) with modified t_i values. In [118] another optimization algorithm which enjoys better convergence properties is used. We therefore resort to the latter implementation.

In order to have unbiased $f(x_i)$ values to infer parameters A and B, we will use a fixed validation set or v-fold cross validation. In case of v-fold cross validation, the training set is split into v parts. Each of v classifiers are trained permutations of $v - 1$ parts, and $f(x_i)$ are computed on the remaining part. The union of all v sets of $f(x_i)$ values is used to as a training set to determine the sigmoid parameters. Note that other hyper-parameters can be tuned using the same $f(x_i)$ of the hold-out part(s).

3.7 Multi-class Coding Schemes

The kernel-based algorithms discussed in previous sections are designed for binary classification and it is therefore usually not straightforward to extend these type of classifiers to the multi-class case. Methods for multi-class tasks are based on the combination of binary classifiers or on all-at-once classification. An example of the latter in the context of KLR, which does has a natural extension to the multi-class setting, will be discussed in Chapter 4. In this section we will briefly discuss the combination of binary classifiers to perform multi-class classification.

There are several ways to construct the set of binary classifiers, which can be combined to give a multi-class prediction, such as minimum output coding, error correcting output codes, one-versus-all and one-versus-one output coding [172]. In this thesis we will focus on the latter two, which are most frequently used in practice. In one-versus-all (ova) coding one takes the number of outputs equal to the number of classes C and makes binary classifiers between each class and all other classes. In one-versus-one (ovo) output coding one uses $C(C-1)/2$ smaller classifiers, where each binary model discriminates between two opposite classes in a pairwise manner.

A commonly used technique to combine the binary classifier results is the use of a voting strategy (e.g. applied to SVMs in [62]). However, in this dissertation, we are interested in obtaining a-posteriori probabilities of class membership which gives opportunities to use more sophisticated combination techniques than just using a commonly used voting strategy. A list of possible candidates with a benchmarking study was given in [180] in the context of medical data. The authors concluded that using ensembles of nested dichotomies [61] was the most effective combination technique. This method obtains multi-class probabilities by advancing through a tree of binary classifiers. At the top level, all classes are divided into two subgroups for which a binary classifier is trained. Subgroups that consist of more than one class are again divided into two groups until all classes form a subgroup of their own. The tree then consists of C leaves. The multi-class probability p_{ci} is obtained by multiplying all binary probabilities involving p_{ci}. Since several equally likely trees can be constructed, the overall class probability estimates can be obtained by averaging the estimates obtained from the different trees. The number of different trees for C classes grows exponentially which makes this method impractical for the speech recognition tasks we aim at. In [180] the performance of ensembles of nested dichotomies was closely followed by the performances of the approaches described as first and second method in [191] which are feasible for large data sets. Since the second method (which we will refer to as *pc* in the remainder of this work) is

also used in the LIBSVM toolbox [117] we will consider this combination strategy and briefly review its theory. Due to its simplicity and since in [191] it seems that its performance is quite competitive compared to the others we additionally have implemented the method described in [148] and later on refer to it as *pkpd*.

Pairwise Coupling Let us denote the estimated probability of a data point x_i belonging to the c-th of C classes as $\hat{p}_{ci} = \Pr(\mathbf{Y} = c|\mathbf{X} = x_i)$, and the probability of a data vector belonging to the c_1-th class conditioned on the membership of class c_1 or c_2 as $\mu_{c_1c_2}(x_i) = \Pr(\mathbf{Y} = c_1|\mathbf{Y} \in \{c_1, c_2\}, \mathbf{X} = x_i)$.

The pairwise coupling method to combine results from a one-versus-one decomposition of the multi-class problem was introduced in [77]. The authors estimated the a-posteriori probabilities by minimizing the Kullback-Leibler divergence KL between the $\mu_{c_1c_2}(x_i)$ and $\hat{\mu}_{c_1c_2}(x_i) = \hat{p}_{c_1i}/(\hat{p}_{c_1i} + \hat{p}_{c_2i})$. After setting the derivative of KL to zero, the following constraints are obtained which can be used to estimate the multi-class probabilities

$$\begin{cases} \sum_{c_1=1,c_1\neq c_2}^{C} N_{c_1c_2}\mu_{c_1c_2}(x_i) = \sum_{c_1=1,c_1\neq c_2}^{C} N_{c_1c_2}\hat{\mu}_{c_1c_2}(x_i) \\ \sum_{c=1}^{C} p_{ci} = 1, p_{ci} > 0, \end{cases} \tag{3.60}$$

for $i = 1, \ldots, N$ and where N_{ij} is the number of training data in the i-th and j-th classes.

In [191], based on the assumption that the multi-class data are balanced such that weighting is not necessary, a minimization problem is written for which (3.60) is the global minimum

$$\min_{p_{1i},\ldots,p_{Ci}} \sum_{c_1=1}^{C} \left(\sum_{c_2=1,c_2\neq c_1}^{C} \hat{\mu}_{c_2c_1}(x_i)p_{c_1i} - \sum_{c_2=1,c_2\neq c_1}^{C} \hat{\mu}_{c_1c_2}(x_i)p_{c_2i} \right)^2 \tag{3.61}$$

In their second approach this minimization problem is then further relaxed into

$$\begin{aligned} \min_{p_{1i},\ldots,p_{Ci}} \sum_{c_1=1}^{C} \sum_{c_2=1,c_2\neq c_1}^{C} & (\hat{\mu}_{c_2c_1}(x_i)p_{c_1i} - \hat{\mu}_{c_1c_2}(x_i)p_{c_2i})^2 \\ & \text{such that } \sum_{c=1}^{C} p_{ci} = 1, \end{aligned} \tag{3.62}$$

which can be solved by a simple linear system resulting in a combination scheme which is simple in both practical implementation and algorithmic analysis.

Price et al. This method uses the probability of a union of events, and by stating that $\hat{\mu}_{c_1 c_2}(x_i) \approx \mu_{c_1 c_2}(x_i)$, one ends up with the following a-posteriori probability estimates

$$\Pr(\mathbf{Y} = c_1 | \mathbf{X} = x) = \frac{1}{\left(\sum_{c_2=1, c_2 \neq c_1}^{C} \frac{1}{\hat{\mu}_{c_1 c_2}(x)} - (C-2) \right)}. \tag{3.63}$$

The probability outcomes in (3.63) are normalized so that they sum to one for each evaluation.

3.8 Learning with a universum

Semi-supervised classification is an important topic in current machine learning research. It refers to a special form of classification. Traditional classifiers, as explained before, use only labeled data (x_i, y_i) pairs to train. However, labeled examples are often difficult to obtain since they require human effort to annotate them. Unlabeled data on the other hand might be relatively easy collected. Semi-supervised learning addresses the problem of how to use (a large amount of) unlabeled data, together with the labeled data, to build "better" classifiers. An overview of semi-supervised learning is given in [205].

A special case of semi-supervised classification is when the set of unlabeled examples are known to belong to any other class. The latter collection of data, called universum data in [183] and [188], allows one to model prior knowledge. We suppose the universum to contain data from the same domain as the problem at hand and expect it to represent meaningful information related to the classification task. In this thesis universum data occurs naturally in the employed segmental speech recognition setup. This speech recognition system is based on classifying fixed-length feature vectors representing a phone. The feature vectors are calculated given the phone boundaries which assumes a given segmentation of the phone utterance. Training a model might only employ data generated from a correctly segmented utterance. However, during the evaluation phase the models are confronted with fixed-length feature vectors which are generated using a bad segmentation (since the correct segmentation is not known beforehand). Hence, vectors from the same domain which do not belong to any phone class are generated. During the training process a set of such universum vector can also be easily generated. In case this data is incorporated into the training process a better classifier might be obtained.

In [183] a framework is presented to incorporate universum data in the training phase. This framework can be seen as an alternative to the large margin concept

explained in [184]. In [188] an algorithm, in the context of SVMs, is given to leverage the universum by maximizing the number of observed contradictions. This algorithm is explained next.

In case of SVM one selects the function out of an equivalence class $\mathcal{F}$ (see Section 3.2), which is a set of functions having the same labeling on the non-universum training data, that maximizes the margin. As an alternative one could measure the number of contradicting functions in the equivalence class. A data point x^* from the universum data makes a contradiction on the equivalence class $\mathcal{F}$ if in $\mathcal{F}$ there are two functions $f_1(x)$ and $f_2(x)$ such that $f_1(x^*) > 0$ and $f_2(x^*) < 0$. The decision boundary is encouraged to pass through the universum. One interpretation is similar to the maximum entropy principle: the classifier should be confident on labeled examples, yet maximally ignorant on unrelated examples.

In [188] and [166] this idea of maximizing the number of contradictions is approximately implemented by penalizing the function outputs $f(x_i^*)$ for universum data which are far from the decision boundary more than those which are close it. As if only a small change in $f(x_i^*)$ will cause a contradiction on x_i^*. This corresponds to minimizing the distance of the universum data to the classifier boundary. One way to implement this is by adding $\sum_{x^* \in \mathcal{U}} L(f(x^*))$ (with $\mathcal{U}$ the set of universum examples and $|\mathcal{U}|$ the number of examples in the set) as a penalization term to the objective of a kernel method. In our experiments, which are explained in Chapter 4 and Chapter 6 we will use a L_2 loss for this purpose. Following the general notation in (3.7) we can now write

$$\min_{w,b} \frac{1}{2} w^T w + \varrho \sum_{i=1}^{N} L(f(x_i), y_i) + \varrho_{\mathcal{U}} \sum_{x^* \in \mathcal{U}} f(x^*)^2, \tag{3.64}$$

where $\varrho_{\mathcal{U}}$ is controlling the influence of the universum penalization term.

This idea is illustrated in figure Fig. 3.2. Here, a simple 2 dimensional example is presented which indicates how universum samples can affect the decision boundary of a classifier. The stars and squares stand for samples from two different classes, and the dots represent the universum data. The decision boundary of SVM (which does not incorporate the universum data) is given by the dashed line. The decision boundary of a classifier using a learning objective as in (3.64) (hence the universum data is used in the training procedure) and using the hinge loss is shown by the solid line.

Remark 14. In [166] the authors claim that one should keep the universum samples positioned in between different classes, since it gives the right direction to which

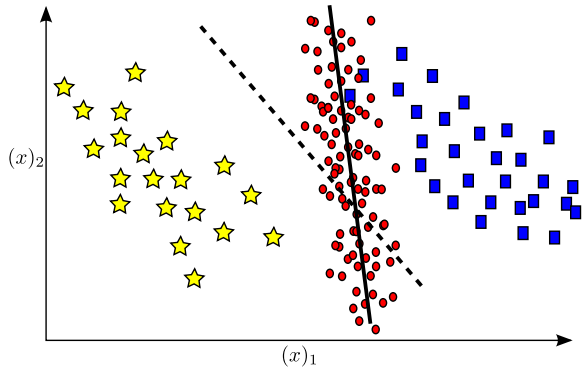


Figure 3.2: A simple 2 dimensional example of how universum samples can affect the decision boundary of a classifier. The stars and squares stand for samples from two different classes, and the dots represent the universum data. The decision boundary of a max-margin classifier (which does not incorporate the universum data) is given by the dashed line. The decision boundary of a classifier using a learning objective as in (3.64) (hence the universum data is used in the training procedure) is shown by the solid line.

the decision boundary should be dragged to. In [34], a method is proposed to select the "in-between" universum samples.

Chapter 4

Fixed-size Kernel Logistic Regression

In this chapter, emphasis is put on a non-linear "kernelized" variant of logistic regression called Kernel Logistic Regression (KLR) [88]. Unlike the empirical risk minimization principle utilized by SVM, (K)LR yields a-posteriori probabilities of membership in each of the classes based on a maximum likelihood argument. Thus, besides predicting class labels, (K)LR additionally provides a probabilistic interpretation about this labeling. (K)LR has an additional advantage that the extension to the multi-class case is well described, which must be contrasted to the coding approach (see e.g., [153], [172]). In [202] it has been shown that kernel logistic regression has related properties, and is comparable to SVM when it comes to classification accuracy. We propose a fast and accurate approximate implementation of KLR for large-scale data sets. This chapter is organized as follows: in Section 4.1 a brief motivation and some pitfalls when using KLR in speech recognition are given, in Section 4.2 we give some insight in the precise statistical setup of linear multi-class logistic regression and its non-linear extension kernel logistic regression and additionally provide modifications to the standard formalism. In Section 4.3 we give details about an approximate large-scale KLR implementation. In Section 4.4 we empirically verify the approximate KLR algorithm and its modifications on several public available data sets. We conclude in Section 4.5.

4.1 Introduction

Current Automatic Speech Recognizer (ASR) architectures are based on a probabilistic framework in which sub-models, such as acoustical and language models, are used. Since KLR directly estimates a-posteriori probabilities it seems to offer ample perspective for an integration into a speech recognizer. Moreover, the extension to the multi-class case is well described. However, there are three major difficulties which arise:

- First, although there are fast and efficient implementations available [158], [100], the classical dual formulation of KLR does not scale to large problems which are common in ASR. In contrast to SVM the optimization of the Kernel Logistic Regression objective does not lead to a sparse model as was seen in Section 3.4.
- Second, speech signals typically have variable lengths, while KLR is a static classifier, meaning that the observations are assumed to be fixed-dimensional vectors.
- Third, a speech signal corresponds to a sequence of words, and each word relates to an unknown portion of the speech signal. Thus, speech recognition is a sequence labeling problem, with unknown segmentation, while logistic regression is designed to predict a single label only.

We first formulate a solution to the problem of scalability. In this context the literature provides different adaptations to the original KLR algorithm to obtain sparseness (see Section 3.4.2). In this work we describe a different practical technique, suited for large data sets, based on Fixed-Size Least Squares Support Vector Machines (FS-LSSVMs) [172], which we can use because KLR is related to a weighted version of LS-SVMs [93]. To solve the approximate problem we use the optimization algorithm described in [119] as a basis. Our contributions with respect to this part are:

- The extension of the optimization algorithm mentioned above to the natural multi-class KLR setting.
- The comparison of different types of multi-class schemes for our approximate algorithm in terms of training complexity, classification performance, sparseness and evaluation speed.
- An empirical comparison of sampling techniques required for the approximation.

- The introduction of a reduced memory implementation of our approximate algorithm.
- The empirical evaluation of a different weighting in the KLR objective to correct for an unbalanced distribution of the training data over the different classes.

All of these resulted in a practical and efficient classification algorithm suitable for speech data.

In this chapter we aim at deriving an all-at-once multi-class implementation. We therefore review the statistical setup but different to Section 3.4 we directly start from a multi-class setting and work towards one single optimization process opposed to commonly used decomposition techniques described in Section 3.7.

4.2 Modeling with multi-class kernel logistic regression

4.2.1 Linear multi-class logistic regression

At first, we give some insight in the precise statistical setup of (penalized) Logistic Regression (LR) [3], [80]. Suppose we have a random variable $(\mathbf{X}, \mathbf{Y})$ taking values in $\mathbb{R}^D \times \{1, \ldots, C\}$ where D denotes the dimensionality of the input vectors, and C denotes the number of classes. The aim of logistic regression is to produce an estimate of the a-posteriori probability of membership in each of the classes for the given random vector. The multi-class logistic model is given as

$$\begin{cases} Pr(\mathbf{Y} = 1 \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_1(x))}{\sum_{c=1}^{C} \exp(f_c(x))} \\ Pr(\mathbf{Y} = 2 \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_2(x))}{\sum_{c=1}^{C} \exp(f_c(x))} \\ \vdots \\ Pr(\mathbf{Y} = C \mid \mathbf{X} = x; f_1, ..., f_C) = \frac{\exp(f_C(x))}{\sum_{c=1}^{C} \exp(f_c(x))}, \end{cases} \tag{4.1}$$

where $f = \{f_c\}_{c=1}^{C}, f_c : \mathbb{R}^D \to \mathbb{R}$ consists of C different functions. In the linear case we assume that the functions can be written as

$$f_c(x) = w_{c\cdot}^T x + w_{c0}, \quad c = 1, \ldots, C, \tag{4.2}$$

where $w \in \mathbb{R}^{CD}$, $w = (w_{11}, w_{12}, \ldots, w_{1D}, \ldots, w_{C1}, \ldots, w_{CD})^T$ (see [1]), $w_{c.}$ refers to the elements of w linked to class c and w_{c0} is the intercept term of class c. Note that we will use this notation for all double indexed vectors as is explained in the introduction. For notational convenience in the sequel, unless noted otherwise, the x vectors are augmented with a value 1 such that the intercept term is incorporated in the parameter vector $w_{c.}$ as the first component.

Given the previous model, one can compute easily the class probabilities for a new sample $x^* \in \mathbb{R}^{D+1}$. If one is to single out a class belonging to x under the logistic model, the best one could do (achieving the Bayes risk), is the following prediction rule

$$\underset{c \in \{1,2,\ldots,C\}}{\arg\max} \quad Pr(\mathbf{Y} = c | \mathbf{X} = x^*; f_1, ..., f_C). \tag{4.3}$$

Now we are interested in how to infer the parameters (functions) of the logistic model given a set of observations of $(\mathbf{X}, \mathbf{Y})$. Specifically, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be a set consisting of N i.i.d. samples from the above logistic model. The classical method for doing inference of the statistical model is based on maximizing the likelihood

$$\ell(f_1, ..., f_C) = \prod_{i=1}^{N} Pr(\mathbf{Y} = y_i | \mathbf{X} = x_i; f_1, ..., f_C)^{\frac{1}{N}}, \tag{4.4}$$

over the unknown functions f_i. Since we will later consider functions which are very flexible, it was argued that a penalized version of maximum likelihood is more appropriate in [55]. The penalized negative log likelihood (PNLL) criterion is defined as follows [2]

$$\ell^{\nu}(f_1, ..., f_C) = -\frac{1}{N} \log \left(\prod_{i=1}^{N} Pr(\mathbf{Y} = y_i | \mathbf{X} = x_i; f_1, ..., f_C) \right) + \frac{\nu}{2} \sum_{c=1}^{C} w_{c.}^T w_{c.}, \tag{4.5}$$

[1] Because of the multi-class structures in this chapter we need a convenient way of representing data point and class indices at the same time for a certain vector as is done in [158]. For instance $w_{ci} \in \mathbb{R}^{CD}$ selects the i-th component linked to a model c. In case such a double indexed vector is used its components are ordered as $w = (w_{11}, \ldots, w_{1D}, \ldots, w_{C1}, \ldots, w_{CD})^T$ or in case a $\sim$ is used, e.g. $\tilde{p}$, component ordering is as follows $\tilde{p} = (\tilde{p}_{11}, \ldots, \tilde{p}_{C1}, \ldots, \tilde{p}_{1N}, \ldots, \tilde{p}_{CN})^T$. The notation "." indicates a full index set. For example $w_{c.}$ selects all components which correspond to the c index.

[2] For notational convenience we only used a single ν for all parameters including the intercept. The same reasoning holds for two parameter matrix $\Psi = \text{diag}\left(\left(\nu', \ 1^T \nu \right)^T \right)$ and corresponding regularization term $\sum_{c=1}^{C} w_{c.}^T \Psi w_{c.}$. Now, a separate hyper-parameter ν', which regularizes the intercepts $w_{.0}$, different from ν (which controls the regularization of the other parameters) is available.

where ν is a regularization parameter, which controls the bias-variance trade-off [66], that is tuned using techniques such as cross-validation. Let us now specialize to the logistic multi-class model and the functions f_c:

$$\ell_{LR}^{\nu}(w) = \frac{1}{N}\sum_{c=1}^{C}\sum_{i\in\mathcal{D}_c}\left(-w_{c.}^T x_i + \log(e^{w_{1.}^T x_i} + ... + e^{w_{C.}^T x_i})\right) + \frac{\nu}{2}\sum_{c=1}^{C} w_{c.}^T w_{c.}, \quad (4.6)$$

where $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_C$, $\mathcal{D}_i \cap \mathcal{D}_j = \varnothing, \forall i \neq j$ and $y_i = c, x_i \in \mathcal{D}_c$. In the sequel we use the shorthand notation

$$\tilde{p}_{ci} = \Pr(\mathbf{Y} = c|\mathbf{X} = x_i; w), \quad (4.7)$$

where $\tilde{p} = (\tilde{p}_{11}, \ldots, \tilde{p}_{C1}, \tilde{p}_{1N}, \ldots, \tilde{p}_{CN}), \tilde{p} \in \mathbb{R}^{CN}$ ($\sim$ indicates a specific type of component ordering which is different to that of e.g. w). Since $\log(\sum_{c=1}^{C} e^{w_{c.}^T x})$ is strictly convex (see [16]), the PNLL criterion for penalized logistic regression is strictly convex in the parameters w.

Notice that the penalization term in (4.5) uniquely identifies the parameters w. At the optimum the gradient of the regularized LR objective (4.6), as defined later in (4.17), is equal to zero ($\hat{\Phi}$ in (4.17) is replaced by X in the linear setting as will become clear later). We can therefore write $\sum_{c=1}^{C} g_{c.} = 0_D$ with 0_D a D dimensional vector of all zeros. Substituting g by (4.17) gives

$$\begin{array}{ll} & \sum_{c=1}^{C}\left(X^T(\tilde{p}_{c.} - \tilde{r}_{c.}) + \nu w_{c.}\right) = 0_D \\ \Rightarrow & X^T\left(\sum_{c=1}^{C}\tilde{r}_{c.} - \sum_{c=1}^{C}\tilde{p}_{c.}\right) = \nu\sum_{c=1}^{C} w_{c.}. \end{array} \quad (4.8)$$

By definition $\sum_{c=1}^{C}\tilde{r}_{ci} = 1$ (see later in (A.4)) and $\sum_{c=1}^{C}\tilde{p}_{ci} = 1$ (see (4.7)) which sets the left hand side to zero, such that the following symmetric constraint on the parameters w is obtained

$$\sum_{c=1}^{C} w_{c.} = 0_D. \quad (4.9)$$

Remark 15. In case no penalization is used as in (4.4) the parameters w are not unique: if we add a common term w' the probability estimates of the former model will not change. In the literature it is often seen that authors make the parameters identifiable by choosing a reference class. This is done by setting $w_{c^{ref}.} = 0_D$ where c^{ref} indicates the reference class. Now, the regression coefficient $w_{ci}, c \neq c^{ref}$ can be interpreted as the log odds ratio of class c versus the reference class c^{ref}. Unfortunately, the solution to this formulation depends on the specific choice of reference class [103].

4.2.2 Multi-class kernel logistic regression

In this section we describe a non-linear extension using kernels to previous models called Multi-class Kernel Logistic Regression (MKLR). Instead of using an appropriate *Representer Theorem* [105] for dealing with representing kernels, an optimization argument and associated primal-dual formulation is used. The latter formulation has the advantage that it can give insights on how to build approximate models in the primal space. In [93] it is shown that MKLR can be fit within a scheme of iteratively re-weighted Least Squares-SVMs (irLS-SVM). Note that in [202] the relation of KLR to Support Vector Machines (SVM) is stated. Next we summarize the primal-dual formulation. The details are given in Appendix A.

The objective function in (4.6) can be advanced with a nonlinear extension to kernel machines where the inputs x are mapped to a high dimensional space. Define $\varphi : \mathbb{R}^D \to \mathbb{R}^{D_\varphi}$ as the feature map induced by a positive semi-definite kernel. With the application of the Mercer's theorem for the kernel matrix Ω as $\Omega_{ij} = K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j), i, j = 1, \ldots, N$ it is not required to compute explicitly the nonlinear mapping $\varphi(\cdot)$ as this is done implicitly through the use of positive kernel functions K. Note that for notational convenience by writing $\varphi(x)$ a, by value one extended version of the transformed non-augmented vector x is meant. In the kernel version of LR the C models are defined as in equation (4.1) but x is mapped to a high dimensional space, i.e. $f_c(x) = w_{c.}^T\varphi(x)$. Note that to simplify the notation a single mapping $\varphi(\cdot)$ is used for each class, but that class-dependent mappings $\varphi_c(\cdot)$ are also possible.

The minimization of objective function (4.6) can be reformulated as an iteratively re-weighted least squares (IRWLS) problem, which is also described in [127], [153], and [202] . This gives the primal formulation at iteration k

$$\min_{s^{(k)}} \frac{1}{2}||Fs^{(k)} - \tilde{q}^{(k)}||^2_{W^{(k)}} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}), \quad (4.10)$$

where the definitions of weights matrix $W^{(k)}$, regressor vector $\tilde{q}^{(k)}$ and multi-class feature matrix F are respectively given in (A.6), (A.10), and (A.12) in Appendix A.

To conveniently write down a dual formulation of this problem we use the following notation for the multi-class kernel matrix $G = FF^T$. The solution to the MKLR problem can then be found by iteratively solving the linear system

$$\left(\frac{1}{\nu}G + (W^{(k)})^{-1}\right)\tilde{\alpha}^{(k)} = \tilde{q}^{(k)} + \frac{1}{\nu}G\tilde{\alpha}^{(k-1)}. \quad (4.11)$$

The probabilities of a new point x^* given by C different models can be predicted using (4.1) and

$$f_c(x^*) = w_{c.}^T \varphi(x^*) = \frac{1}{\nu} \sum_{i=1}^{N} \tilde{\alpha}_{ci}^{(L)} K(x_i, x^*), \tag{4.12}$$

where the $\tilde{\alpha}^{(L)}$ are the estimated parameters obtained at the final iteration L when minimizing objective function (4.10), and $\tilde{\alpha}^{(k)} = (\tilde{\alpha}_{11}^{(k)}, \tilde{\alpha}_{21}^{(k)}, \ldots, \tilde{\alpha}_{C1}^{(k)}, \ldots, \tilde{\alpha}_{1N}^{(k)}, \ldots, \tilde{\alpha}_{CN}^{(k)})^T$.

4.2.3 Correction for unbalanced data sets

For some learning problems it is preferable to correct the unbalance in the number of data examples in the different classes in order to find the model parameters w. The most common situation in which this correction is used is where the proportions of data examples in the different classes is known not to be representative of the class frequencies while the system is used. This problem can be addressed by re-balancing the training distribution, either by over-sampling the minority class, or generating artificial examples of the minority class [29], or by down-sampling the majority class. However, under-sampling may discard relevant pieces of information and over-sampling is not computationally efficient and might introduce bias. Another strategy consists in post-processing standard classification techniques, such as tuning a bias term after learning to correct for the original decision bias term [172], but this scheme fails to discover the changes in the shape or orientation of the decision boundary that may be required to isolate the minority class. We therefore apply different weights to the different classes. Note that a similar setup is used in for instance [26], [136]. This results in altering the objective function (4.6) as follows,

$$\begin{aligned} \ell_{LRb}^{\nu}(w) = & \sum_{c=1}^{C} \frac{1}{CN_c} \sum_{i \in \mathcal{D}_c} \left(-w_{c.}^T \varphi(x_i) + \log(e^{w_{1.}^T \varphi(x_i)} + \ldots + e^{w_{C.}^T \varphi(x_i)}) \right) \\ & + \frac{\nu}{2} \sum_{c=1}^{C} w_{c.}^T w_{c.}, \end{aligned} \tag{4.13}$$

where N_c are the local class frequencies.

4.2.4 Inference with a universum

Similar to the ideas given in Section 3.8 we can enhance the MKLR algorithm to incorporate universum knowledge by adding the term $\sum_{c=1}^{C}\sum_{i=1}^{|\mathcal{U}|} L(f_c(x_i))$ which favors maximal contradiction on universum data to the KLR objective in (4.10). The set $\mathcal{U}$ is called the universum and L denotes the loss function. KLR outcomes have a probabilistic interpretation, hence minimizing the distance of universum points to the decision boundary can be seen in this context along similar lines as maximizing the entropy of the MKLR model outcomes on the universum data. Maximum entropy is achieved when $\tilde{p}_{ci} = 1/C$ for $c = 1, ..., C$. This is the case when all function values $f_c(x), \forall x \in \mathcal{U}$ are equal (see (4.1)). This requirement is only met when $f_c(x) = 0$, for $c = 1, ..., C$ which can be seen in Section 4.2.1 where it is shown that L_2 norm constraint in (4.10) implies the symmetric constraint $\sum_{c=1}^{C} f_c(x) = 0$. Hence, minimizing $\sum_{c=1}^{C}\sum_{i=1}^{|\mathcal{U}|} L(f_c(x_i))$ results in maximizing the entropy and thus minimizing the distance of universum data to the decision boundary.

Several loss functions are possible. We choose here to use the L_2 loss function which gives:

$$\begin{aligned}\sum_{c=1}^{C}\sum_{x\in\mathcal{U}} L(f_c(x)) &= \sum_{c=1}^{C}\sum_{x\in\mathcal{U}} (f_c(x))^2 \\ &= \sum_{c=1}^{C} w_{c.}^T \sum_{x\in\mathcal{U}} \left(\varphi_c(x)\varphi_c(x)^T\right) w_{c.}. \end{aligned} \tag{4.14}$$

Note that in case of SVMs, adding a regularizer term results in a maximization of the margin. In KLR, adding regularization has similar properties [202]. Hence, by adding the term (4.14) we change the original objective for defining the decision boundary to a trade-off between the usual regularization on the training (non-universum) data and maximizing the entropy of the universum data. In the Section 4.4 we experimentally indicate that the inclusion of universum data can improve the accuracy in terms of correct classifications and the likelihood of test data, over labeled data alone.

4.2.5 Multi-class coding schemes

Deriving the multi-class implementation as described above results in one large (all-at-once) learning problem. In contrast to the approaches discussed in Section 3.7

MKLR has the advantage that no coding scheme has to be chosen. Since one-versus-all and one-versus-one coding schemes (see Section 3.7) are often used, we compare these to the all-at-once method in terms of computational training and evaluation complexity (see Section 4.3.4), classification performance, sparseness and evaluation speed (see Section 4.4).

4.3 Large-scale implementation

This section describes a collection of practical insights which were crucial to give our implementation of KLR the characteristics needed for speech data sets.

4.3.1 Fixed-size approach: estimation in the primal weight space

Similar to FS-LSSVM in Section 3.3.3 an approximation to the feature map $\varphi \in \mathbb{R}^{D_\varphi}$ using the Nyström method [189] can be calculated which means that the functions $f_c(x)$ can be approximated as follows

$$\begin{aligned}\hat{f}_c(x) &= \sum_{i=1}^{M} (w_{ci}\hat{\varphi}_i(x)) + w_{c1} \\ &= \sum_{i=1}^{M} \left(w'_{ci} \frac{\sqrt{M}}{\sqrt{\lambda_i^s}} \sum_{k=1}^{M} (u_i)_k K(x_k, x) \right) + w_{c1}. \end{aligned} \tag{4.15}$$

where $w'_{c(i-1)} = w_{ci}$ for $i = 2, \ldots, D_{\hat{\varphi}} + 1$ and λ_i^s and u_i the corresponding sample eigenvalues and eigenvectors of a subspace of kernel matrix Ω.

Prototype vector selection criterion: k-center clustering

In order to select M PVs different sampling strategies exist: random selection and Rényi entropy based selection were already discussed in Section 3.3.3. In this chapter we additionally experiment using a third approach based on k-center clustering. In [201] the observation is made that the Nyström low-rank approximation depends crucially on the quantization error induced by encoding the sample set with landmark points. This suggests that one can simply use the clusters obtained with a k-center (such as k-means) algorithm, which finds a local minimum of the quantization error (Note that using this technique the PVs do not necessarily coincidence with the training data). A simple and fast greedy algorithm which

approximates the k-center clustering problem is proposed in [71] and called farthest-point clustering. This algorithm works as follows: pick an arbitrary point $\bar{p}_0$ as the center of the first cluster and add it to the center set $\mathcal{P}$. Then, for $i = 1$ to k, do the following: at step i, for every point x, compute its distance to the set $\mathcal{P}$, $d(x_j, \mathcal{P}) = \min_{\bar{p}' \in \mathcal{P}} ||x_j - \bar{p}'||$. Set $\bar{p}_j = x_j$ and let x_j be the point that is farthest from $\mathcal{P}$, i.e., the point for which $d(x_j, \mathcal{P}) = \max_x d(x, \mathcal{P})$. Add $\bar{p}_i$ to set $\mathcal{P}$. Report the points $\bar{p}_0, \bar{p}_1, ..., \bar{p}_{k-1}$ as the cluster centers. Each point is assigned to its nearest center. A fast implementation in [58] has a complexity of $O(N \log k)$.

The active prototype vector selection methods described, suffer from the fact that they will select outliers as prototypes. In cases were M is relatively small, the fraction of prototypes chosen to represent the non-outlier and outlier data is unbalanced and therefore the classification performance will not be optimal. When the number of prototype vectors is increased, the performance will also increase to that of KLR. Hence taking into account outliers can result in a sparser model. To point out in Section 4.4 that removal of outliers as prototype vectors is beneficial we implemented a simple iterative scheme based on the k-center method, stated in Algorithm 2. An extra hyper-parameter ζ can be tuned using 10-fold cross-validation on the likelihood scores.

Algorithm 2 k-center with outlier removal

Input: The input data matrix $X \in \mathbb{R}^{N \times D}$ containing N data samples described by D attributes, M number of PVs, ζ tuning parameter.
Output: M prototype vectors.
Define: Index sets $\mathcal{S} = \{1, .., N\}$ and $\mathcal{S}^r = \emptyset$, $X(\mathcal{S})$ the rows of X selected by the index set $\mathcal{S}$, matrix with M prototype vectors $\mathcal{P} \in \mathbb{R}^{M \times D}, \mathcal{P}^T = (\bar{p}_i, \ldots, \bar{p}_M)$, $N_{\bar{p}_i}$ the number of data points belonging to the same cluster as prototype vector $\bar{p}_i$.
repeat
  Select M vectors out of $X(\mathcal{S})$ using the k-center method and store in $\mathcal{P}$.
  $i := 1$
  while $i <= M$ **do**
    $i := i + 1$
    if $N_{\bar{p}_i} < \frac{\zeta N}{M}$ **then**
      $\mathcal{S}^r := \mathcal{S}^r \bigcup \{$ *all* x_j *in the same cluster as* $\bar{p}_i\}$.
    end if
  end while
  $\mathcal{S} := \mathcal{S} \backslash \mathcal{S}_r$.
  Select X^p out of $X_{\mathcal{S}}$ using the k-center method.
until $\mathcal{S}^r \neq \emptyset$

Remark 16. Note that prototype vectors selected by the random and the entropy based methods are a subset of the training samples. The k-center method introduces an extra degree of freedom because now the prototype vectors do not necessarily have to coincidence with the training samples.

In a multi-class environment one can opt to sample stratified or not. Stratified PV sampling means that an equal number of PVs are selected for each of the classes separately. In the experimental section it is empirically investigated which option is preferable.

Selecting the number of prototype vectors

An important task in this framework is to determine the number of prototype vectors M which are used to build the FS-MKLR model. A large M gives more modeling flexibility but slower evaluation times than a smaller M which gives less complex models but faster evaluation times. To determine an M a cross-validation score can be computed for all values of M up till $M = N$. For each different value for M, the hyper-parameters (e.g. regularization parameter ν, kernel band-with σ) are tuned using cross-validation. The best cross-validation score per M is stored and used to select the smallest value of M within one tenth of the standard deviation of the best cross-validation score (standard deviation was calculated from the different estimates computed using cross-validation). See [78] (Section 7.10, page 216) where a similar approach, referred to as the "one-standard-error", is given. However, exploring all possible values for M would be computationally too expensive for large-scale data sets. Therefore, a set of predefined values for M is chosen. Note that in order to exclude random effects in dividing the data the same cross-validation partitioning is kept for computing the cross-validation estimates for each value of M.

To illustrate this process, a subset of the `timit` (see Appendix B) training set ($N = 4,613$) is selected i.e. the examples belonging to phone classes /b/ and /d/ (the feature extraction is explained later in Appendix B). Then for a set of M values 10-fold cross-validation scores (based on a Negative Log Likelihood (NLL) criterion) for a tuned FS-MKLR model are calculated. This is plotted in Figure 4.1 where both one tenth of the standard deviation and mean of the 10 cross-validation measurements (using tuned hyper-parameters) are given per M value. At a certain point adding more PVs does not improve the NLL based criterion which indicates that only a subset of the training points (called PVs) is needed to build a model. The final selected M is marked by the circle.

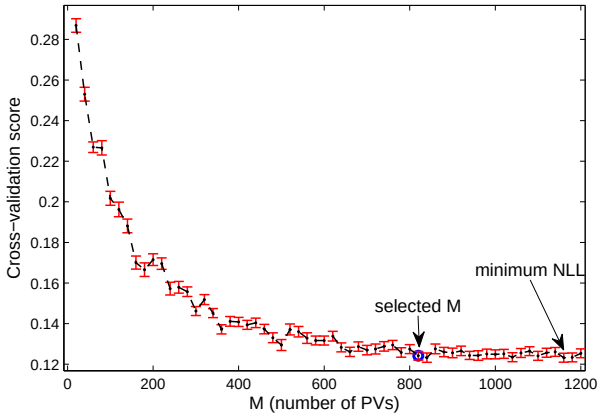


Figure 4.1: For a set of M values 10-fold cross-validation scores (based on a Negative Log Likelihood (NLL) criterion) are calculated for a tuned FS-MKLR model. On the figure both the mean and one tenth of the standard deviation of the 10 cross-validation measurements are given per M value. The final selected $M = 820$ is marked by the circle and was selected using the smallest value of M within one tenth of the standard deviation of the best cross-validation score (which is $M = 1,160$).

4.3.2 Solving the problem in primal space: Newton Trust Region approach

In this work the sparse kernel logistic regression problem is solved in the primal space and therefore optimization algorithms developed for linear logistic regression can be used. In the literature there are many different methods for solving the logistic regression problem. All these optimizations are iterative procedures which can be distinguished by the trade-off being made between cost per iteration and speed of convergence.

Given the finite dimensional approximation to the feature map $\hat{\varphi}$, which is defined as in (3.29) where instead of N only $M \ll N$ samples are used to approximate φ, the problem in (4.10) can be solved in the primal weight space with parameters w using the Newton's based method which iteratively updates the weight vector $w^{(k)}$,

$$w^{(k)} = w^{(k-1)} + \tau s^{(k)}, \tag{4.16}$$

over k until convergence where τ is the step size and $w^{(k)}$ the vector of all parameters in the k-th iteration. In each iteration the step $s^{(k)} = -{H^{(k)}}^{-1} g^{(k)}$ can be computed

where the gradient and Hessian are respectively defined as

$$g^{(k)} = \begin{pmatrix} \frac{\partial \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.}} \\ \vdots \\ \frac{\partial \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.}} \end{pmatrix},$$

and

$$H^{(k)} = \begin{pmatrix} \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.} \partial w^{(k)}_{1.}} & \cdots & \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{1.} \partial w^{(k)}_{C.}} \\ & \ddots & \\ \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.} \partial w^{(k)}_{1.}} & \cdots & \frac{\partial^2 \ell^{\nu}_{LR}}{\partial w^{(k)}_{C.} \partial w^{(k)}_{C.}} \end{pmatrix}.$$

Applied to (4.10) this gives

$$g^{(k)} = \begin{pmatrix} \hat{\Phi}^T(\tilde{p}^{(k)}_{1.} - \tilde{r}^{(k)}_{1.}) + \nu w^{(k-1)}_{1.} \\ \vdots \\ \hat{\Phi}^T(\tilde{p}^{(k)}_{C.} - \tilde{r}^{(k)}_{C.}) + \nu w^{(k-1)}_{C.} \end{pmatrix}, \tag{4.17}$$

with $\tilde{r}$ as defined in (A.4) and $\hat{\Phi} = (\hat{\varphi}(x_1), ..., \hat{\varphi}(x_N))^T, \hat{\Phi} \in \mathbb{R}^{N \times (M+1)}$,

$$H^{(k)} = \begin{pmatrix} \hat{\Phi}^T T^{(k)}_{11} \hat{\Phi} + \nu I & \cdots & \hat{\Phi}^T T^{(k)}_{1C} \hat{\Phi} \\ & \ddots & \\ \hat{\Phi}^T T^{(k)}_{C1} \hat{\Phi} & \cdots & \hat{\Phi}^T T^{(k)}_{CC} \hat{\Phi} + \nu I \end{pmatrix}, \tag{4.18}$$

where

$$T^{(k)}_{ab} = \operatorname{diag}\left(t^{ab}_i, \ldots, t^{ab}_N\right), \tag{4.19}$$

with

$$t^{ab}_i = \begin{cases} a = b & \tilde{p}^{(k)}_{ai}(1 - \tilde{p}^{(k)}_{bi}) \\ a \neq b & -\tilde{p}^{(k)}_{ai} \tilde{p}^{(k)}_{bi}. \end{cases} \tag{4.20}$$

For large scale multi-class data this matrix might be too large to be stored. The size of the Hessian is proportional to C and the length of $\hat{\varphi}(\cdot)$ which is $D_{\hat{\varphi}} + 1$. Therefore certain iterative methods where the Hessian is always used in a product with a vector d are more appropriate here. This fact can be used to exploit the structure of the Hessian and therefore storage of the full Hessian is not needed. For Newton methods the conjugate gradient (CG) algorithm is mostly used for this

purpose. Instead of performing all iterations until a stopping criterion is met, the loop can be interrupted earlier which is done in case of using truncated Newton methods as in [107]. Both issues mentioned above are tackled in the Newton Trust Region algorithm described in [119]. This algorithm yielded the best performance compared to state-of-the-art alternatives. A good balance between convergence speed and cost per iteration is found in that low cost approximate Newton steps are taken in the beginning of the algorithm and full Newton directions at the end for fast convergence. We therefore used their optimization structure and extended it to the multi-class case.

At each iteration a second order Taylor approximate of ℓ_{LR}^{ν} subject to a trust region constrained is minimized which results in the following optimization problem

$$\begin{aligned} \hat{s}^{(k)} = &\arg\min_{s^{(k)}} g^{(k)T} s^{(k)} + \frac{1}{2} s^{(k)T} H^{(k)} s^{(k)} \\ &\text{such that } ||s^{(k)}|| < \Delta^{(k)}, \end{aligned} \tag{4.21}$$

at iterate $w^{(k)}$ and a trust region $\Delta^{(k)}$ to obtain a step direction $\hat{s}^{(k)}$. Then $w^{(k)}$ is updated according to (4.16) and $\Delta^{(k)}$ is updated by checking the ratio

$$\rho^{(k)} = \frac{\ell_{LR}^{\nu}(w^{(k+1)}) - \ell_{LR}^{\nu}(w^{(k)})}{g^{(k)T}\hat{s}^{(k)} + \frac{1}{2}\hat{s}^{(k)T} H^{(k)} \hat{s}^{(k)}}, \tag{4.22}$$

of the actual reduction in the function to the predicted reduction in the quadratic model as in [119]. The direction is accepted if $\rho^{(k)}$ is large enough:

$$w^{(k+1)} = \begin{cases} w^{(k)} + \hat{s}^{(k)} & \text{if } \rho^{(k)} > \eta_0 \\ w^{(k)} & \text{if } \rho^{(k)} \leq \eta_0, \end{cases} \tag{4.23}$$

where $\eta_0 > 0$ is some predefined value. Note that $\hat{s}^{(k)}$ differs from the Newton step $s^{(k)}$ given in (4.16) since it is calculated using the quadratic model in (4.21) which uses the trust region constraint.

In each iteration it is necessary to compute the probabilities $\tilde{p}^{(k)}$ using (4.1) where

$$\hat{f}_c(x) = w_{c.}^T \hat{\varphi}(x), \tag{4.24}$$

using (4.15) this can be written as

$$\hat{f}_c(x) = \frac{1}{\nu} \sum_{i=1}^{M} \tilde{\alpha}_{ci} K(x_i, x) + w_{c1}, \tag{4.25}$$

with

$$\tilde{\alpha}_{c.} = SUw'_{c.}, \tag{4.26}$$

with $S = \Lambda^{-\frac{1}{2}}$ where Λ is the diagonal matrix with the eigenvalues of a subspace (size $M \times M$) of Ω and $U = (u_1, \ldots, u_M)^T$ the matrix with corresponding eigenvectors.

The Fixed-Size MKLR (FS-MKLR) trust region algorithm is showed in Algorithm 3. We terminate Algorithm 3 when the relative difference of the objective likelihood

Algorithm 3 FS-MKLR

Input: training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$,$M$
Parameters: $w^{(k)}$
Output: probabilities $\Pr(\mathbf{X} = x_i|\mathbf{Y} = y_i; w^{opt}), i = 1, ..., N$ and w^{opt} is the converged parameter vector
Initialize: $k := 0$,$w^{(0)} = 0_{CM}$
Define: $g^{(k)}$ and $H^{(k)}$ according to resp. (4.17) (4.18).
Selection of M PVs using k-center clustering
compute features $\hat{\Phi}$ as in (3.29)
repeat
  $k := k + 1$
  compute $\Pr(\mathbf{X} = x_i|\mathbf{Y} = y_i; w^{(k-1)}), i = 1, ..., N$
  calculate $g^{(k)}$
  $\hat{s}^{(k)} = \arg\min_{s^{(k)}} g^{(k)T} s^{(k)} + \frac{1}{2} s^{(k)T} H^{(k)} s^{(k)}$ such that $||s^{(k)}|| \leq \Delta^{(k)}$ as in (4.21)
  compute $\rho^{(k)}$
  $w^{(k+1)} = w^{(k)} + \hat{s}^{(k)}$
  obtain $\Delta^{(k+1)}$
until convergence

is smaller than some threshold, or when the number of iterations reaches some maximum.

To solve the constrained minimization problem of (4.21) CG is used similarly as in [119]. An outline of this method is given in Algorithm 4. The most CPU intensive task within linear CG is the product of the Hessian H and a vector $d = (d_{11}, \ldots, d_{1D}, \ldots, d_{C1}, \ldots, d_{CD})$ which can be efficiently implemented without explicitly storing the complete Hessian. Because the Hessian is structured as in

(4.18) the resulting vector can be written as follows

$$\begin{pmatrix} \hat{\Phi}^T \left(T_{11}(\hat{\Phi} d_{1.}) + ... + T_{1C}(\hat{\Phi} d_{C.}) \right) + \nu d_{1.} \\ \vdots \\ \hat{\Phi}^T \left(T_{C1}(\hat{\Phi} d_{1.}) + ... + T_{CC}(\hat{\Phi} d_{C.}) \right) + \nu d_{C.} \end{pmatrix}, \tag{4.27}$$

where T_{ij} is defined as in (4.19). In a practical implementation first all vectors $h_{c.} = \hat{\Phi} d_{c.}$ are calculated and stored for $c = 1, ..., C$ then each component of Hd is calculated. Notice that a matrix-vector product in case of diagonal matrices as is with T_{C1} can be efficiently implemented as follows $h'_{i'} = (\text{diag}(T_{ij}))_{i'} \, h_{ci'}$.

Algorithm 4 Conjugate Gradient algorithm for solving (4.21).

1: *Define*: $H^{(k)}$ and $g^{(k)}$ respectively as in (4.18) and (4.17) with X replaced by Φ_c
2: *Final estimate*: $\hat{s}^k$
3: *Initialize*: $\hat{s}^{(0)} = 0_{CM}, r'^{(0)} := -g^{(k)}, d^{(0)} = -r'^{(0)}, i := 0$
4: **repeat**
5: $h^{(i)} = H^{(k)} d^{(i)}$
6: $\beta = -\left(r'^{(i)T} r'^{(i)}\right) / \left(d^{(i)T} t^{(i)}\right)$
7: $\hat{s}^{(i+1)} = \hat{s}^{(i)} + \beta d^{(i)}$
8: if $||\hat{s}^{i+1}|| \leq \Delta^{(k)}$, compute τ such that
9: $||\hat{s}^i + \tau d^{(i)}|| = \Delta^{(k)}$
10: $r'^{(i+1)} = r'^{(i)} + \beta h^{(i)}$
11: $\kappa^{(i+1)} = \left(r'^{(i+1)T} r'^{(i+1)}\right) / \left(r'^{(i)T} r'^{(i)}\right)$
12: $d^{(i+1)} = -r'^{(i+1)} + \kappa^{(i+1)} d^{(i)}$
13: $i := i + 1$
14: **until** $||r'^{(i)}|| \leq \xi$

The stopping criterion on line (14) in Algorithm 4 implies that $\hat{s}^{(i)}$ is an approximate solution of the linear system (4.10). Although Algorithm 4 is different from standard CG because it is constrained. It is known that with $\hat{s}^{(0)} = 0$ [167], the following can be stated [3]

$$||\hat{s}^{(i)}|| < ||\hat{s}^{(i+1)}||, \forall i. \tag{4.28}$$

This proves the fact that in a certain number of conjugate gradient iterations either the (14) is satisfied or the trust-region constraint is violated.

[3] We use this fact to speed up the CG algorithm between Newton iterations. When an increase in objective function is detected (negative ρ) the trust-region is shrunken to a smaller value and a the CG loop is repeated with the same initial values. We can prevent re-computation by storing intermediate values of the previous CG loop.

4.3.3 Very large data sets

For very large data sets, storage of the full $\hat{\Phi} \in \mathbb{R}^{N \times M}$ matrix can become problematic. The fastest implementation first stores the complete $\hat{\Phi}$ matrix in memory and then starts training in primal space. However, it is not necessary to fully store this matrix. At the cost of more processor time one can save memory by recomputing the $\hat{\Phi}$ in blocks at times it is necessary in the algorithm. It can be seen that $\hat{\Phi}$ appears at two different locations in the algorithm. Firstly, it is used in computing the gradient g in the main Newton loop and secondly to compute the matrix vector product Hd in the CG algorithm.

Gradient computation

Let N_{b} be a predefined maximum number of transformed input vectors which are allowed to be stored at once. Given N_{b}, the matrix X can be divided in Z blocks defined by the index sets $\mathcal{B}_i, \forall i = 1, \ldots, Z$, where $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$. Based on the fact that a matrix vector product can be decomposed as follows $\hat{\Phi}^T a = \hat{\Phi}(\mathcal{B}_1)^T a(\mathcal{B}_1) + \ldots + \hat{\Phi}(\mathcal{B}_Z)^T a(\mathcal{B}_Z)$, the gradient as defined in (4.17) can be computed as indicated by the pseudo code in Algorithm 5. The features are now calculated in blocks and stored in a temporary matrix $\hat{\Phi}'^T$. In this case $\hat{\Phi}$ need not be fully stored at a cost of an increased computational cost due to recomputing features in $\hat{\Phi}'^T$.

Algorithm 5 Reduced memory implementation for gradient computation

Define: X is the matrix with input vectors, $\mathcal{B}_i$ is block index set, $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$ where Z equals the number of blocks, $a = \tilde{p} - \tilde{r}$
for $z = 1, ..., Z$ **do**
  compute features of $X(\mathcal{B}_z)$ and store in $\hat{\Phi}'$
  for $c = 1, ..., C$ **do**
    $g_{c.} := g_{c.} + \hat{\Phi}'^T a_{c.}(\mathcal{B}_z)$
  end for
end for
$g := g + \nu w$

Matrix-vector computation

Given the result in (4.27), a decomposition similar to that used in Algorithm 5, can be used to calculate the matrix-vector product Hd. This again reduces memory

requirements to calculate the result. The pseudo code is given in Algorithm 6.

Algorithm 6 Reduced memory implementation for matrix-vector product Hd

Define: $X \in \mathbb{R}^{N \times D}$ is the matrix of test input vectors, $\mathcal{B}_k$ is block index set, $\mathcal{B}_1 \cap \mathcal{B}_2 ... \cap \mathcal{B}_Z = \emptyset$ where Z is the number of blocks, vector $d = (d_{11}, \ldots, d_{1D}, \ldots, d_{C1}, \ldots, d_{CD})$, and H according to (4.18).
for $z = 1, ..., Z$ **do**
  compute features of $X(\mathcal{B}_z)$ and store in $\hat{\Phi}'$
  for $c = 1, ..., C$ **do**
    $a_i = \sum_{j=1}^{C} (T_{cj})_{ii} (\hat{\Phi}'^T d_{j.})_i, \forall i$
    $h_{c.} := h_{c.} + \hat{\Phi}' a$
  end for
end for
$h := h + \nu d$

4.3.4 Computational complexity

The computational cost of the FS-MKLR algorithm (without the feature map approximation) is dominated by the matrix vector product Hd which has a complexity of $\mathcal{O}(CM(2N+C+1))$. This means that the matrix-vector product and hence the training procedure scales linearly in M and N and scales quadratically in C. A complete training run requires $k_1 k_2$ matrix-vector products. Where k_1 is the number of Newton iterations and k_2 is the number of CG iterations in Algorithm 3.

The complexity of calculating the Nyström approximation (with eigenvalue decomposition of the kernel matrix of size M) is $\mathcal{O}(M^3 + M^2 N)$. When only one joint kernel is used (one feature map) and all features $\hat{\Phi}$ can be stored in memory this operation is only executed once independent of the number of classes. When the reduced memory implementation is used (see Algorithm 5, 6) the features are recomputed for each gradient and Hessian computation.

For a comparison with binary coupled classifiers we assume a joint kernel, that the features fit in memory, an equal number of training samples N_c and PVs M_c per class. The complexity term of FS-MKLR can then be written as $\mathcal{O}(C^2 M_c(2CN_c + C + 1)) = \mathcal{O}(C^3(2M_c N_c + ((C+1)/C)M_c))$. In case of using a one-versus-all coding scheme we have the following complexity $\mathcal{O}(C(2CM_c(2CN_c + 2 + 1)) = \mathcal{O}(2C^3(2M_c N_c + (3/C)M_c))$. Which is about 2 times more costly than that of FS-MKLR, since M_c is usually a lot smaller than N_c. However, in practice we have noticed that coefficients k_1 and k_2 are in most cases a little lower than those of FS-

MKLR. For the pairwise approach, this reduces to $\mathcal{O}(1/2C(C-1)(2M_c(4N_c+3))) = \mathcal{O}(2(C^2-C)(2M_cN_c+(3/2)M_c))$. Thus, using a one-versus-one approach results in a reduction of computational costs inversely proportional to the number of classes.

If training time is considered the one-versus-one approach seems to be more suitable for practical use than the other methods. However, if evaluation speed is considered then the other approaches are more interesting. A factor which dominates the evaluation speed is the number of unique PVs. Although a prototype vector may be associated to different binary classifiers, computer time is saved by calculating and storing all $K(x_i, x)$ first, where x_i are the selected unique prototype vectors and x is test data. Another factor is the total number of model parameters. This number will explode in case of one-versus-one type classifiers when the number of classes to consider increases. Due to this and the fact that FS-MKLR can produce sparser results with comparable accuracy as one-versus-one classifiers the evaluation speed of FS-MKLR models is much faster opposed to one-versus-one models for problems with a large number of classes such as in phone recognition. Since, in our implementation we force one-versus-all to have the same PVs for each of the class models, the same reasoning holds as for FS-MKLR.

Remark 17. If the number of universum data examples is modest, adding the penalization term (4.14) does not have a major impact on the overall training complexity. If we assume using a joint kernel the total costs are the calculation of the correlation matrix with complexity $\mathcal{O}(M|\mathcal{U}|^2 + M^2|\mathcal{U}|)$ (this is required only once at the beginning of the training procedure) and k_1 times $\mathcal{O}(CM^2)$ for computing the penality.

4.4 Experiments

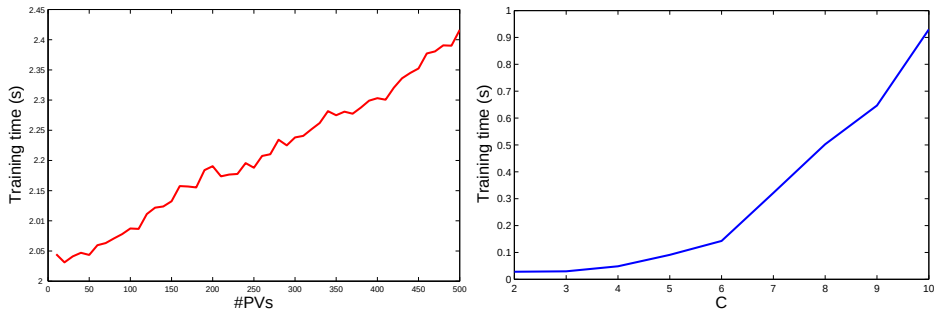
All FS-(M)KLR experiments in this section are carried out in MATLAB. We empirically test different design choices for the FS-(M)KLR algorithm discussed in the previous paragraphs and compare to state-of-the-art classification techniques. We opted to use a separate hyper-parameter ν' which controls the regularization for all the intercept terms. In our experiments ν' is set to an arbitrary small value 10^{-9}. The parameter ν, which regularizes all model weights, is tuned using cross-validation.

Deriving the multi-class implementation as described above results in one large (all-at-once) learning problem without the use of a coding scheme. Besides the all-at-once scheme also other possible multi-class implementations can be build by combining several independent binary classifiers via a common coding scheme

approach (e.g. one-versus-one, one-versus-all, see Section 3.7). In this section the all-at-once multi-class scheme is compared to the one-versus-one and one-versus-all coding scheme variants in terms of training complexity, classification performance, sparseness and evaluation speed using several public domain available data sets. The characteristics for each of the data sets, which are used to benchmark, are given in Appendix B. The FS-(M)KLR MATLAB software can be downloaded from `ftp://ftp.esat.kuleuven.be/pub/SISTA/karsmakers/software/FS-MKLR.zip`.

4.4.1 Training time

In Section 4.3.4 it was seen that FS-(M)KLR scales linearly with the number of PVs and quadratically with the number of classes C. In Fig. 4.2 this is verified by plotting the training time in function of the number of PVs in Fig. 4.2a and classes Fig. 4.2b on the `pb` data set.



(a) *(b)*

Figure 4.2: The training time in function of the number PVs is shown in (a) and in function of the number of classes in (b). For this experiment the Peterson and Barney data set is used.

4.4.2 Negative log likelihood in terms of number of classes

In Fig. 4.3 we plot the log likelihoods and accuracies in terms of correct classifications of test data produced by models inferred with three discriminative multi-class versions of FS-(M)KLR and a generative model trained with LDA (Linear Discriminant Analysis) as a function of the number of classes. The first multi-class model, which we here will refer to as FS-MKLR, is an all-at-once model as in (4.1), the second is built using binary subproblems coupled via a one-versus-all

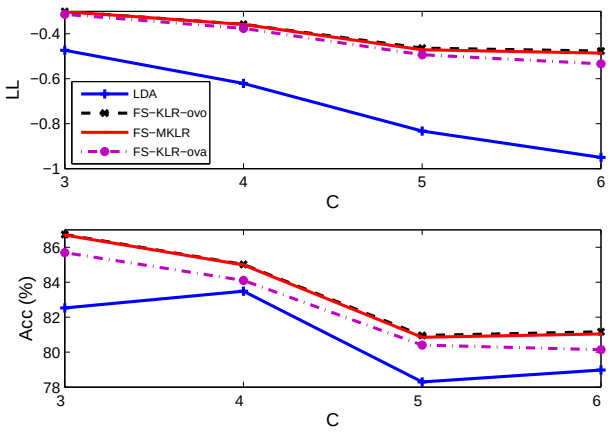


Figure 4.3: Two figures are plotted respectively presenting the mean of the log likelihood (LL) and classification accuracies on an independent test set of 20 runs in function of the number of classes in the learning problem. Classification techniques used were LDA (Linear Discriminant Analysis), FS-MKLR (Fixed-Size Multi-class KLR), FS-KLR-ovo and FS-KLR-ova binary Fixed-Size KLR models respectively coupled via a one-versus-one and one-versus-all coding scheme.

encoding scheme which we call FS-KLR-ova and the third is via an one-versus-one encoding scheme and is called FS-KLR-ovo. In order to be able to compare with LDA, which is a linear classifier, the linear kernel was used for all experiments. For this experiment we used a toy data set which consists of 600 data points in each of the C classes. The data in each class is generated by a mixture of 2 dimensional Gaussians. All classes are plotted in Fig. 4.4. Now, classes are added by increasing number. Each time a class is added, ν and M are tuned using a 10-fold cross-validation and the log likelihood and accuracy averaged over 20 runs are plotted. It can be seen that the FS-(M)KLR variants all performed better on an independent test set compared to LDA. As was expected FS-KLR-ova consistently scored worse than the other KLR alternatives because the linear kernel was used. Notice that FS-KLR-ovo and FS-MKLR have similar performance which is better than the alternatives. The difference is growing larger when the number of classes increases.

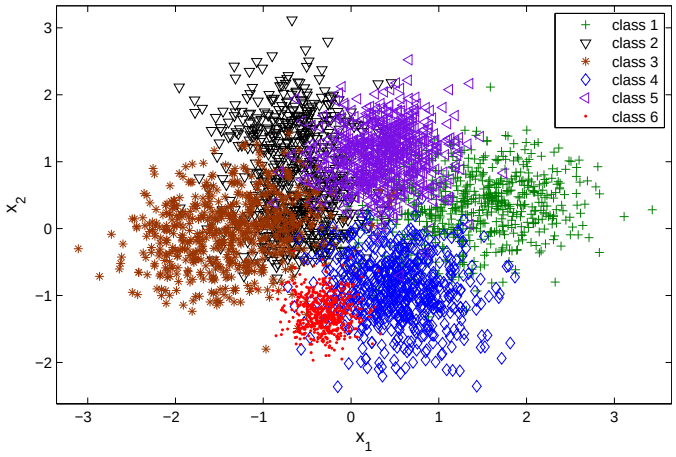


Figure 4.4: Artificial 2 dimensional multi-class ($C = 6$) toy data set. The data in each class was generated by a mixture of 2 Gaussians.

4.4.3 Prototype vector selection

Next, we assess the performance of different PV selection strategies on a suite of thirteen public domain benchmark data sets used in the study by [128]. The same set of 100 random partitions of the data (20 in the case of the `image` and `splice` benchmarks) to form training and test sets used in that study are also used here. Model selection is performed independently for each realization for the data set, such that the standard errors reflect the variability of both the training algorithm and the model selection procedure with changes in the sampling of the data. The RBF kernel as (3.1) was used for all algorithms. Three different PV selection procedures were used: *k-center* which refers to the farthest-point clustering algorithm [71], *renyi* which refers to the method using a Renyi entropy criterion described in [172] [18] and *random* which refers to random selection of the PVs. Table 4.1 shows a comparison of the error rates and log likelihood scores of FS-MKLR using different PV selection methods. In order to compare the different methods we arbitrarily set the parameter M to 5% of training set size. It can be seen that the *k-center* method results for 8 out of 13 cases in the best score. When the *renyi* method has the highest result it is not significantly better than the *k-center* method. Another observation is that the active sampling is outperforming the random PV selection except for the data sets `image` and `german`. A closer look at the data learns that this is due to outliers in the data and the relative low number of PVs, which was 5% of the training data. As an example we have plotted a 2-D projection of the `image` data onto the first two principal components

in Fig. 4.5. The same projection was used to show the selected PVs (square marks) by the *k-center* method. It is seen that a relatively large part of the PVs are wasted to represent the outliers while only a relative small amount of PVs is used to represent the data space where most data examples lay in. The same effect is seen in case of using the *renyi* method. Using the k-center clustering with outlier removal algorithm described in Algorithm 2, we redid the experiments on the `image` and `german` data sets. The tuning parameter ζ is tuned using 10-fold cross-validation on the likelihood scores. The results are shown in 4.2. It is seen that these results are now better than using random PV selection. The results for all other data sets are within the range of those without the removal of outliers. We can now state that the use of active sampling produces better results than just random PV selection but in case outliers are present special treatment is needed. Since, the *k-center* option is preferred in most cases we will use this for all coming experiments.

	k-center		renyi		random	
	err	log likelihood	err	log likelihood	err	log likelihood
banana	**10.63**(±**0.5**)	*-0.2463*(±*0.0101*)	10.72(±0.5)	−0.2501(±0.0128)	10.68(±0.52)	−0.2501(±0.0114)
breast-cancer	27.52(±4.71)	*-0.5426*(±*0.0485*)	**27.42**(±**4.52**)	*-0.5436*(±*0.047*)	27.94(±4.53)	−0.5473(±0.0492)
diabetis	**23.23**(±**1.83**)	−0.4841(±0.0186)	23.3(±1.84)	−0.4856(±0.0184)	23.6(±1.68)	*-0.4823*(±*0.0177*)
flare-solar	33.67(±1.65)	*-0.5725*(±*0.0139*)	**33.62**(±**1.61**)	−0.573(±0.0136)	34.02(±1.88)	−0.5731(±0.0166)
german	24.29(±1.99)	−0.4961(±0.0235)	24.24(±2.18)	−0.4966(±0.0244)	**24.07**(±**2.15**)	*-0.4954*(±*0.024*)
heart	**16.7**(±**3.55**)	*-0.4071*(±*0.0582*)	18.33(±4.16)	−0.4295(±0.0648)	17.75(±3.49)	−0.4167(±0.0548)
image	8.31(±0.97)	−0.2143(±0.0322)	8.37(±1.14)	−0.2496(±0.1617)	**4.71**(±**0.74**)	*-0.1223*(±*0.0162*)
ringnorm	**1.85**(±**0.28**)	−0.0575(±0.0156)	2.29(±0.34)	−0.0654(±0.009)	1.91(±0.3)	*-0.0569*(±*0.0089*)
splice	**16.61**(±**1.03**)	*-0.3743*(±*0.0137*)	17.78(±1.31)	−0.3926(±0.0183)	17.3(±0.92)	−0.3821(±0.0154)
thyroid	**4.95**(±**2.23**)	*-0.1388*(±*0.1029*)	5.67(±2.72)	−0.17(±0.0954)	5.36(±2.63)	−0.1482(±0.0973)
titanic	22.71(±0.98)	*-0.5336*(±*0.039*)	**22.62**(±**0.86**)	−0.6377(±0.8776)	23.13(±1.43)	−0.5727(±0.1812)
twonorm	**2.79**(±**0.32**)	*-0.0769*(±*0.0072*)	2.94(±0.33)	−0.0808(±0.0126)	2.99(±0.31)	−0.0805(±0.0082)
waveform	**10.13**(±**0.56**)	*-0.224*(±*0.0105*)	10.93(±0.47)	−0.2402(±0.0084)	10.92(±0.53)	−0.2402(±0.0105)

Table 4.1: Error rates of FS-MKLR using different PV selection methods over thirteen benchmark data sets. Parameter is M is set to 5% of training set size. Three different PV selection procedures: *k-center* refers to the Farthest-point clustering algorithm [71], *renyi* refers to the method using a Renyi entropy criterion described in [172] [18] and *random* refers to random selection of the PVs. The results for each method are presented in the in the form of the mean error rate over test data for 100 realizations of each data set (20 in the case of the `image` and `splice` data sets), along with the associated standard error. The best error rates are shown in bold face and the best log likelihood scores in italics.

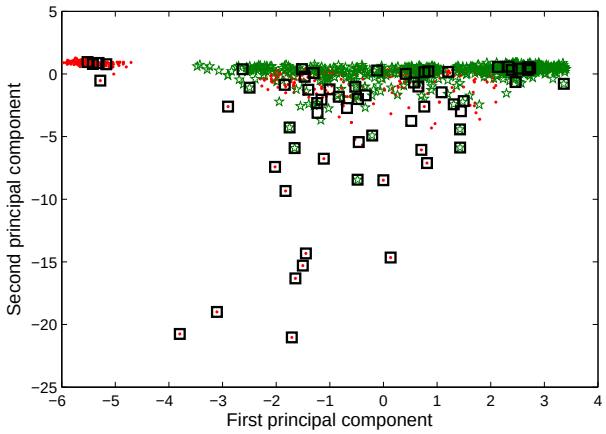


Figure 4.5: The figure shows a 2-D projection of the image data onto the first two principal components. The squares denote the selected PVs (which might coincidence with training data but do not necessarily have to).

Besides choosing which PVs to select there one also has to decide upon the complexity parameter M discussed in the theoretical part. This parameter controls the amount of prototype vectors used to build the FS-(M)KLR model. To assess model performance for different settings of M the `dna` and `satimage` data sets were used. In Fig. 4.6 the classification accuracy on test examples for the algorithms FS-MKLR and FS-KLR-ovo using a linear kernel in function of the number of selected prototype vectors is shown. Because the linear kernel was used the scores produced by the FS-KLR-ova classifier were much worse than those of the alternatives and

	k-center	
	err	log likelihood
german	23.89($\pm$2.07)	$-0.4939(\pm 0.0246)$
image	4.09($\pm$0.6)	$-0.1145(\pm 0.0193)$

Table 4.2: Error rates of FS-MKLR using *k-center* clustering method with outlier removal on the `german` and `image` data sets. Parameter is M is set to 5% of training set size. The results are presented in the in the form of the mean error rate over test data for 100 realizations of the `german` data set and 20 in the case of the `image`, along with the associated standard error.

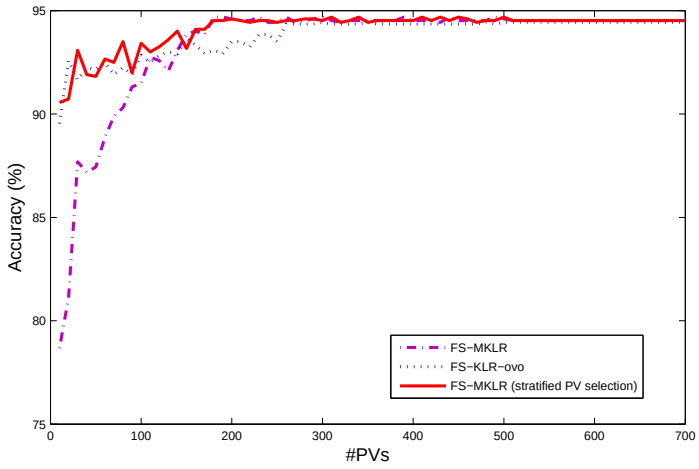


Figure 4.6: The figure shows the amount of correctly classified test examples in function of the number of selected support vectors for the `dna` data set. FS-MKLR, FS-KLR-ovo with a linear kernel are used.

were therefore excluded from the figure. The results for the `satimage` data set are plotted in Fig. 4.7. In this case we opted to use the RBF kernel. For each given M the hyper-parameter ν is tuned via 10-fold cross-validation after this tuning step the classification accuracy on the separate test set is computed. In case of using a one-versus-one coding scheme it is easier from a implementation perspective to select the PVs using stratified selection. For FS-MKLR we have tested both options stratified and not stratified. Note that in case of stratified selection the same PVs are used for all KLR related methods. For both experiments we see that the curves saturate to about the same maximum score but FS-MKLR obtains this maximum for sparser models than FS-KLR-ovo does. Note that a similar observation was made for multi-class all-at-once SVMs (multi-class problem is solved using one SVM formulation) which produce sparser results than binary coupled SVMs in [81]. In the remainder of this work the stratified PV sampling option will be used since it is preferred for both experiments.

Instead of exploring all possible values of M we use the simple procedure which is described in Section 4.3.1. We executed this procedure on the same suite of 13 public domain data sets as was used before. A range of values $[0.01N, 0.03N, 0.04N, 0.05N, 0.07N, 0.1N, 0.15N, 0.2N, 0.25N]$ which were floored and where N is the training set size, were chosen as candidate values for M. After tuning an FS-MKLR model was trained using the tuned hyper-parameters and finally the classification performance in terms of percentage of misclassifications

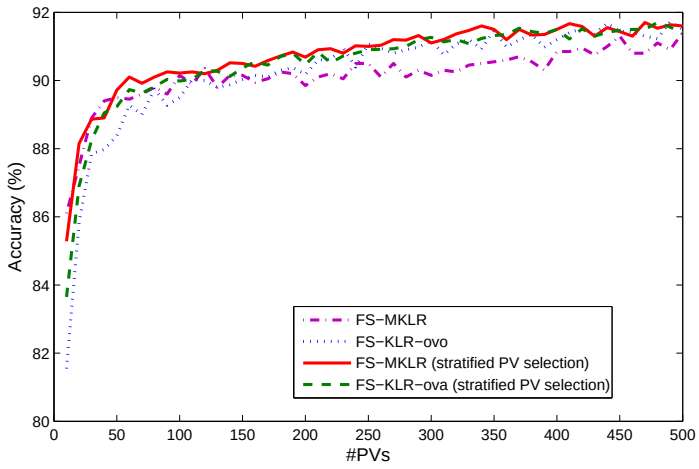


Figure 4.7: The figure shows the amount of correctly classified test examples in function of the number of selected support vectors for the `satimage` data set. The results for FS-MKLR, FS-KLR-ovo and FS-KLR-ova using an RBF-kernel are used. For FS-MKLR stratified as well as not-stratified PV sampling was tested. For all the others only the stratified selection was used.

on an independent test set was evaluated. This was repeated for 100 realizations of each data set (20 in the case of the `image` and `splice` data sets). The results are presented in the in the form of the mean error rate over test data along with the associated standard error as was the case before in Table 4.3. To compare the model sparsity as well as the misclassification rate we added the results of SVM and a sparse SVM version referred to as SpSVM-2 in [101] (the authors presented no results on `image` and `splice`). Note that our results for (non-sparse) KLR are comparable to those reported in [28]. A first observation is that FS-MKLR and the non-sparse KLR have comparable error rates. Although the misclassification rate is similar, FS-MKLR produces far more sparser results than the SVM. The cardinality is comparable to that of the sparse SVM method SpSVM-2.

	FS-MKLR		**KLR**	**SpSVM-2** [101]		**SVM** [101]	
	err (%)	M	err (%)	err (%)	# Basis	err (%)	n_{SV}
`banana`	10.68(±0.5)	20.8(±7)	10.6(±0.47)	10.87(±1.74)	17.3(±7.3)	10.54(±0.68)	211.7(±66.98)
`breast-cancer`	27.32(±4.6)	4.3(±4.1)	26.86(±4.67)	29.22(±2.11)	12.1(±5.6)	28.18(±3.00)	185.8(±16.44)
`diabetis`	23.57(±1.93)	14.3(±1.4)	23.4(±1.78)	23.47(±1.36)	13.8(±5.6)	23.73(±1.24)	426.3(±26.91)
`flare-solar`	33.65(±1.57)	19(±0)	34.2(±1.7)	33.90(±1.10)	8.4(±1.2)	33.98(±1.26)	629.4(±29.43)
`german`	23.77(±2.20)	20(±4.7)	23.72(±2.14)	24.90(±1.50)	14.0(±7.3)	24.47(±1.97)	630.4(±22.48)
`heart`	16.55(±3.55)	7.5(±2.2)	16.8(±3.10)	15.50(±1.10)	4.3(±2.6)	15.80(±2.20)	166.6(±8.75)
`image`	2.97(±0.49)	135.9(±35.2)	3.10(±0.52)	-	-	-	-
`ringnorm`	1.98(±0.29)	26.6(±19.7)	1.71(±0.15)	1.97(±0.57)	12.9(±2.0)	1.68(±0.24)	334.9(±108.54)
`splice`	13.22(±0.93)	147(±41.4)	11.43(±0.73)	-	-	-	-
`thyroid`	4.48(±2.25)	18.6(±7.2)	4.53(±2.25)	5.47(±0.78)	10.6(±2.3)	4.93(±2.18)	57.80(±39.61)
`titanic`	23.55(±1.61)	3.7(±2.2)	22.6(±0.09)	22.68(±1.88)	3.3(±0.9)	22.35(±0.67)	150(±0.0)
`twonorm`	2.8(±0.35)	11.2(±10.2)	2.76(±0.13)	2.96(±0.82)	8.7(±3.7)	2.42(±0.42)	330.30(±137.02)
`waveform`	10.18(±0.60)	16.5(±12.9)	9.73(±0.48)	10.66(±0.99)	14.4(±3.3)	10.04(±0.67)	246.9(±57.80)

Table 4.3: Error rates of FS-MKLR, standard (non-sparse) KLR, SVM and a sparse version of SVM denoted by SpSVM-2 over thirteen benchmark data sets. In case of FS-MKLR parameter M was chosen via 10 fold cross-validation out of a set of values $(0.01N, 0.03N, 0.04N, 0.05N, 0.07N, 0.1N, 0.15N, 0.2N, 0.25N)$, which were floored and where N is the training set size. k-center clustering was used as the PV selection procedure. The results for each method are presented in the in the form of the mean error rate over test data for 100 realizations of each data set (20 in the case of the image and splice data sets), along with the associated standard error. The results for SVM as well as those for SpSVM-2 are obtained from [101].

4.4.4 Balancing and tuning multiple hyper-parameters

In this section we compare the performance of different multi-class FS-(M)KLR models using different algorithmic parameters on several real-world publicly available data sets. For each of the experiments the RBF kernel was used and except for attribute standardization no preprocessing was done. Note that the training data points for `satimage` and `usps` are respectively distributed over the different classes as follows: $(1072, 479, 961, 415, 470, 1038)$, $(1194, 1005, 731, 658, 652, 556, 664, 645, 542, 644)$. Each tuple contains an entry containing the number of data samples in a specific class. The `isolet` data has balanced classes.

The following parameters are tested:

- **Balanced training and/or model selection criterion:** The balanced training method minimizes the objective function given in (4.13). To infer the hyper-parameters we evaluate which likelihood criterion, balanced or not-balanced, is preferred. The balanced likelihood criterion is defined as the sum of the likelihoods of each of the individual classes divided by their class frequencies.
- **Multiple or single kernel(s):** In case of binary coupled classifiers we additionally evaluated the choice of using different or a single set of hyper-parameters per class model.

Hyper-parameters are tuned using a 5-fold cross-validation criterion which is employed for each method. Results here are averaged over 5 randomly drawn partitions of the training set (the same partitions are used for the different methods). The results on the independent test sets are shown in Table 4.4. The column parameters is a trinary tuple containing as a first entry the codes SS and MS respectively standing for single set or multiple sets of hyper-parameters. The second and third entry respectively point out if a balanced training method and balanced selection criterion is used or not. The algorithmic setup was set such that the conditions for each of the algorithms was as similar as possible.

If we consider the `satimage` task, multiple hyper-parameter sets seem to be beneficial for the FS-KLR-ova approach, a balanced selection criterion seem to only have an impact on the misclassification rate when multiple hyper-parameter sets are used. For the `USPS` data set, balancing the training set is consistently better for all methods except one (i.e. FS-KLR-ova, using parameters (MS,yes,yes)) if accuracy is used as an evaluation criterion. This is not the case if the log likelihoods

are evaluated. The data in the `isolet` data set is balanced. This is reflected in Table (4.4) as for methods FS-MKLR and FS-KLR-ovo balancing has no effect. There is still a minor difference because of the randomness in the tuning process. On the other hand using a one-versus-all strategy creates C highly unbalanced classes but balancing seems to have a negative influence for that data set.

The following conclusions were drawn:

- Balancing the training and/or model selection does not give a significant advantage.
- Although we expected to always achieve better results when tuning multiple sets of hyper-parameters, the results indicate that this is not always the case. This is possibly due to the fact that tuning hyper-parameters per binary classifier never evaluates multi-class performance in the model selection process but instead only optimizes the hyper-parameters locally.
- A disadvantage of using multiple sets of hyper-parameters is on the one hand an increased training time and on the other hand a slower evaluation since different kernels are used per class. An important observation is that, when using FS-MKLR, one consistently obtains similar or better likelihood scores compared to the others.

Note that we did not aim at obtaining the highest possible scores for as we wanted to make a fair comparison between different settings for our algorithms. But nevertheless we next state some state-of-the-art figures which we found in the literature. In [158] standard KLR (non-sparse) experiments were performed. The author obtained for the KLR-ova, using 12 hyper parameters 8.00%($\pm$0.1%) and for MKLR, using two hyper-parameters 8.35%($\pm$0.2%). These results are comparable to ours. Unbalanced FS-MKLR using one hyper-parameter set has a misclassification rate of 8.41%($\pm$0.04%) and FS-KLR-ova using multiple hyper-parameter sets has a result of 8.21%($\pm$0.1%). For the `usps` data set in [158] a test error rate of 4.77%($\pm$0.18%) was reported for MKLR using different hyper-parameter sets for each of the classes. Our experiment which possibly is most alike is that of FS-KLR-ova without balancing and with multiple hyper-parameter sets presents a score of 5.01%($\pm$0.08%). Although, we approximate the KLR solution we still obtain comparable results to the standard formulation with the advantage that the FS-(M)KLR models are sparse. In [98] the authors reported 3.02% misclassification rate for the full `isolet` task using binary Import Vector Machine (IVM) classifier coupled via a one-versus-one coding scheme and 2.98% for standard KLR. Note that the FS-KLR-ovo model used to generate the results

in Table 4.4 only uses 19 PVs per class which results in an average number of PVs per classification pair of 38. The IVM has an average of 59.5 kernel functions.

Method	**Parameters**	`satimage`		`usps`		`isolet`	
		err (%)	llh	err (%)	llh	err (%)	llh
FS-MKLR	SS,no, no	8.41(±0.04)	−0.2261(±0.0004)	5.34(±0.05)	−0.185(±0.0006)	4.04(±0)	−0.1292(±0.0004)
FS-MKLR	SS,no,yes	8.36(±0)	−0.227(±0.0005)	5.32(±0.05)	**-0.182**(±0.0004)	3.98(±0.05)	−0.1289(±0.0003)
FS-MKLR	SS,yes,no	8.70(±0.179)	−0.2327(±0.0008)	5.26(±0.06)	−0.1859(±0.0009)	4.03(±0.03)	**-0.1283**(±0.0001)
FS-MKLR	SS,yes,yes	8.61(±0.09)	−0.2365(±0.0005)	5.25(±0.03)	−0.1858(±0.0006)	4.02(±0.04)	−0.1293(±0.0004)
FS-KLR-one-versus-all	MS,no,no	8.21(±0.1)	−0.2195(±0.0003)	5.01(±0.08)	−0.2033(±0.0023)	3.87(±0.11)	−0.1502(±0.002)
FS-KLR-one-versus-all	MS,no,yes	**8.07**(±0.06)	−0.2213(±0.0014)	4.94(±0.05)	−0.2047(±0.0028)	4.09(±0.17)	−0.1498(±0.0019)
FS-KLR-one-versus-all	MS,yes,no	8.19(±0.07)	−0.2291(±0.0031)	4.95(±0.03)	−0.2068(±0.0041)	3.91(±0.09)	−0.1681(±0.0038)
FS-KLR-one-versus-all	MS,yes,yes	8.6(±0.18)	−0.2476(±0.0021)	5.27(±0.11)	−0.2046(±0.0038)	4.13(±0.1)	−0.2502(±0.008)
FS-KLR-one-versus-all	SS,no,no	8.67(±0.06)	−0.2291(±0.0012)	5.27(±0.11)	−0.2046(±0.0038)	**3.69**(±0.06)	−0.1444(±0.0005)
FS-KLR-one-versus-all	SS,no,yes	8.71(±0.09)	−0.2241(±0.0006)	5.31(±0.03)	−0.1945(±0.001)	4.02(±0.06)	−0.1475(±0.0006)
FS-KLR-one-versus-all	SS,yes,no	8.41(±0.07)	−0.2308(±0.0008)	5.16(±0.03)	−0.195(±0.0011)	3.94(±0.04)	−0.1894(±0.0011)
FS-KLR-one-versus-all	SS,yes,yes	8.61(±0.11)	−0.2309(±0.0009)	4.97(±0.04)	−0.2009(±0.0012)	5.49(±0.22)	−0.3992(±0.0108)
FS-KLR-one-versus-one	MS,no,no	8.67(±0.19)	−0.2216(±0.0043)	5.13(±0)	−0.203(±0.0036)	4.34(±0.1)	−0.1403(±0.0013)
FS-KLR-one-versus-one	MS,no,yes	8.31(±0.07)	**-0.2127**(±0.0008)	5.01(±0.11)	−0.2078(±0.0017)	4.25(±0.12)	−0.1356(±0.0015)
FS-KLR-one-versus-one	MS,yes,no	9(±0.09)	−0.2222(±0.0005)	4.92(±0.09)	−0.2177(±0.0017)	4.37(±0.05)	−0.1402(±0.0024)
FS-KLR-one-versus-one	MS,yes,yes	9.03(±0.09)	−0.2241(±0.003)	4.94(±0.05)	−0.2174(±0.0028)	4.35(±0.05)	−0.1342(±0.0024)
FS-KLR-one-versus-one	SS,no,no	8.66(±0.02)	−0.2316(±0.0007)	4.96(±0.03)	−0.1902(±0.0008)	4.22(±0.03)	−0.1468(±0.0005)
FS-KLR-one-versus-one	SS,no,yes	8.73(±0.04)	−0.2315(±0.0005)	5.01(±0.03)	−0.1936(±0.0008)	4.23(±0.05)	−0.1478(±0.0004)
FS-KLR-one-versus-one	SS,yes,no	9.25(±0)	−0.2426(±0.0002)	**4.73**(±0.09)	−0.193(±0.0017)	4.21(±0.1)	−0.1431(±0.0006)
FS-KLR-one-versus-one	SS,yes,yes	9.37(±0.03)	−0.241(±0.0004)	4.87(±0.04)	−0.192(±0.0017)	4.20(±0.09)	−0.1421(±0.0005)

Table 4.4: Scores of different multi-class FS-(M)KLR models using different algorithmic parameters are shown. The column parameters is a trinary tuple containing as a first entry the codes SS and MS respectively standing for single set or multiple sets of hyper-parameters. The second and third entry respectively point out if a balanced training method and balanced selection criterion is used or not. The algorithmic setup was set such that the conditions for each of the algorithms was as similar as possible. Hyper-parameters are tuned using a 5-fold cross-validation criterion which is employed for each method. Results here are averaged over 5 randomly drawn partitions of the training set (the same partitions are used for the different methods). For each of the experiments the RBF kernel was used and except for attribute standardization no preprocessing was done.

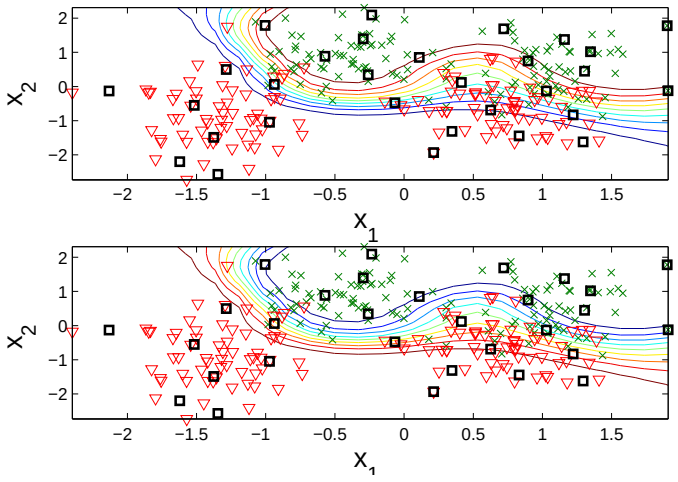
4.4.5 Include universum data

In this section we illustrate potential advantages of using the universum penalty explained in Section 4.2.4 using two experiments

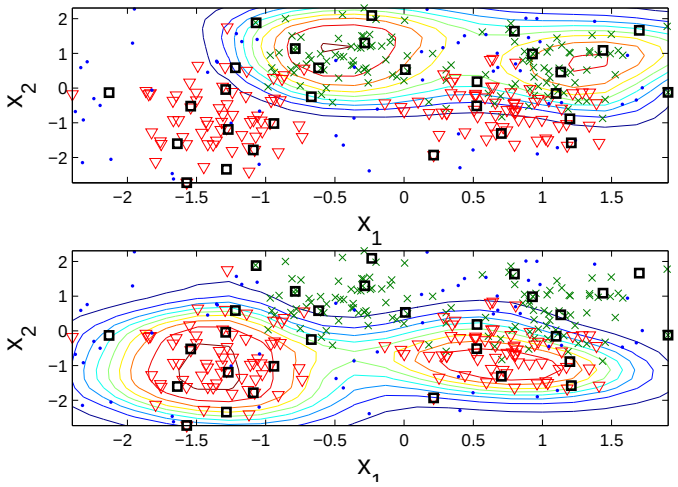
- Firstly, an FS-MKLR model is trained and tuned on Ripley's synthetic data [151] where we took uniformly distributed noise as universum data. Using the universum penalization term described in (4.14) we try to maximize the likelihood of the training data as well as minimizing the likelihood of the universum data as much as possible. The RBF-kernel is used. Without the information of the universum class we obtain a probability landscape for the first and second class respectively presented in the first and second row of Fig. 4.8a. In case we include the universum data and choose an appropriate tuning parameter $\nu^{\mathcal{U}}$, we obtain more compact probability landscapes as given in Fig. 4.8b. Note that the universum data is presented with dots.
- Secondly, instead of using random data universum examples should be collected to reflect information about the domain of our problem of interest. In the next experiment we build a binary classifier which has to discriminate between two letters selected from the `isolet` [179] data set. The remaining letters were used as universum data points. We selected 5 pairwise classification problems which produced the most confusion when considering the confusion matrix of a model obtained in a previous experiment. The RBF-kernel was used and 200 PVs were selected via the k-center method. Hyper-parameters σ^2,ν,$\nu^{\mathcal{U}}$ were tuned with NLL as the performance criterion. Results where averaged over 5 runs. The partitions as well as the selected PVs are kept constant to make a fair comparison possible between experiments with and without universum data. The results are shown in Table 4.5. For FS-MKLR$_U$ we observe that the misclassification rates in 4 out of 5 cases are similar or even better compared to FS-MKLR. If we consider the NLL score then FS-MKLR$_U$ has better scores for all cases.

4.5 Conclusion

We presented a practical and scalable implementation for Kernel Logistic Regression which exhibits characteristics that make it appropriate to be used in speech recognition architectures. We tested the algorithm on several smaller data sets. The main conclusions are:



(a)



(b)

Figure 4.8: In (a) the probability landscape of a tuned FS-MKLR on the Ripley data set is presented. The probabilities of the first and second class are respectively presented in the first and second row. When incorporating uniformly distributed noise as universum data into the model we obtained the result shown in (b). For both experiments an RBF kernel was used.

Table 4.5: The table shows results for FS-MKLR and FS-MKLR$_{\mathcal{U}}$ using the same 200 PVs on a binary `isolet` classification problem. In case of FS-MKLR$_{\mathcal{U}}$ the other letters not selected as the binary couple were used as universum data. The hyper-parameters were tuned using the negative log likelihood as a criterion. Column err and llh represent respectively the misclassification rate and log likelihood on test data available for the two considered classes.

data	**FS-MKLR**		**FS-MKLR$_{\mathcal{U}}$**	
	$Err(\%)$	NLL	$Err(\%)$	NLL
N-M	11.93(±0.70)	0.2326(±0.0059)	**10.92**(±**0.59**)	*0.227*(±*0.0068*)
P-T	**5.00**(±**0.00**)	0.2013(±0.0046)	**5.00**(±**0.00**)	*0.1442*(±*0.0472*)
F-S	3.33(±0.59)	0.0901(±0.0022)	**3.00**(±**0.75**)	*0.0801*(±*0.0051*)
V-B	**4.33**(±**0.37**)	0.122(±0.0022)	4.50(±0.75)	*0.1191*(±*0.0028*)
B-D	**2.50**(±**0.00**)	0.0776(±0.0068)	**2.50**(±**0.00**)	*0.0734*(±*0.014*)

- k-center clustering with outlier removal as an active prototype vector selection method outperformed the other alternatives for KLR. It was also seen that stratified selection is the preferred option in a multi-class environment.
- Compared to SVM, FS-MKLR produces models which are much sparser while having only a slightly worse classification accuracy and comparable recognition rate.
- Out of three different multi-class formulations we saw that the KLR all-at-once formulation (FS-MKLR) achieves the sparsest results. In terms of classification performance no scheme was consistently better than the others. However, in terms of likelihood the FS-MKLR alternative was consistently similar or better in case the same number of hyper-parameters was used as its alternatives.
- While being computationally more demanding the use of multiple sets of hyper-parameters did not have a major impact on classification performance.
- The use of a weighted version of the KLR objective to account for a correction for unbalanced data sets did not give a significant advantage.
- Adding universum data consistently improved the likelihood scores compared to building a model without including universum information.

Chapter 5

Sparse Conjugate Directions Pursuit with application to FS-LSSVM

This chapter studies an optimization scheme resulting in a sparse approximate solution of an over-determined linear system. Sparse Conjugate Directions Pursuit (SCDP) aims to construct a solution as good as possible using only a small number of non-zero (i.e. nonsparse) coefficients. One of the motivations of this work can be found in a setting of machine learning where sparse coefficients (a model as simple as possible) typically exhibit better generalization performance and lead to fast evaluations. The main idea is to build up iteratively a conjugate set of vectors of increasing cardinality, in each iteration solving a small linear subsystem. By exploiting the structure of this conjugate basis, an algorithm is found (i) converging in at most D iterations for D-dimensional systems, (ii) with computational complexity close to the classical conjugate gradient algorithm, and (iii) which is especially efficient when a few iterations suffice to produce a good approximation. As an example the application of SCDP to fixed size LS-SVM (FS-LSSVM) is discussed resulting in a scheme which efficiently finds a good model size (M) for the FS-LSSVM setting (note that the same ideas can be applied to FS-MKLR). Given a tuning-parameter set, the SCDP algorithm can efficiently compute FS-LSSVM models for increasing sizes. Further discussion includes algorithmic features such as scalability, component selection criteria (addressing the question which data points to include in the model), influence of additional tuning-parameters,

determination of a suitable stopping criterion and classification performances in the multi-class setting. In Section 5.1 the problem setting is sketched. Different approaches to solve the problem defined earlier are discussed in Section 5.2. This literature study points out that strong connections to the family of algorithms in the context of matching pursuit are present. Then details about the implementation of SCDP are given in Section 5.3 and the application of the resulting algorithm to the LS-SVM framework is given in Section 5.4. The final algorithm is validated on several public available data sets as reported in Section 5.5 and conclusions are drawn in Section 5.6.

5.1 Introduction

Consider integers $D > 0, N > 0$ and a task of determining a vector of parameters $w \in \mathbb{R}^D$ such that

$$Xw = y, \tag{5.1}$$

where $y \in \mathbb{R}^N$ is a target vector and the matrix $X \in \mathbb{R}^{N \times D}$ typically contains N measurements of size D. As illustrated in Fig. 5.1, three problem cases can be distinguished:

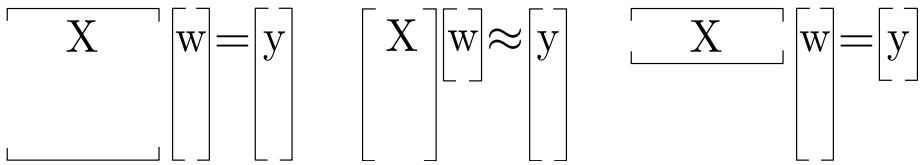
- **Uniquely-determined linear system ($N = D$):** When X is of full rank and $N = D$, a single unique solution exists.
- **Over-determined linear system ($N > D$):** When X is of full rank and $D < N$, no solution exist.
- **Under-determined linear system ($D > N$):** When X is of full rank and $D > N$, infinitely many solutions exist.

Linear systems are omnipresent in different computational sciences. This chapter focusses, but is not restricted, to over-determined systems originating from least squares estimation problems. Since these over-determined system in general cannot be exactly solved, one typically prefers the solution solving the system (5.1) with minimal least squares residuals $r = Xw - y$, or

$$w^* = \arg\min_w \frac{1}{2} \|Xw - y\|_2^2, \tag{5.2}$$

which is equivalent to solving the normal equations defined as

$$Aw = b, \tag{5.3}$$



(a) (b) (c)

Figure 5.1: The figure illustrates in (a) a uniquely-determined linear system with an equal number of dimensions D and measurements N (b) an over-determined linear system with $N > D$, and (c) an under-determined linear system with $D > N$.

where $A = X^T X \in \mathbb{R}^{D \times D}$ and $b = X^T y \in \mathbb{R}^D$. Several methods have been developed for solving such sets of linear equations. See the reference works [133] and citations. One of the most successful iterative algorithm for solving such large linear systems is the Conjugate Gradient (CG) algorithm. The linear conjugate gradient method was first proposed by Hestenes and Stiefel [85] in the 1950s as an iterative method for solving linear systems with positive definite matrices. Later on, the same ideas were used for solving non-convex optimization problems, notably in a context of artificial neural networks [129].

The aim of the present chapter is to device a computationally efficient algorithm which gives a sparse approximate solution of such a (possibly large-scale) system. Motivations to obtain sparse approximative solutions are found in many applications. Examples include:

- Zeros in the parameters yield interpretation of some sort. For example, in a standard estimation problem, sparse coefficients indicate non-relevant feature vectors. In signal processing a related view is found in the context of sparse signal reconstruction, where a sparse signal is constructed from a small number of basis signals selected from a large dictionary.
- In machine learning applications zeros in the model parameters often lead to better generalization performance.

Our main goal can be formalized as follows

Definition 5.1.1 (Sparse Approximative Solution)**.** *Let $\delta > 0$ denote an appropriate error tolerance then we seek the vector $w^{0,*} \in \mathbb{R}^D$ such that*

$$w^{0,*} = \arg\min_w \|w\|_0 \quad \textit{such that} \quad \|Xw - y\|_2^2 \leq \delta. \tag{5.4}$$

In the signal processing community only recently a new area of sparse modeling of signals and images emerged [22]. Although the focus is on under-determined systems, many algorithms are also applicable to (5.4). Consider $X \in \mathbb{R}^{N \times D}$ with $N \ll D$, in this case the under-determined system $Xw = y$ has infinitely many solutions. To narrow down the choice, one can include requirements for solution w such as to additionally minimize the sparsity of w using the L_0-norm. This gives

$$w^{0,*} = \arg\min_{w} \|w\|_0 \text{ such that } \quad Xw = y. \tag{5.5}$$

Often the exact constraint $Xw = y$ is relaxed to tolerate a slight discrepancy between Xw and y. In case the L_2 norm is used for evaluating the error the approximation of the optimization problem in the under-determined setting becomes equal to that of (5.4).

After positioning the considered approach to tackle problems like (5.4) in the literature of statistics, signal processing and machine learning strong connections to matching pursuit [125] type of algorithms and more specifically to the ideas in [121], [13] were found. The main novelty with regard to this work is therefore in the application part:

- A conceptual novel integration of the conjugate gradient algorithm with matching pursuit heuristics.
- The implementation of a new algorithm which approximatively solves the LS-SVM problem, resulting in a sparse model structure and relate to existing alternatives.
- A comparison of the new sparse LS-SVM method to the classical LS-SVM in terms of training complexity, classification performance, sparseness.
- An empirical comparison of different alternatives which iteratively solve the FS-LSSVM linear system with increasing complexity in terms of computer speed and misclassification rate.
- The extension of the binary classifier (i) to generate outcomes with probabilistic interpretation and (ii) to operate in the multi-class setting.

5.2 Heuristics to obtain a sparse approximation

Since the optimization task in (5.4) is combinatorial in nature, a good heuristic for solving that problem is needed. Starting from a model obtained using

all components, Backward Stepwise Regression (BSR) [78] sequentially deletes components from the solution. Since one of the goals of this work is to tackle large scale problems for which solving the full problem might be unfeasible, backward selection techniques are disregarded. Instead techniques applicable to large scale data are considered. A good survey of such techniques is given in [22]. We now proceed by briefly summarizing the key techniques, indicate relations and state which method we will further use throughout this chapter. Note that in the remainder we assume an appropriate scaling on the columns of X (the *features* of the measurements) in order to avoid bias in selecting components.

Greedy heuristic

A first approach is to handle the problem heuristically as defined in (5.4) by greedy, stepwise algorithms. A greedy strategy avoids exhaustive search for solving (5.4) in favor of a series of locally optimal single-term updates. In general, the approximation error is always reduced as much as possible given the starting approximation and the fact that only one term can be selected at one iteration. This explains the name "greedy". Examples include:

- A simple version in this class is called forward stepwise regression (FSR) in the statistical community. This is equivalent to Matching Pursuit (MP) in the signal processing and wavelet community [125]. A principal scheme is shown in Algorithm 7. MP (FSR) decomposes any y (e.g. signal) into a linear expansion of components (e.g. waveforms) that are selected from X (e.g a redundant dictionary of waveforms). Consider the normal equation in (5.3). The algorithm starts with an empty *active set* (which is a set of indices associated to selected components) and a solution vector $w^{(1)} = 0$, $0_N = (0, \ldots, 0)^T \in \mathbb{R}^D$ and then iteratively increments the cardinality of $w^{(k)}$. Given a collection of possible components, the one having the largest absolute correlation with the least squares residuals $c^{(k)} = Aw^{(k)} - b = X^T r^{(k)}$ (see [1]) is selected, and denoted as $(c^{(k)})_j$ (see [2]). The corresponding element in the parameter vector $(w)_j$ is then updated such that $(c^{(k+1)})_j = 0$. After calculating $c^{(k+1)}$ the process is repeated. The algorithm is stopped when an appropriate stopping criterion is reached, possibly resulting in a sparse parameter vector $w^{(k)}$.

[1] In case we mention residuals we indicate the residues from a linear system which in this chapter can be $r = Xw - y$ or $c = Aw - b$. Which linear system is considered should be clear from the context.

[2] Notation $(x)_j$ selects the j-th element of x.

- Because in previous mentioned greedy algorithms only a single element of $w^{(k)}$ is updated in each iteration, in case of MP (FSR) the set of components selected in each iteration ($k \geq 2$) is suboptimal in least squares sense and so are the corresponding elements of $w^{(k)}$. This can be corrected in each step using back-fitting (i.e. solving the linear system using the components selected thus far), what is known as Orthogonal Matching Pursuit (OMP) [140]. This is slightly different from Orthogonal Least Squares (OLS) [31] which has a different component selection strategy. While being computational more demanding, often more accurate results are obtained.

- Since greedy strategies add only a single component at each iteration to the active set, they require at least as many iterations as there are non-zero elements in the solutions vector w. A recently proposed promising variant of OMP is called stagewise OMP [48] which selects more than one component at each iteration using a thresholding procedure (iterative thresholding algorithm). Such approach can result in reducing the final number of iterations, but require slightly more work in each iteration.

- MP (FSR) is a greedy fitting technique that can be overly greedy, perhaps eliminating at the next step useful components that happen to be correlated with the already selected components. Forward Stagewise Regression (FStR) is a more cautious version of FSR, which follows the same algorithm except that it only takes a fraction (using a shrinkage factor) of the FSR step. This results in smaller steps which may increase the computation time.

Algorithm 7 Matching Pursuit.

Define: Normal equations $Aw = b$ originating from $\min_w \|Xw - y\|_2^2$
Final Estimate: $w^{(k)}$
Initialize: $w^{(1)} = 0_D$, $A = (a_1, \ldots, a_N)^T$, $c^{(1)} = Aw^{(1)} - b = -b$, $\mathcal{A} = \emptyset$, $k := 1$
repeat
  $j = \arg\max_i(|(c^{(k)})_i|)$, $\mathcal{A} = \mathcal{A} \bigcup \{j\}$
  $\beta = (c^{(k)})_j / (a_j)_j$
  $(w^{(k+1)})_j = (w^{(k)})_j + \beta$
  $c^{(k+1)} = c^{(k)} - a_j \beta$
  $k := k + 1$
until Stopping criterion is reached

L_1-norm heuristic

Another approach is to relax the zero norm $\|\cdot\|_0$ to the convex $\|\cdot\|_1$, resulting in the class of L_1-regularized algorithms. Here, one solves the problem

$$w^{*,1} = \arg\min_w \|w\|_1 \qquad \text{such that} \quad \|Xw - y\|_2^2 \leq \delta, \tag{5.6}$$

which is equivalent to the optimization problem

$$\min_{w,t} \frac{1}{2} \|Xw - y\|_2^2 + \lambda \sum_{i=1}^{N} t_i \quad \text{such that} \quad -t \leq w \leq t. \tag{5.7}$$

This type of relaxation is used in for instance the LASSO [175][3], for the purpose of model selection in statistics or Basis Pursuit (BP) [32] with applications such as compressed sensing (e.g. [24, 47]) and sparse signal recovery in signal processing (e.g. [49]). Because of the characteristics of these constraints, the formulation tends to produce coefficients which are exactly zero. Several special purpose solvers can be utilized to solve (5.7) such as

- **Iteratively re-weighted Least Squares (IRLS) algorithms:** by defining an appropriate Langragian, (5.7) can be stated as an unconstrained minimization problem which can be solved by IRLS in case we write $\|w\|_1 \equiv w^T W^\dagger w$ where $W = \text{diag}(|W|)$, $\dagger$ denotes a pseudo-inverse. (see e.g. [39]).
- **Iterative shrinkage methods:** for large scale problems an approximate strategy is more realistic. Iterative shrinkage methods iteratively use a simple one dimensional operation that set small entries to zeros (a *shrinkage* operation). These methods can compete with greedy methods both in simplicity and efficiency but are not yet as mature as the greedy alternatives (see [22] and the reference therein for a discussion).
- **Stepwise algorithms:** are heuristic methods, inspired by L_1 solvers, which iteratively add or remove components from the active set. Examples are (Least Angle RegreSsion) LARS [54] and the homotopy method [135]. Consider the problem (5.7). In [54], and [135] is was observed that the regularization path (i.e. the sequence of solutions, as λ varies from 0 to ∞) is piecewise linear, changing only at critical values of λ. Each vertex on this regularization path indicates a different sparse solution, i.e. vectors having non-zero elements only on a subset of the potential components. At each vertex the active

[3] LASSO was mainly developed in the over-determined case.

set is updated through the addition and removal of components. Based on this observation an heuristic algorithm was developed that, starting from an empty active set, iteratively adds and removes components. Note the difference with the greedy approaches which can only increase the number of indexes in the active set. Once a component is in the active set it remains there.

- **Dedicated solvers:** by exploiting the structure of the resulting quadratic programming, efficient dedicated solvers are described, see e.g. [104] which uses a specialized interior-point algorithm, or [59] where the authors propose to use an efficient gradient projection algorithm.

Remark 18. Opposed to the greedy techniques, using an L_1 penalty introduces an often undesirable bias. As a consequence (i) the intermediate solutions are not exactly equal to the least squares estimates (ii) component selection is based on a biased estimator. To overcome the former a *de-bias* step can be opted for. After finding an approximate solution, the de-biased solution is found by fixing the zero elements of w at zero and doing an unconstrained minimization of $Aw - b$ over the non-zero elements. In other words, having selected the components, a reduced linear system is solved in these components alone.

Relations

In [54] the LARS algorithm was proposed. LARS is a less greedy algorithm than OMP (FSR with back-fitting), it uses a similar strategy but only uses as much of a component as it "deserves":

- Consider the normal equations defined in (5.3) and let the index set of currently selected components be $\mathcal{A} = \{s_1, \ldots, s_{N_A}\}$, the set of not yet selected components $\bar{\mathcal{A}}$, selection matrix $\Sigma_{\mathcal{A}} = (e_{s_1}, \ldots, e_{s_{N_A}}), \Sigma_{\mathcal{A}} \in \mathbb{R}^{D \times N_A}$ where $e_i \in \mathbb{R}^D$ the i-th unit vector of size D, and a submatrix $A_{\mathcal{A}} = \Sigma_{\mathcal{A}}^T A \Sigma_{\mathcal{A}}$ of A.
- Start by selecting the j^*-th column of X which correlates the most with $Xw^{(1)} - y$ or $j^* = \arg\max_j |b_j|$ (since $w^{(1)} = 0$).
- The parameter vector $w^{(2)}$ is changed[4] according to the direction defined by the joint least squares solution of the currently selected components

[4] In [54] the authors observed that both LASSO and FStR behaved very similar. In an attempt to reduce the number of iterations of FStR, the authors defined LARS which "optimizes" the small FStR steps to larger steps, but smaller than the FSR stepsize, which greatly reduces the computer processing time.

$A_{\mathcal{A}}\Sigma_{\mathcal{A}}^T w^{(2)} - \Sigma_{\mathcal{A}}^T b = 0$ until a competitor $j \in \bar{\mathcal{A}}$ has as much correlation with the current least squares residuals ($|A_{\mathcal{A}}\Sigma_{\mathcal{A}}^T w^{(2)} - \Sigma_{\mathcal{A}}^T b|)_i = |e_j^T A \Sigma_{\mathcal{A}} \Sigma_{\mathcal{A}}^T w^{(2)} - e_j^T b|$ for $i \in \{1, \ldots, N_A\}$.

- Then this new component is inserted in $\mathcal{A}$ and removed from $\bar{\mathcal{A}}$, and the process is continued.

The following observations bridged the gap between L_1-minimization and greedy OMP [49] [54]:

- Only a simple modification to LARS (when a nonzero coefficient hits zero, drop it from $\mathcal{A}$ and recompute the current joint least squares direction to LARS) is needed to make LARS exactly reproduce the LASSO regularization path (the resulting algorithm is very similar to the homotopy method described in [135]).
- LARS and OMP are based on the same fundamental principle, which solves a sequence of least-squares problems on an increasingly large subspace, defined by the active set.

This helps explaining the similarity of both approaches in several theoretical and numerical studies (e.g. [178], [22]). Since its simplicity and the fact that OMP has similar behavior as the more sophisticated L_1-norm based optimization method we now proceed defining an efficient algorithm for (greedy) OMP and show in the experimental section that it can outperform a L_1 related alternative in terms of speed while having comparable accuracy.

Fig. 5.2 schematically summarizes the different approaches to tackle (5.4).

5.3 Greedy heuristic: Sparse Conjugate Directions

This section describes a new approach to solve (5.4) which belongs to the category of greedy heuristics.

The idea is to adapt CG such that the iteratively constructed conjugate basis yields solutions of increasing cardinality. In case the algorithm can be terminated in less than D steps a sparse solution is obtained. Adapting CG in this way results in an efficient implementation of OMP, which is a greedy based heuristic to solve (5.4) as was pointed out in previous section.

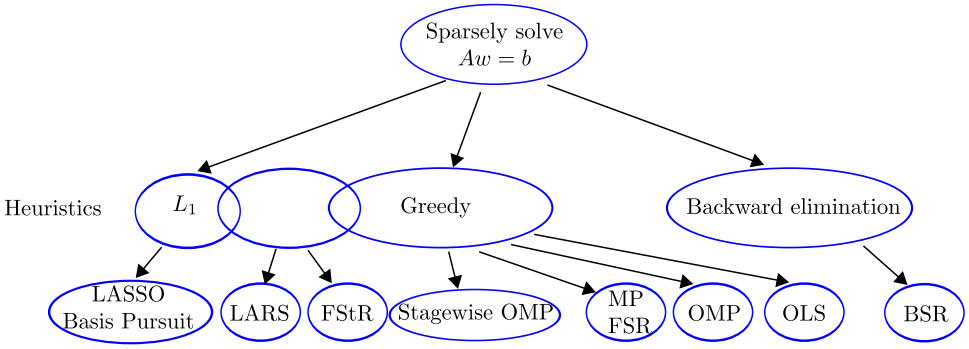


Figure 5.2: Overview of different approaches to tackle (5.4).

5.3.1 Conjugate Gradient Method

Consider a linear system with a symmetric positive definite matrix, such as in (5.4). This linear system can be written as an optimization problem as follows

$$\hat{w} = \arg\min_{w} \Psi(w) = \frac{1}{2} w^T A w - b^T w. \tag{5.8}$$

Now the first order condition for optimality coincides with the linear system, as the gradient $c \in \mathbb{R}^D$ becomes

$$\nabla \Psi(w) = Aw - b = c. \tag{5.9}$$

We now state two interesting properties of CG without proof, further details can be found in [132]. One of the remarkable properties of the conjugate gradient method is its ability to generate very cheaply a set of conjugate vectors. A set of non-zero vectors $\{p^{(1)}, p^{(2)}, ..., p^{(D)}\} \subset \mathbb{R}^D$ (see [5]) is said to be conjugate with respect to the positive definite matrix A if and only if

$$p^{(i)T} A p^{(j)} = 0, \quad \forall i \neq j. \tag{5.10}$$

Therefore, given a starting point $w^{(1)} \in \mathbb{R}^D$ and a set of conjugate vectors $\{p^{(1)}, p^{(2)}, ..., p^{(D)}\}$ one can solve the linear system by iteratively minimizing over the individual directions by setting

$$w^{(k)} = w^{(k-1)} + \eta^{(k)} p^{(k)}, \tag{5.11}$$

[5] For simplicity we first assume that all conjugate directions are given beforehand. However, we already denote the different conjugate directions via $p^{(k)}$ since the set will be iteratively generated in the final algorithm such that $p^{(k)}$ is generated in the k-th iteration.

where $\eta^{(k)}$ is the minimizer along the direction $w^{(k)} = w^{(k-1)} + \eta p^{(k)}$ and the $^{(k)}$ denotes the dependency of the variable on the iteration number k. This term is given explicitly as

$$\eta^{(k)} = -\frac{c^{(k)T} p^{(k)}}{p^{(k)T} A p^{(k)}}, \tag{5.12}$$

exploiting the conjugate property. Note that $c^{(k)}$ is defined as c in (5.9). Since for a matrix $A \in \mathbb{R}^{D \times D}$, only D different conjugate vectors can be constructed, the following result follows immediately:

Proposition 5. *[132] The CG algorithm converges to solution $w^{(L)}$ of (5.8) in at most D steps.*

For example, if the Hessian A would be diagonal then the subspace spanned by the unit vectors $\{e_{\pi(1)}, e_{\pi(2)}, ..., e_{\pi(k)}\}$ ($e_{\pi(i)} \in \mathbb{R}^D$ is the i-th unit vector of size D) would be minimized after k one-dimensional minimizations along the corresponding unit vectors directions. An analogous reasoning can be performed when working with a non-diagonal matrix A and a set of conjugate directions.

Proposition 6. *[132] Given $k-1$ conjugate vectors $p^{(i)}, i = 1, \ldots, k-1$, $k \leq D$, and suppose that $\{w^{(i)}\}_{i=1}^{k}, \{c^{(i)}\}_{i=1}^{k}$, and $\{c^{(i)}\}_{i=1}^{k}$ are generated by CG applied to (5.8) then*

$$c^{(k)T} p^{(i)} = 0, \quad \forall i = 1, ..., k-1, \tag{5.13}$$

and $w^{(k)}$ minimizes the objective of (5.8) over the vectors in set $\text{span}\{w^{(1)}, p^{(1)}, p^{(2)}, ..., p^{(k-1)}\}$.

Then, a key idea is to use a small number of iterations which produces a good solution. This is achieved by incrementally constructing the conjugate basis set according to the highest remaining gradient of the optimization problem. This is computed in a clever memory-friendly way, by using the remarkable property that a conjugate vector $p^{(k+1)}$ can be computed based on the previous conjugate vector $p^{(k)}$ only. Therefore, we can write $p^{(k+1)} = -c_k + \xi^{(k+1)} p^{(k)}$ where the new search direction is determined by a linear combination of the steepest descent (negative gradient) direction and the previous search direction. An outline of CG is shown in Algorithm 8.

As explained earlier the conjugate gradient algorithm will give the exact solution in at most D iterations. Now, in most conditions one converges indeed using a small number of steps. In e.g. textbooks [70], [132] this is formalized by the following

Algorithm 8 The standard Conjugate Gradient algorithm for solving $Aw = b$.

Define: $A = A^T \succ 0, b \in \mathbb{R}^D$
Final estimate: w^k
Initialize: $w^{(1)} = 0_D, c^{(1)} := -b, p^{(1)} = -c^{(1)}, k := 1$
repeat
  $\eta^{(k)} = -\frac{c^{(k)T} p^{(k)}}{p^{(k)T} A p^{(k)}}$
  $w^{(k+1)} = w^{(k)} + \eta^{(k)} p^{(k)}$
  $c^{(k+1)} = c^{(k)} + \eta^{(k)} A p^{(k)}$
  $\xi^{(k+1)} = \frac{c^{(k+1)T} c^{(k+1)}}{c^{(k)T} c^{(k)}}$
  $p^{(k+1)} = -c^{(k+1)} + \xi^{(k+1)} p^{(k)}$
  $k := k + 1$
until Stopping criterion is met

Proposition 7. *[132] If A has only V distinct eigenvalues, then CG will give the exact solution in at most V iterations.*

Alternatively, if the eigenspectrum of A occurs in V distinct clusters it is generally true that CG will approximately solve the problem in about V iterations [132]. These results indicate that CG, in case some conditions are met, can obtain the exact solution in a significantly smaller number of iteration than D. In case this property is preserved while using a sparse conjugate directions basis, a sparse (approximate) solution can be obtained. The following section discusses a modified version of CG which efficiently computes a sparse conjugate basis and, if stopped early, gives a sparse solution.

5.3.2 Modified CG using Sparse Conjugate Directions

As mentioned previously our goal is to formulate an algorithm which approximately solves the constrained minimization problem in (5.4). The algorithm goes as follows: The idea is to adapt CG such that it can produce sparse solutions by using a set of sparse conjugate vectors. Given $w^{(1)} = 0_D$ and sparse conjugate vectors (starting with cardinality one) for which in each step the cardinality is increased by one, sparse solutions are obtained. The conjugacy property is preserved so that at most D iterations are needed and the update does always decrease the norm of the residual. If the algorithm is stopped at iteration $k < D$ using an appropriate stopping criterion, a sparse parameter vector $w, \|w\|_0 = k$ is obtained. We call this algorithm Sparse Conjugate Directions Pursuit (SCDP).

Fig. 5.3 displays the convergence plots for gradient descent, conjugate gradient and sparse conjugate gradient executed on a 2-dimensional quadratic objective function associated with a linear system $Aw = b$. It is seen that the first direction of SCDP is sparse (only parameter $(w)_1$ is non-zero) and that similar to CG only 2 iterations are needed to converge to the global optimum. Gradient descent needs more iterations to converge as is seen by the dotted line in Fig. 5.3.

Without claiming any general statement let us now give an example indicating that SCDP, given a specific problem, can outperform LASSO. Therefore, a simple linear system $(X^TX + \varepsilon I)w = X^Ty$ with $\varepsilon = 10^{-9}$, where values from $X \in \mathbb{R}^{20\times 15}$ and $y \in \mathbb{R}^{20}$ where randomly sampled from a normal distribution with zero mean and standard deviation one, is considered. Fig. 5.4 displays the least squares objective function values as a function of the cardinality for different methods. Results are averaged over 100 runs. A first curve, called Pareto front, is created by performing an enumerative exhaustive search (which rapidly becomes impractical when D increases) for finding the most optimal subset of components given a cardinality. Hence, solving the problem defined in (5.4) exactly. This curve sets an lower bound on the Mean Squared Error (MSE), $1/N||Xw - y||_2^2$. A second curve presents the MSE in function of the cardinality (iteration number) for SCDP. SCDP computes its solutions in each iteration based on a conjugate basis and a given sparsity pattern. A third and fourth curve present results obtained using the LASSO and a de-biased LASSO. SCDP does not introduce an extra bias as is the case for LASSO which adds an additional L_1 penalization term to the optimization objective. For the LASSO it is therefore advised to use a second de-biasing step (see e.g. [59]) based on a least square estimate with the obtained sparsity pattern. Note that for the (de-biased) LASSO, different values of λ might give the same sparsity pattern. In Fig. 5.4 only that sparsity pattern providing the least MSE value is plotted.

Algorithm 9 gives the pseudo-code of the basic version of SCDP (a more detailed version is described later) . The outline of the algorithm resembles closely this of CG in Algorithm 8, while important differences include the construction of the conjugate basis as explained in previous section. Unfortunately, it is in general not possible to construct a new conjugate vector based on the previous solution only. In [132] it is seen that this update rule relies on the fact that $p^{(1)} = -c^{(1)}$ and that it is not applicable for other choices of $p^{(1)}$. As a consequence, the sparse conjugated directions algorithm has to store all previous directions.

We proceed by explaining how to exploit the sparse structure of the conjugate basis in order to efficiently compute $p^{(k+1)}$. Let $\pi : \{1, \ldots, D\} \to \{1, \ldots, D\}$ be a permutation which denotes the order in which the components of A are selected in the algorithm. The next section will discuss how to select the components,

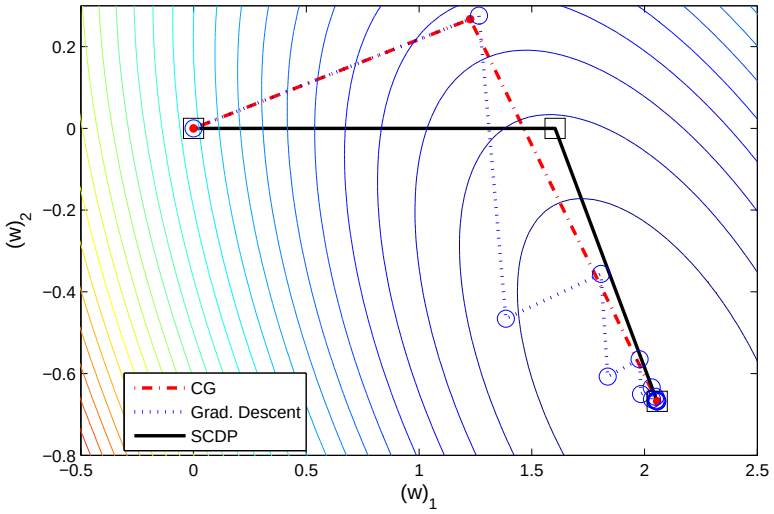


Figure 5.3: Given the quadratic objective function associated with a linear system $Aw = b$ of size 2×2, the solutions in different iterations of respectively gradient descent (given an appropriate step size), conjugate gradient and sparse conjugate gradient are plotted. It is seen that the first direction of SCDP is sparse, only parameter $(w)_1$ is nonzero, and that similar to CG only 2 iterations are needed to converge to the global optimum. Gradient descent needs 9 iterations to converge.

but for now we assume a given π. Define a matrix $P \in \mathbb{R}^{D \times D}$ which stores all search directions, or $P = \left(p^{(1)}, \ldots, p^{(D)}\right)^T$. The conjugate vectors are chosen such that the cardinality of each conjugate vector equals the iteration number.When we select k columns of P according to permutation π via selection matrix $\Sigma^{(k)} \in \mathbb{R}^{D \times k}$; $\Sigma^{(k)} = (e_{\pi(1)} \ \ldots \ e_{\pi(k)})$, the matrix structure is defined as follows

$$\Sigma^{(k)T} P \Sigma^{(k+1)} = \begin{pmatrix} (p^{(1)})_{\pi(1)} & 0 & & & 0 \\ (p^{(2)})_{\pi(1)} & (p^{(2)})_{\pi(2)} & 0 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ (p^{(k)})_{\pi(1)} & \cdots & & (p^{(k)})_{\pi(k)} & 0 \end{pmatrix}. \tag{5.14}$$

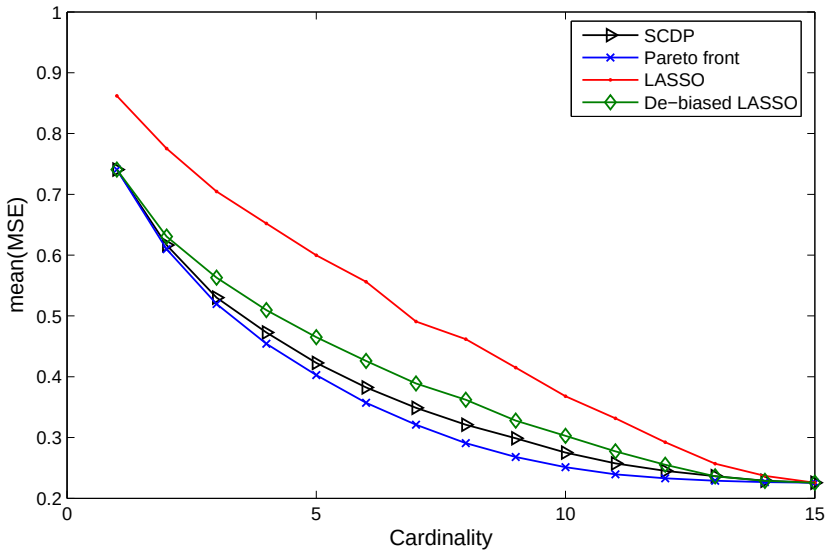


Figure 5.4: A comparison between SCDP, the LASSO, de-biased LASSO and a Pareto front (the exact solution of the combinatorial problem (5.4)) is made based on a small randomly generated linear system of size 20×15. Results are averaged over 100 runs. The MSE of the linear system is plotted as a function of the number of nonzero components of w. Note that in case of the (de-biased) LASSO, multiple MSE values are available per model size. For visibility only the minimum values are plotted. In this example SCDP resembles more closely the Pareto front compared to its alternatives.

Because we want to obtain a conjugate basis of search directions, the following equations has to hold in each iteration

$$\left(\Sigma^{(k)T} P \Sigma^{(k+1)}\right) A^{(k+1)} \left(\Sigma^{(k+1)T} p^{(k+1)}\right) = 0,$$

$$\text{such that} \quad \begin{cases} \Sigma^{(k+1)T} A p^{(k+1)} \in \text{span}\{a_1^{(k+1)}, \ldots, a_{k+1}^{(k+1)}\}, \forall k, \\ \|p^{(k+1)}\|_0 \leq k+1 \end{cases} \tag{5.15}$$

where $a_i^{(k+1)}$ is the i-th row vector of $A^{(k+1)} = \Sigma^{(k+1)T} A \Sigma^{(k+1)}$. As such, given a permutation π, one can compute the sequence of search directions by solving iteratively for $p^{(k+1)}$. Let us now define $B^{(k+1)} = \Sigma^{(k)T} P \Sigma^{(k+1)} A^{(k+1)}$, then

Algorithm 9 Sparse Conjugate Directions Pursuit for solving $Aw = b$

Define: $A = A^T \succ 0$, $A \in \mathbb{R}^{D\times D}$, $A^{(k)} = \Sigma^{(k)T} A \Sigma^{(k)}$, $A^{(k)} \in \mathbb{R}^{k\times k}$ and $a_i^{(k)}$ is the i-th row vector of $A^{(k)}$.
Final Estimate: $w^{(k)}$
Initialize: $w^{(1)} = 0$, a permutation $\pi : \{1, \ldots, D\} \to \{1, \ldots, D\}$,
$k := 1$, $p^{(1)} = e_{\pi(1)}, c^{(1)} = -b$, $\Sigma^{(1)} = e_{\pi(1)}$; $p^{(1)}, c^{(1)}, e_{\pi(1)} \in \mathbb{R}^D$.
repeat
$\eta^{(k)} = -\frac{c^{(k)T} p^{(k)}}{p^{(k)T} A p^{(k)}}$
$w^{(k+1)} = w^{(k)} + \eta^{(k)} p^{(k)}$
$c^{(k+1)} = c^{(k)} + \eta A p^{(k)}$
$\Sigma^{(k+1)} = [\Sigma^{(k)} \quad e_{\pi(k)}]$
find $p^{(k+1)}$ such that $\begin{cases} \|p^{(k+1)}\|_0 = k+1 \\ p^{(i)T} \Sigma^{(k+1)} A^{(k+1)} \Sigma^{(k+1)T} p^{(k+1)} = 0, \forall i \in \{1, 2, ..., k\} \\ \Sigma^{(k+1)T} A p^{(k+1)} \in \text{span}\{a_1^{(k+1)}, \ldots, a_{k+1}^{(k+1)}\} \end{cases}$
$k := k + 1$
until Stopping criterion is met

Proposition 8. *The matrix $B^{(k+1)}$ has a upper block diagonal matrix structure defined as follows*

$$B^{(k+1)} = \begin{pmatrix} B_{11}^{(k+1)} & B_{12}^{(k+1)} & \cdots & B_{1k}^{(k+1)} & B_{1(k+1)}^{(k+1)} \\ 0 & B_{22}^{(k+1)} & & & \vdots \\ \vdots & \ddots & \ddots & & \\ 0 & \cdots & 0 & B_{kk}^{(k+1)} & B_{k(k+1)}^{(k+1)} \end{pmatrix}. \tag{5.16}$$

Proof. Given a symmetric positive definite matrix A, it can be seen from (5.15) that the following equations hold

$$\Sigma^{(k+1)T} p^{(i)T} a_j^{(k+1)} = 0 \text{ for } i = 2, ..., k+1 \text{ and } \forall j = 1, ..., i-1.$$

The first direction $p^{(1)}$ can be chosen arbitrarily (with the constraint that $\|p^{(1)}\|_0 = 1$) at the start of SCDP and is therefore not considered in this formula. This will result in a upper block diagonal structure for matrix $B^{(k+1)}$. □

Exploiting the fact that by definition $\|p^{(k+1)}\| = k+1$ we can define a linear system (with sparse structure) as follows

$$B^{(k+1)} \begin{pmatrix} (p^{(k+1)})_{\pi(1)} \\ \vdots \\ (p^{(k+1)})_{\pi(k+1)} \end{pmatrix} = 0. \tag{5.17}$$

Since this is an under-determined system with $k+1$ parameters and k equations we choose the parameter value $(p^{(k+1)})_{\pi(k+1)}$ to be 1 in each iteration without having loss of generality. If we define $U = B_{ij}^{(k+1)}$ for $i, j = 1, ..., k$ this will result in the following linear system

$$U \begin{pmatrix} (p^{(k+1)})_{\pi(1)} \\ \vdots \\ (p^{(k+1)})_{\pi(k)} \end{pmatrix} = \begin{pmatrix} B_{1(k+1)}^{(k+1)} \\ \vdots \\ B_{k(k+1)}^{(k+1)} \end{pmatrix}. \tag{5.18}$$

Because U is an upper triangular matrix this linear system can be easily solved by backward substitution [133].

On the choice of the next component

A very important step in the algorithm is the determination of the next search direction in which the matrix $\Sigma^{(k)}$ selects the components of A. In relaxation techniques such as Gauss-Seidel (GS) [194] one iteratively sets one residual in each iteration to zero and hopes that this results in turning the new tentative solution closer to the correct solution. In order to ameliorate the convergence speed one can use an ordering scheme in which the individual residuals of $c^{(k)} = Aw^{(k)} - b$ (which can be correlations if the linear system originated from a least squares setup) are set to zero according to Southwell iteration [194] which means that the components of the largest residuals are selected first.

In SCDP we use the same idea. In each iteration we add a component of A which has the largest absolute residual

$$s^{(k+1)} = \arg\max_{i \notin \mathcal{A}} |(c^{(k)})_i|, \tag{5.19}$$

where $\mathcal{A}$ is the set of previously selected indexes. We expect that this results in a rapidly decreasing objective function.

5.3.3 Computational Complexity

In Algorithm 10 a detailed version of SCDP is shown. As with CG the most expensive operation is the matrix vector product $Ap^{(k)}$. Because in SCDP we have that $\|p^{(k)}\|_2^2 = k$, this product has complexity term $\mathcal{O}(kD)$. Different to CG there is no simple update rule for determining a new conjugate vector. SCDP therefore needs to determine a new conjugate direction based on all previous ones. The most expensive operation regarding the conjugate basis is the computation of matrix $B^{(k)}$. Fortunately, it is not needed to recompute it in each iteration. At iteration k we can retain the previous values of $B^{(k-1)}$ and update the elements $B^{(k)}_{k(k-1)}$ and $B^{(k)}_{ik}$ for $i = 1, ..., k$. Which means that $B^{(k)}$ can be updated iteratively at a cost of $\mathcal{O}(k^2)$. This gives an overall computational complexity of $\mathcal{O}(kD + k^2)$ per iteration k compared to $\mathcal{O}(D^2)$ for standard CG. For large linear systems, SCDP will be much faster than CG in case only a small number of iterations are needed. In terms of memory usage of the algorithm (without incorporating the storage of A) we have that CG just needs to store the single previous direction, residuals and parameter values which gives $\mathcal{O}(3D)$. SCDP needs to store the residuals $c^{(k)}$, the non-zero elements of $p^{(k)}$ and $w^{(k)}$ and matrix $B^{(k)}$ which then gives $\mathcal{O}(D + 2k + k^2)$. In iteration k of SCDP the cardinality of the parameter vector $w^{(k)}$ is k compared to (most likely) D in case of CG. The results are summarized in Table 5.1.

	SCDP	CG
Memory	$\mathcal{O}(D + 2k + k^2)$	$\mathcal{O}(3D)$
Computation	$\mathcal{O}(kD + k^2)$	$\mathcal{O}(D^2)$
Cardinality	k	D

Table 5.1: Comparisons of SCDP and CG in terms memory requirement, training cost in iteration k and the cardinality of the parameter vector w (number of non-zero elements).

5.3.4 Computational issues

Stopping criterion

Defining a stopping criterion is to some extent dependent on the application. In the signal processing related literature algorithms are typically stopped when $\|r^{(k)}\|_2^2$ drops below some predefined threshold. In machine learning, model selection is

Algorithm 10 Sparse Conjugate Directions Pursuit

Define: $A = A^T \succ 0$, $A^{(k)} = \Sigma^{(k)T} A \Sigma^{(k)}, A^{(k)} \in \mathbb{R}^{k \times k}$
Final Estimate: $w^{(k)}$
Initialize: $w^{(1)} = 0_D$, $c^{(1)} = -b$,
$s^{(1)} = \arg\max_i(|(c^{(1)})_i|)$, $\mathcal{A} = \{s^(1)\}$
$p^{(1)} = e_{s^{(1)}}$,$\Sigma^{(1)} = e_{s^{(1)}}$, $P = p^{(k)}$, $B^{(1)} = \emptyset$, $k := 1$
repeat
$t = Ap^{(k)}$
$\eta^{(k)} = -\frac{c^{(k)T} p^{(k)}}{p^{(k)T} t}$
$w^{(k+1)} = w^{(k)} + \eta^{(k)} p^{(k)}$
$c^{(k+1)} = c^{(k)} + \eta^{(k)} t$
$s^{(k+1)} = \arg\max_{i \notin \mathcal{A}}(|(c^{(k)})_i|)$, $\mathcal{A} := \mathcal{A} \bigcup \{s^{(k+1)}\}$
$\Sigma^{(k+1)} = \begin{bmatrix} \Sigma^{(k)} & e_{\pi(s^{(k+1)})} \end{bmatrix}$
$B^{(k+1)} = \Sigma^{(k)T} P \Sigma^{(k+1)} A^{(k+1)}$
$(p^{(k+1)})_{s^{(k)}} = \frac{B^{(k+1)}_{k(k+1)}}{B^{(k+1)}_{kk}}$
$(p^{(k+1)})_{s^{(i)}} = \frac{(B^{(k+1)}_{i(k+1)} - \sum_{j=i+1}^{k} B^{(k+1)}_{ij} p^{(k+1)}_{s^{(j)}})}{B^{(k+1)}_{ii}}$, $i = k-1, k-2, ..., 1$
$P = [P^T \quad p^{(k+1)T}]^T$
$k := k + 1$
until Stopping criterion is met

usually performed using the error estimated on independent validation sets. Since the application in the next section is found in the latter context where the aim is to have a low misclassification rate on the training set and high generalization performance on unseen data, and since the number of iterations of the algorithm is defined to be equal to the number of model parameters, we shall rather choose the CV error to decide when to stop. In case the algorithm can be stopped early at iteration $k < D$ we have a model with a cardinality of k. In fact the cross-validation process as used to obtain the hyper-parameters is employed to determine the sparsity of the model as will be explained in Section 5.4.1.

Probabilistic speed-up

In order to decrease the computational complexity, algorithms similar to SCDP often only use a subsample of components to choose from. In case of large scale systems the main computational bottleneck of SCDP is to update $c^{(k)}$ (assuming that the algorithm can be stopped long before reaching D). In case $c^{(k)}$ is not

used as a stopping criterion it is only used to add a new component to the active set. Instead of using the full search space (for determining the novel nonsparse component in an iteration) of dimension $D-k$ only a random subset of size υ might be employed. This reduces the computation time. The resulting algorithm is called SCDP Probabilistic (SCDPP). In SCDPP there is no need to compute $Ap^{(k)}$ entirely, but instead only $A^{(k)}\Sigma^{(k)T}p^{(k)}$ has to be computed and $c^{*(k)} = \Sigma^{*T}A\Sigma^{(k)}\Sigma^{(k)T}w^{(k)} - \Sigma^{*T}b$ (see [6]) where $\Sigma^* \in \mathbb{R}^{D\times\upsilon}$ is a randomly drawn selection matrix for the purpose of selecting υ components of A.

In order to obtain a residual with a probability of 0.95 among the 5% largest residuals, we only need a random subset of size $\upsilon = 59$ (e.g. [147]). This means that in each iteration we only compute 59 residuals and select from within that set the largest residual and corresponding component of A. In case the A is available in memory, the training complexity can now approximately be written as $\mathcal{O}(\upsilon k + k^2)$. The total memory used by the algorithm is now $\mathcal{O}(\upsilon + 2k + k^2)$.

However, using this speed-up affects reproducibility of the results since each run of the algorithm might give a different solution. As will be shown in the experimental section in some cases the results may deviate strongly from those of standard SCDP, resulting in less accurate predictions.

5.4 Application: using SCDP on FS-LSSVM formalism

In order to model the data using FS-LSSVM (see Section 3.3.3) with sufficient accuracy while having reasonable training and evaluation times, it is important to determine the amount of Prototype Vectors (PV) used in the approximation. In this section the goal is to approximately solve the FS-LSSVM problem using SCDP. Here we efficiently compute k_{m} (the maximum number of SCDP iterations) FS-LSSVM models with increasing complexity. Using a cross-validation criterion (see Section 5.4.1) the "optimal" (according to the SCDP heuristic) model complexity is then chosen.

[6] Indeed, in case the probabilistic speed up is used the update rule $c^{(k+1)} = c^{(k)} + \eta^{(k)}t$ (see Algorithm 10) cannot be used anymore since not all residuals are computed in each SCDP iteration.

The FS-LSSVM method is a least squares estimating problem applied to an over-determined system of linear equations,

$$\left(\begin{array}{c|c} \hat{\Phi} & 1_N \end{array} \right) \left(\frac{w}{b} \right) \approx y, \tag{5.20}$$

where $\hat{\Phi} \in \mathbb{R}^{N \times M}$ is defined according to (3.34), $w \in \mathbb{R}^M$ are the model parameters and $y \in \mathbb{R}^N$ the class labels in case of classification [7]. In this type of problems we have that $M \ll N$, which matches the considered SCDP setting explained earlier. As is seen in Section 3.3.3 the FS-LSSVM training boils down to solving the following linear system,

$$\left(\begin{array}{c|c} \hat{\Phi}^T\hat{\Phi} + \frac{1}{\gamma} I & \hat{\Phi}^T 1_N \\ \hline 1_N^T \hat{\Phi} & 1_N^T 1_N \end{array} \right) \left(\frac{w}{b} \right) = \left(\frac{\hat{\Phi}^T y}{1_N^T y} \right). \tag{5.21}$$

Starting from an initial set of PVs, which are possibly expressed in terms of all training data, the linear system in (5.21) can be computed. By applying SCDP on (5.21) and assuming that the algorithm has been executed for k_{m} iterations, different model complexities ranging from 1 to k_{m} are available. The best model size according to the employed SCDP heuristic is easily selected using CV scores.

Unfortunately, it is not straightforward to apply SCDP to the standard FS-LSSVM formulation since the eigenvectors and values, needed to compute the features $\hat{\Phi}$, cannot be updated in case an additional dimension (or an additional PV in this context) is added. We therefore rearrange the FS-LSSVM system such that the parameters w are solved up to a linear transformation resulting in the same predictions as the original formulation without the use of an EVD as was used in Section 3.3.3. The transformed linear system can then be solved using the previously explained SCDP algorithm. This transformation is explained in Lemma 9.

Lemma 9. *The linear system (5.21) is equivalent to*

$$\left(\begin{array}{c|c} \Omega'^T\Omega' + \frac{1}{\gamma}\Omega & \Omega'^T 1_N \\ \hline 1_N^T \Omega' & 1_N^T 1_N \end{array} \right) \left(\frac{\tilde{w}}{b} \right) = \left(\frac{\Omega'^T y}{1_N^T y} \right), \tag{5.22}$$

where $\Omega_{ij} = K(x_i, x_j)$ with $x_i, x_j \in \mathcal{X}_{\mathrm{PV}}$ and $\mathcal{X}_{\mathrm{PV}}$ the set of selected PVs, $\Omega \in \mathbb{R}^{M \times M}$; $\Omega' \in \mathbb{R}^{N \times M}$ with $\Omega'_{ij} = K(x_i, x_j)$, $x_i \in \mathcal{X}_{\mathrm{tr}}$, and $x_j \in \mathcal{X}_{\mathrm{PV}}$ with $\mathcal{X}_{\mathrm{tr}}$ the set of training vectors; $\Omega U = U\Lambda$ according to (3.27); $\tilde{w} = USw$ with $S = \Lambda^{-\frac{1}{2}}$. Λ is assumed to be a diagonal matrix containing strictly positive eigenvalues.

[7] The parameter b denotes the intercept (bias) term in the context of FS-LSSVM, which differs from the meaning of symbol b in a linear system used in the previous section.

Proof. Recall that according to Section 3.3.3 the computation of the approximate feature map $\hat{\varphi}$ is based on the eigenvectors and values of a reduced kernel matrix $\Omega U = U\Lambda$, $\Omega \in \mathbb{R}^{M\times M}$ resulting in

$$\hat{\Phi} = \Omega' U S, \tag{5.23}$$

where $\Omega' \in \mathbb{R}^{N\times M}, \Omega'_{ij} = K(x_i, x_j), x_i \in \mathcal{X}_{\text{tr}}, x_j \in \mathcal{X}_{\text{PV}}$ with $\mathcal{X}_{\text{PV}}$ the set of selected PVs and $\mathcal{X}_{\text{tr}}$ the set of training vectors and $S = \Lambda^{-\frac{1}{2}}$ (where we assume that the eigenvalues in Λ are strictly positive). Combining (5.23) and (5.21) gives

$$\begin{aligned}
&\left(\begin{array}{c|c} SU^T\Omega'^T\Omega' US + \frac{1}{\gamma} I & SU^T\Omega'^T 1_N \\ \hline 1_N^T \Omega' US & 1_N^T 1_N \end{array}\right)\left(\begin{array}{c} w \\ \hline b \end{array}\right) = \left(\begin{array}{c} SU^T\Omega' y \\ \hline 1_N^T y \end{array}\right) \\
\Rightarrow\quad &\left(\begin{array}{c|c} \Omega'^T\Omega' US + \frac{1}{\gamma}(SU^T)^{-1} & \Omega'^T 1_N \\ \hline 1_N^T \Omega' US & 1_N^T 1_N \end{array}\right)\left(\begin{array}{c} w \\ \hline b \end{array}\right) = \left(\begin{array}{c} \Omega' y \\ \hline 1_N^T y \end{array}\right) \\
\Rightarrow\quad &\left(\begin{array}{c|c} \Omega'^T\Omega' + \frac{1}{\gamma} US^{-2}U^T & \Omega'^T 1_N \\ \hline 1_N^T \Omega' & 1_N^T 1_N \end{array}\right)\left(\begin{array}{c} \tilde{w} \\ \hline b \end{array}\right) = \left(\begin{array}{c} \Omega' y \\ \hline 1_N^T y \end{array}\right),
\end{aligned} \tag{5.24}$$

where

$$\tilde{w} = USw. \tag{5.25}$$

Since $\Lambda = S^{-2}$, $\Omega U = U\Lambda$ and $U^T U = I$ we can write $US^{-2}U^T = \Omega$. If we substitute this result in the latter linear system we get (5.22). □

In case of binary prediction we have that $\hat{y}^* = \text{sign}\left(w^T\hat{\varphi}(x^*) + b\right)$ which can be written as $\hat{y}^* = \text{sign}\left(\Omega^* USw + b\right)$ according to (5.23) where $\Omega^* \in \mathbb{R}^{1\times M}, \Omega^*_{1i} = K(x^*, x_i)$. Combining the latter and (5.25) we have $\hat{y}^* = \text{sign}\left(\Omega^*\tilde{w} + b\right)$. This gives

$$\begin{aligned}
\hat{y}^* &= \text{sign}(w^T\hat{\varphi}(x^*) + b) \\
&= \text{sign}\left(\sum_{i=1}^{M} \tilde{w}_i K(x^*, x_i) + b\right).
\end{aligned} \tag{5.26}$$

In order to ensure positive definiteness of the matrix in (5.22) we add a L_2 penalization term on the intercept term b controlled by a small predefined positive regularization constant ν (in our experiments set to 10^{-8}). Lemma 5.27 indicates the positive definite characteristic of the matrix in the considered linear system.

Lemma 10. *For $\nu > 0$ and $\gamma > 0$, the matrix*

$$\left(\begin{array}{c|c} \Omega'^T\Omega' + \frac{1}{\gamma}\Omega & \Omega'^T 1_N \\ \hline 1_N^T\Omega' & 1_N^T 1_N + \nu \end{array}\right), \tag{5.27}$$

is positive definite.

Proof. An $N \times N$ real symmetric matrix A is positive semi-definite if $p^T Ap \geq 0$ for all non-zero vectors $p \in \mathbb{R}^M$ with real entries. By applying such multiplication to (5.27) we can write [8]

$$h = \left(\begin{array}{c|c} d^T & c \end{array}\right) \left(\begin{array}{c|c} \Omega'^T\Omega' + \frac{1}{\gamma}\Omega & \Omega'^T 1_N \\ \hline 1_N^T\Omega' & 1_N^T 1_N + \nu \end{array}\right) \left(\begin{array}{c} d \\ \hline c \end{array}\right),$$

which can be written as

$$\begin{aligned} h &= (1c)^T 1c + \nu c^2 + 2\left((1c)^T \Omega' d\right) + (\Omega' d)^T \Omega' d + \frac{1}{\gamma} d^T \Omega d \\ &= (1c + \Omega' c)^2 + \frac{1}{\gamma} d^T \Omega d + \nu c^2. \end{aligned}$$

Since the kernel matrix Ω by definition is positive semi-definite the product $d^T\Omega d$ is greater or equal than zero and hence because $\gamma > 0$, $\nu > 0$, $h \geq 0$. Thus, the matrix (5.27) is positive definite. □

Remark 19. In case the probabilistic speed-up is used (see Section 5.3.4), only a small fraction of the kernel matrix Ω' need to be available. However, it is good practice to keep a cache of kernel evaluations to exclude or reduce the number of kernel recalculations. Although intelligent caching methods could be defined, we simply choose an initial set of PVs, from which the SCDP algorithm can choose from, such that the corresponding entire Ω' matrix fits in memory.

Remark 20. Note that our setup allows to select PVs other than those in the training set alone using e.g. k-center clustering (see Section 4.3.1).

5.4.1 Determining the final model size

Let us first remark that in case the initial set of PVs equals the full training set, our method will approach the LS-SVM solution when the final model size is large. The

[8] For clarity we note that $\Omega' \in \mathbb{R}^{N\times M}$, $\Omega \in \mathbb{R}^{M\times M}$, $1 \in \mathbb{R}^N$, $c \in \mathbb{R}$, and $d \in \mathbb{R}^M$.

same is true for the standard FS-LSSVM formulation. However, we are interested in having as small a cardinality of the final model as possible.

Since in iteration k of the SCDP algorithm the number of non-zero model parameters is k, it is seen that the stopping criterion is directly related to the final model size. As mentioned earlier in machine learning the aim is to have a model with a low misclassification rate on the training set and good generalization performance on unseen data. We therefore advocate the use of a validation error for deciding when to stop. For this purpose we can use the cross-validation performance also used to determine the hyper-parameters (regularization and kernel parameters). To save computer resources we keep track of the validation score as a function of k (cardinality) and stop in case this function does not change much anymore.

We proceed as follows. First an appropriate value for k_{m} is chosen (as high as possible according to the available computer resources). For a given choice of hyper-parameters, the SCDP-FSLSSVM method is then applied on each separate CV fold using the following stopping criterion: run the algorithm until the number of iterations reaches k_{m} or the validation score remains nearly the same. Combining the results for each fold gives an estimate of the v-fold CV error for each model size ranging from 1 to at most k_m. After combination of the results of the different folds, we select the final model size such that the CV criterion is lowest. This procedure can be repeated for different hyper-parameter values, including γ, σ, etc.

The v-fold CV error $f_{\mathrm{cv}}^{(k)}$ at iteration k is defined as

$$f_{\mathrm{cv}}^{(k)} = \frac{1}{N_v} \sum_{v=1}^{N_v} \mathrm{MSE}(y(\mathcal{S}_v), \hat{y}_v(\mathcal{S}_v)), \tag{5.28}$$

where the indices of the N_v CV subsample sets are denoted as $\mathcal{S}_v, v = 1, ..., N_v$, $\mathcal{S}_1 \bigcap \ldots \bigcap \mathcal{S}_{N_v} = \emptyset$, $\mathrm{MSE}(y, z)$ as defined in (3.54), and $\hat{y}_v = (\hat{f}_v(x_1), \ldots, \hat{f}_v(x_N))^T$ with $\hat{f}_v$ the model which was trained in the v-th CV iteration.

Remark 21. Note that in case the probabilistic speed-up is used, the random selection of candidate directions is fixed in order to exclude undesirable influence of the random effects in this tuning process. Thus, at iteration k, the different SCDP runs (using different values for the hyper-parameters) will choose the same random sets from which a new PV can be selected.

Remark 22. We found empirically that an adequate stopping rule would be

$$\frac{|\frac{1}{\Delta_k} \sum_{i=1}^{\Delta_k} f_{\mathrm{cv}}^{(k-i)} - f_{\mathrm{cv}}^{(k)}|}{|f_{\mathrm{cv}}^{(k)}|} < \epsilon, \tag{5.29}$$

where $f_{\mathrm{cv}}^{(k)}$ defined as in (5.28), and Δ_k is a fixed integer which determines the window size from which an average validation performance is calculated.

5.4.2 Algorithmic parameters

As illustrated in the previous section, four parameters: M, v, ϵ and k_m control the processor and memory use of the algorithm and can have impact on the prediction accuracy of the final model. We therefore briefly summarize the effect of the different parameters in Table 5.2.

Parameter	**Influence**
M	Controls the initial size of the matrix in (5.22). We can set $M = N$ and select all training data as initial PVs or can according to some criterion determine M PVs which represent the training set. The proposed framework allows the incorporation of PVs which do not coincidence with the training data. In case of combining several binary classifiers in the multi-class setting this parameter defines an upperbound on the number of unique PVs for all binary models.
v	The size of the random subset out of which the PVs are selected in the probabilistic speed-up case (SCDPP). The smaller v the less $c^{(k)}$ need to be computed in each iteration. However, this means that different solution vectors with certain cardinality are possible for a given linear system while for SCDP always the same solution is found. The larger v is, the more the solution will be similar to that of SCDP.
ϵ	Stoppingcriterion parameter as defined in (5.29). At a certain point the addition of extra PVs to the model has almost no effect on the validation performance. In this case the algorithm can be stopped early which saves computer resources. In case enough resources are available, probably the safest option is to set this parameter to $-\infty$. Afterwards the cross-validation score will be used to select the best cardinality.
k_m	Sets a maximum on the model size and training time.

Table 5.2: This table surveys the influence of the parameters M, v, ϵ and k_m on the SCDP(P) algorithm.

5.4.3 Fast v-fold cross-validation

We found the following fast implementation of CV in [18]. Although other alternatives exist (see references in [18]) this work opts to use the simple approach given in [18] to calculate a fast v-fold cross-validation (CV) score as is described in Algorithm 11. Consider the shorthand notation $A_{\mathrm{f}}\tilde{w} = b_{\mathrm{f}}$ for the full linear system defined in (5.22). Consider a certain iteration in CV and let the matrix A_{tr} be the matrix based on a subset of the training samples (all folds except one). Instead of computing $A_{\mathrm{tr}}^{(v)}$ (and corresponding $b_{\mathrm{tr}}^{(v)}$) from scratch in each v-th CV iteration the following was proposed. At first the following results are computed: $\Omega' \in \mathbb{R}^{N\times M}$,

$\Omega \in \mathbb{R}^{M \times M}$ with M the number of PVs (which do not have to coincidence with the training set), A_f, and b_f. Then based on the fact that we can use the same PVs for each of the folds we can compute $A_\mathrm{tr}^{(v)}$ and $b_\mathrm{tr}^{(v)}$ for CV iteration v as follows

$$A_\mathrm{tr}^{(v)} = A_\mathrm{f} - \left(\begin{array}{c|c} \Omega'^{(v)T}\Omega'^{(v)} & \Omega'^{(v)T}1_{N^{(v)}} \\ \hline 1_{N^{(v)}}^T\Omega'^{(v)} & N^{(v)} \end{array} \right), \tag{5.30}$$

with $N^{(v)}$ the number of examples of the v-th fold and $\Omega'^{(v)}$ is the selection of rows of Ω' corresponding with the samples in the v-th fold. A similar approach is used for $b_\mathrm{tr}^{(v)}$

$$b_\mathrm{tr}^{(v)} = b_\mathrm{f} - \left(\frac{\Omega'^{(v)}y^{(v)}}{1_{N^{(v)}}^T y^{(v)}} \right), \tag{5.31}$$

where $y^{(v)}$ are the labels corresponding to the v-th fold.

Remark 23. The matrices Ω' and Ω for different kernel parameters can be computed more efficiently depending on which kernel function is used. For example consider the RBF kernel which is defined as $K(x, x') = \exp(-||x - x'||_2^2/\sigma^2)$. Instead of recomputing each kernel function the nominator in the power of the exponent $n = -||x - x'||_2^2$ is only computed once and is reused during the CV loop.

Algorithm 11 Fast v-fold cross-validation for SCDP-FSLSSVM

1: Calculate the matrices Ω', Ω, $A_\mathrm{f} = \left(\begin{array}{c|c} \Omega'^T\Omega' + \frac{1}{\gamma}\Omega & \Omega'^T 1 \\ \hline 1_N^T\Omega' & 1_N^T 1_N + \nu \end{array} \right)$,
$b_\mathrm{f} = \left(\frac{\Omega' y}{1_N^T y} \right)$.
2: Randomly split the data into v disjoint sets of nearly equal size.
3: **for** $v = 1, \ldots, N_v$ **do**
4: Compute $A_\mathrm{tr}^{(v)}$ and $b_\mathrm{tr}^{(v)}$ using respectively (5.30) and (5.31).
5: Solve linear system $A_\mathrm{tr}^{(v)}\tilde{w}^{(v)} = b_\mathrm{tr}^{(v)}$ using SCDP and compute validation score per model size using an appropriate loss function.
6: **end for**

5.4.4 Relation to other methods

In context of boosting, the Conjugate Direction Boosting (CDBoost) algorithm is proposed in [121]. Using a similar idea of sparse conjugate vectors the authors extend the L_2-boosting method [17] and focus on linear regression (but also mention

a classification example in the experimental section). They also showed that the CDBoost algorithm can be improved in terms of prediction accuracy by only taking a small fraction of the optimal step (shrinkage) along the conjugate direction, similar to the idea in FStR [9]. Since this results in a higher computational cost (and extra tuning parameter) we did not considered this in this work. They also indicated that their framework could be generalized to the use of any base learner. Our work specializes the SCDP idea to the kernel-based FS-LSSVM framework. Compared to [121], we use a ridge penalty (i.e. $\|w\|_2^2$) in the least squares problem and incorporate a separate intercept (b) term into the model.

In the literature there are many kernel-based examples which employ a greedy approach such as in SCDP-FSLSSVM in order to increase the model complexity during the training process. Some examples, which also work in the context of a L_2 loss function, include

- In [25] the primal LS-SVM objective (as in (3.17)) is approximated using a sparse approximate kernel expansion where only a relatively small number $M < N$ of vectors contribute

$$w = \sum_{x_i \in \mathcal{X}_{\text{pv}}} \beta_i \varphi(x_i). \tag{5.32}$$

  Eliminating in (3.18) the error variables e_i by substituting the constraint in the objective and combining with (3.18), the same linear system is obtained as in (5.22). However, our approach is based on the FS-LSSVM method which gives additional insight in how to select the PVs. Consequently, the underlying algorithm is different. Note that in the context of SVMs the same sparse approximate kernel expansion is proposed in [101] resulting in a scheme which iteratively solves (5.22) (an iterative scheme is used in order to handle the inequality constraints in the primal SVM formulation (3.7)) .

- Instead of approximating the primal using a sparse model, in [90] the authors directly approximate the dual LS-SVM formulation (3.19) and obtain a sparse model. Although, such a setup does not require the additional computation of a (numerical unstable) kernel matrix product $\Omega'^T\Omega'$, we conclude from preliminary experiments that it requires more iterations to converge, resulting

[9] Due to the small stepsizes, the corresponding subspaces of the matrix in the linear system are no longer solved in a least squares sense. Thus, it may happen that a component already included in the model is rechosen. In this case no new conjugate directions can be found since there are only k conjugate directions in a k-dimensional subspace. In [121] the authors propose to restart the algorithm. The actual $\hat{w}^{(k-1)}$ is set as the starting $\hat{w}^{(1)}$, all information about previous directions is deleted and the algorithm is started again.

in a less sparse model and it requires comparable or even longer training times than SCDP-FSLSSVM (note that the authors [90] made the same observation when comparing to Kernel Matching Pursuit (KMP) [187]). [90] elaborates on the comparison of this technique, Kernel Matching Pursuit (KMP), and Sparse Greedy Gaussian Process (SGGP) [163].

- The idea of KMP is mainly to select basis functions in a greedy manner for non-regularized kernel least-squares problems. The authors describe 3 versions of KMP of which the back-fitting procedure is very similar to SCDP-FSLSSVM. Besides the absence of a regularization term and an intercept term, the order selection criterion is also different to that of SCDP-FSLSSVM. In case of KMP this is $s = \arg\max_{i \notin \mathcal{A}} |(r)_i|, r = \Omega w - y$ while for SCDP-FSLSSVM this is $s = \arg\max_{i \notin \mathcal{A}} |(c)_i|, c = \Omega^T(\Omega w - y)$.

- In [33] the authors propose a general kernel learning framework based on Orthogonal Least Squares (OLS) regression. Given an orthogonal decomposition of the full kernel matrix $\Omega \in \mathbb{R}^{N \times N}, \Omega = QR$ where Q is the matrix with orthonormal columns, the authors proceed by writing $f(x) = QRw = Qg$ and solving for g instead of w (which is possible since the space spanned by the columns of Ω is identical to those spanned by columns in Q). Using the fact that $Q^TQ = I$ an efficient computation of g and Leave-One-Out (LOO) scores can be obtained. Opposed to back-fitting (used by SCDP), OLS uses a pre-fitting scheme. This has impact on the PV selection procedure which is explained as follows. A new component is selected based on the LOO scores of all combinations of the not yet selected components with the current selected components. However, this pre-fitting requires the projection of all not yet selected columns to the subspace spanned by the selected components which is computationally more demanding than that of SCDP and not suitable for large-scale data sets. Additionally an Automatic Relevance Determination (ARD) framework [122] is used to tune the hyper-parameters.

5.4.5 Comparison of computational complexity

Assuming a "good" fixed set of PVs of size M is available, the cost of computing a classical FS-LSSVM model is dominated by 3 parts: (i) the computation of $\hat{\Phi}$ (which includes an eigenvalue decomposition of the reduced kernel matrix Ω) with an associated cost of $\mathcal{O}(M^3 + M^2N)$, (ii) the computation of $\hat{\Phi}^T\hat{\Phi}$ in (5.21) with a computational cost of $\mathcal{O}(N^2M^2)$ and (iii) solving the linear system using for

instance CG (in case the intercept term is regularized) with a cost $\mathcal{O}(k_1 M^2)$ (k_1 the number of iterations to converge).

The two most computational expensive operations in the SCDP-FSLSSVM algorithm are: (i) the computation of $\Omega'^T\Omega$ in (5.22) with a computational cost of $\mathcal{O}(N^2M^2)$ and (ii) solving the system using SCDP with an approximate cost of $\mathcal{O}(\sum_{k=1}^{M}(kN+k^2))$, or using $\mathcal{O}(\sum_{k=1}^{M}(k\upsilon+k^2))$ operations in case the probabilistic speed up (with $\upsilon = 59$ in our experiments). It is expected that solving the small system using CG is faster than using SCDP which makes FS-LSSVM potentially faster than SCDP-FSLSSVM if considering only a single model size. However, SCDP-FSLSSVM does not require the computation of the feature map $\hat{\Phi}$ explicitly, and SCDP-FSLSSVM gives directly the model parameters for model sizes ranging from 1 to M.

5.5 Experiments

All SCDP experiments in this section are carried out in MATLAB. We empirically test some of the algorithmic parameters for the SCDP-FSLSSVM algorithm discussed in the previous paragraphs and compare to state-of-the-art classification techniques. The hyper-parameter ν which controls the regularization for the intercept term is set to 10^{-8} for all experiments. The other hyper-parameters are tuned using 10-fold cross-validation. For each experiment tuning was performed using the optimization procedure described in Section 3.5. Details about the publicly available data sets which are used in the experiments are given in Appendix B. The SCDP-FSLSSVM MATLAB software can be downloaded from `ftp://ftp.esat.kuleuven.be/pub/SISTA/karsmakers/software/SCDP-FSLSSVM.zip`.

5.5.1 Binary classification problems

Small scale problems

We first assess the performance of SCDP-FSLSSVM on a suite of thirteen public domain benchmark data sets as used in the study by [128]. The 100 random partitions of the data (20 in the case of the `image` and `splice` benchmarks) to form training and test sets as was used in that study, are also used here. Model selection is performed independently for each realization of the data set. The RBF kernel ($K(x,x') = \exp\left(-||x-x'||_2^2/\sigma^2\right)$) was used for all kernel methods. In Table 5.3 some misclassification rates and corresponding model cardinalities are

shown. The column corresponding to SCDP-FSLSSVM reflects results when all training data points are used as initial PVs and no probabilistic speed-up is used. The column with acronym SCDP-FSLSSVM$_{\mathrm{PV}}$ first selects $0.3N$ PVs using the farthest-point clustering algorithm [71] (see Section 4.4.3 for more details) from which the final model PVs are chosen, and no probabilistic speed-up was used. In case of the third column, SCDPP-FSLSSVM training is performed using the probabilistic speed-up with a random subset of size $\upsilon = 59$. No preprocessing was done except standardization of the input samples to zero mean and unit variance. We use the continuous MSE(y, z) model selection criterion as defined in Section 3.5 to select the hyper-parameters (including the model size).

In Table 5.3 it is seen that the misclassification rates of the full LS-SVM method are comparable to those of the SCDP variants while more training time is required and no sparse model is obtained (note that similar results are presented in Table 4.3). Both misclassification rates and model complexities of SCDP-FSLSSVM$_{FS}$ are similar to those of SCDP-FSLSSVM while selecting an initial PV set of size $M = 0.3N$ decreased training time considerably.

In case the probabilistic speed-up is used (SCDPP-FSLSSVM with $\upsilon = 59$) the classification accuracies for data sets `ringnorm` and `titanic` deviate strongly from those of the other alternatives. The corresponding large standard deviations indicate that models with a-typical behaviour are present. The same phenomena can be seen when plotting the Pareto frontiers (see Fig. 5.5, and Fig. 5.6) for both SCDP- and SCDPP-FSLSSVM given a single realization and tuned hyper-parameter set. SCDPP-FSLSSVM was repeated 50 times to display the boxplots. The boxes have horizontal lines at the lower quartile, median, and upper quartile values of the performances corresponding to the 50 experiments[10]. The dotted line is the SCDP-FSLSSVM Pareto curve. It is seen that in most cases SCDPP-FSLSSVM performs slightly worse or similar in terms of misclassification rate compared to SCDP-FSLSSVM for different cardinalities. It is seen that for some data sets using SCDPP-FSLSSVM can give significant differences in terms of misclassification rate compared to the SCDP variant. In case υ is increased the SCDPP-FSLSSVM models converge towards those of SCDP-FSLSSVM but still, although less frequently, models with nontypical performance might occur. We therefore choose not to use this option for the remainder of this section and obtain a computational speed-up by selecting an initial PV set using the k-center clustering method.

[10] Additionally the following markers are used in the boxplot. The whiskers are lines extending from each end of the boxes to show the extent of the rest of the data. Outliers (+) are data with values beyond the ends of the whiskers.

	SCDP-FSLSSVM		**SCDP-FSLSSVM$_{\mathrm{FS}}$**		**SCDPP-FSLSSVM**		**LS-SVM**
	err (%)	# PV	err (%)	# PV	err (%)	# PV	err (%)
banana	**10.46**(±0.44)	32.4(±6)	10.52(±0.42)	34.3(±7.1)	10.58(±0.52)	33.3(±7.7)	10.61(±0.53)
breast-cancer	27.21(±4.23)	6.4(±5.4)	**26.47**(±4.51)	13.5(±9.3)	27.04(±4.38)	11(±8.6)	27.1(±4.72)
diabetis	23.73(±1.94)	8.9(±4.2)	23.39(±1.94)	14.9(±8.2)	23.41(±1.88)	14.1(±7.8)	**23.36**(±1.74)
flare-solar	34.82(±1.88)	13.4(±6.6)	**34.01**(±1.6)	11.6(±8.2)	35(±2.08)	13.8(±6.7)	34.25(±1.87)
german	24.01(±2.24)	26(±9.1)	24.3(±2.21)	13.8(±6.3)	24.2(±2.25)	22.5(±9.4)	**23.55**(±2.22)
heart	**16.2**(±3.27)	4.9(±2.9)	16.22(±3.25)	6.5(±3.7)	16.71(±3.16)	4.8(±3)	16.41(±3.45)
image	**2.86**(±0.55)	317.8(±5.6)	3.21(±0.79)	308.3(±10.3)	3.3(±0.83)	318.2(±12)	2.95(±0.76)
ringnorm	1.61(±0.14)	11.8(±8.2)	**1.52**(±0.12)	6.6(±4.2)	9.99(±11.71)	11.9(±7.8)	1.65(±0.16)
splice	11.36(±0.58)	311.2(±10.6)	12.54(±0.68)	188.1(±29)	11.29(±0.62)	313(±11.6)	**10.57**(±0.66)
thyroid	4.44(±2.05)	18.9(±9.9)	**4.13**(±2.14)	12.8(±7.8)	4.29(±2.06)	16.8(±9.4)	4.61(±2.28)
titanic	22.84(±0.96)	6.3(±3)	22.97(±1.3)	5.4(±2.7)	25.32(±12.34)	7.4(±3.1)	**22.57**(±1.35)
twonorm	2.66(±0.14)	38.2(±9.4)	**2.56**(±0.16)	37.5(±10.1)	2.75(±0.22)	26.9(±12.6)	2.82(±0.2)
waveform	9.86(±0.38)	26.3(±7.7)	9.9(±0.37)	24.8(±7.7)	9.85(±0.39)	27.2(±7.1)	**9.82**(±0.45)

Table 5.3: The table shows the misclassification rates and if applicable the number of finally selected prototype vectors (PV) of 3 versions of SCDP-FSLSSVM and LS-SVM over thirteen benchmark data sets. k-center clustering was used as the PV selection procedure. The results for each method are presented in the in the form of the mean error rate over test data for 100 realizations of each data set (20 in the case of the image and splice data sets), along with the associated standard error. The column denoted by SCDP-FSLSSVM reflects results of the SCDP-FSLSSVM algorithm which uses all training data points as initial PVs and no probabilistic speed-up. The column with acronym SCDP-FSLSSVM$_{\mathrm{PV}}$ first selects $0.3N$ PVs using the farthest-point clustering algorithm [71] (see Section 4.4.3 for more details) from which PVs for the final model are chosen, no probabilistic speed-up was used. In case of the third column SCDPP-FSLSSVM training is performed using the probabilistic speed-up with a random subset of size $\upsilon = 59$.

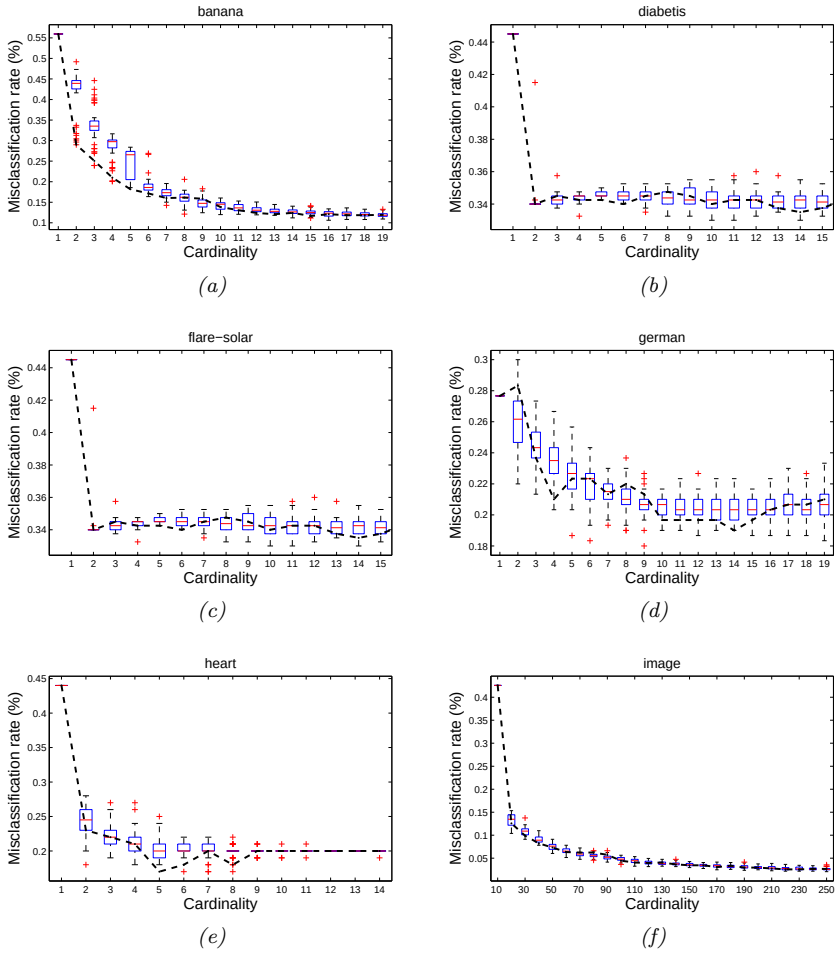


Figure 5.5: Given a single random partition and (tuned) hyper-parameter set, Pareto frontiers are plotted for both SCDP- and SCDPP-FSLSSVM ($\upsilon = 59$) on data sets also used in [128]. SCDPP-FSLSSVM was repeated 50 times in order to make boxplots. The boxes have lines at the lower quartile, median, and upper quartile values of the performances corresponding to the 50 different experiments. The whiskers are lines extending from each end of the boxes to show the extent of performance scores. Outliers (+) are data with values beyond the ends of the whiskers.

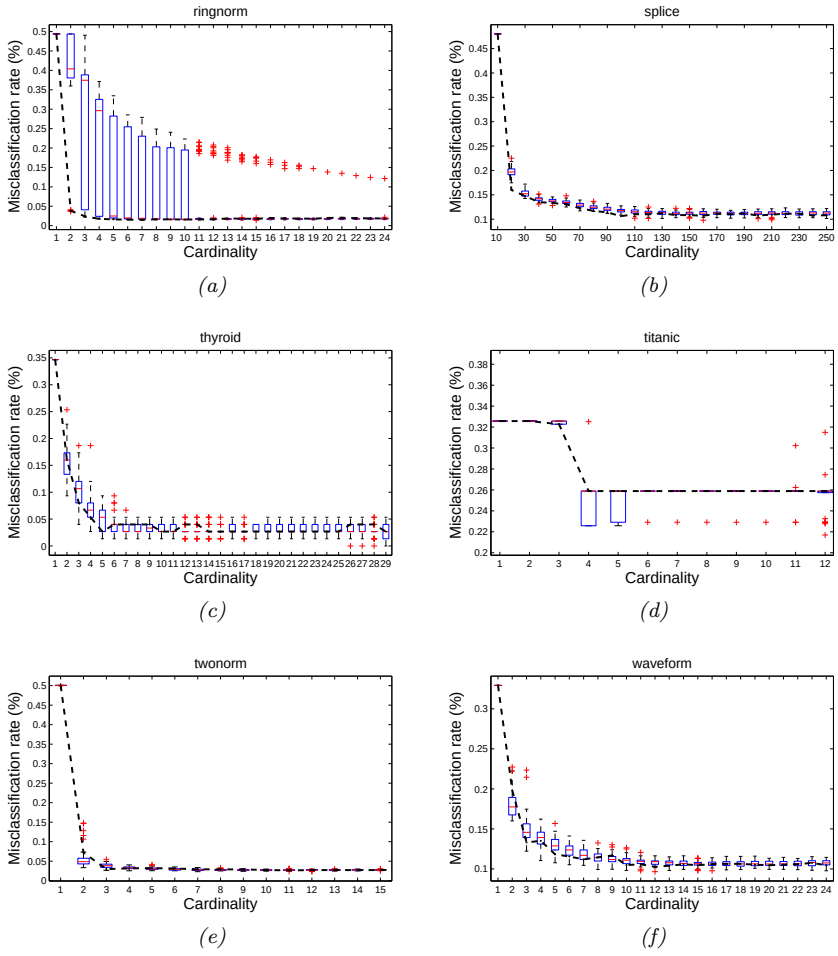


Figure 5.6: Given a single random partition and (tuned) hyper-parameter set, Pareto frontiers are plotted for both SCDP- and SCDPP-FSLSSVM ($\upsilon = 59$) on data sets also used in [128]. SCDPP-FSLSSVM was repeated 50 times in order to make boxplots. The boxes have lines at the lower quartile, median, and upper quartile values of the performances corresponding to the 50 different experiments. The whiskers are lines extending from each end of the boxes to show the extent of the performance scores. Outliers (+) are data with values beyond the ends of the whiskers.

Large scale problems

The algorithm was tested additionally on two larger binary classification tasks. The results are shown in Table 5.4. The misclassification rates and corresponding model

cardinalities are averaged over 5 random partitions of the data sets ($N/3$ data points are used as test set and $2N/3$ examples as training set). No preprocessing was performed except for standardization of the input samples to zero mean and unit variance. Initially, 2000 PVs were selected using k-center clustering. From Table 5.4 it is seen that compared to SVM, SCDP-FSLSSVM shows similar performance while producing much sparser models.

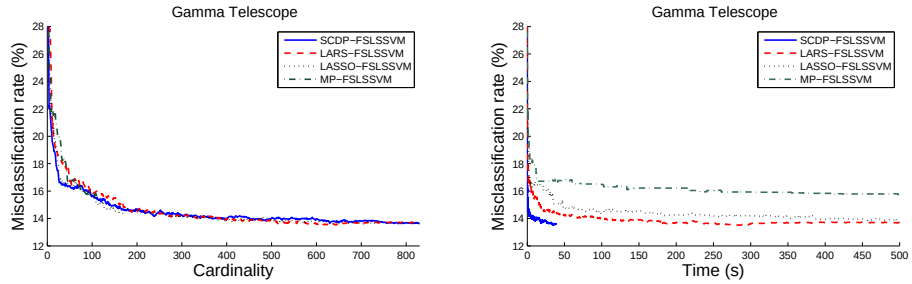
Data set	**SCDP-FSLSSVM**		**SVM**	
	err (%)	# PV	err (%)	# SV
`adult`	14.55($\pm$0.34)	166($\pm$41.6)	14.53($\pm$0.14)	10,494($\pm$212.4.0)
`gamma telescope`	13.90($\pm$0.27)	778($\pm$58.9)	13.3($\pm$0.12)	4,211($\pm$112.6)

Table 5.4: The misclassification rates and corresponding model cardinalities averaged over 5 random partitions of the data sets ($N/3$ data points are used as test set and $2N/3$ examples as training set) for SCDP-FSLSSVM as well as for SVM are shown. No preprocessing was performed except standardization of the input samples to zero mean and unit variance. Initially, 2000 PVs were selected using k-center clustering.

As mentioned in the first part, other methods exist to tackle the optimization problem (5.22) where a sparse model is desired. Examples include MP, LARS and LASSO. We applied these methods for solving (5.22) and compare to SCDP. Using LASSO, LARS and MP introduces a undesirable bias which is removed using an extra de-bias step. Hence, the coefficients of the nonsparse entries are recomputed using ordinary least squares. As such, we used the result of LASSO, LARS and MP only to select the PVs. This must be contrasted to SCDP where such de-bias step is not required.

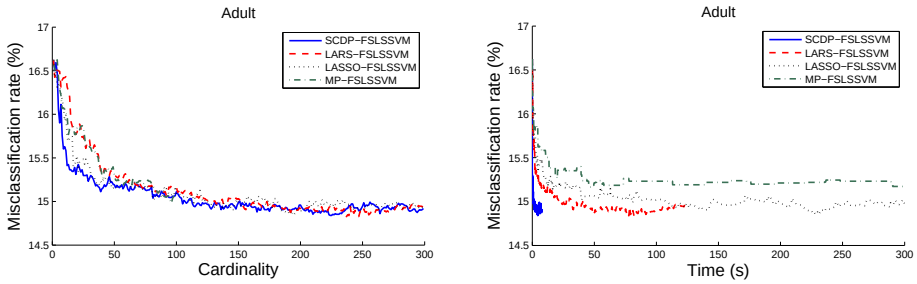
Given a set of (tuned) hyper-parameters, Pareto frontiers are presented in Fig. 5.7a and Fig. 5.8a for respectively the `gamma telescope` and `adult` data sets [11]. For both data sets the different Pareto frontiers converge more or less to the same misclassification rate and similar model sparsity while having a training time which is 7 to 10 times faster than the alternatives. This is mainly due to the fact SCDP does not require an additional de-bias step. As was expected, MP needs many more iterations to converge (convergence is not plotted) compared to its alternatives.

[11] Note that the LASSO in addition can also remove PV from the active set which means that for a certain cardinality multiple misclassification rates can be obtained. In order to enhance the clarity of the figures we only plotted the minimum misclassification rate per cardinality.



(a) *(b)*

Figure 5.7: Given a set of tuned hyper-parameters, Pareto curves and time curves of SCDP, LARS, LASSO and MP applied to (5.22) are shown for the `gamma telescope` data set. The time curves show an average of 20 cumulative plots of the computation time for each of the alternatives per iteration.



(a) *(b)*

Figure 5.8: Given a set of tuned hyper-parameters, Pareto curves and time curves of SCDP, LARS, LASSO and MP applied to (5.22) are shown for the `adult` data set. The time curves show an average of 20 cumulative plots of the computation time for each of the alternatives per iteration.

Probability outcomes

Since we are interested in using the SCDP-FSLSSVM method in the context of speech recognition it is important that we can estimate a-posteriori probabilities. For this purpose we use Platt's method described in Section 3.6 to estimate the probabilities from the latent functions outputs of SCDP-FSLSSVM. The SCDP-FSLSSVM method is initialized using a set of potential PVs containing all training data points. The probability "landscape" produced by the SCDP-FSLSSVM model as well as the fitted sigmoid function (applied to the latent function values) is

shown in Fig. 5.9. In the final PVs marked with the black squares which were automatically selected during training. The RBF kernel was used. From Fig. 5.9 it is is seen that a smooth probability "landscape" might be obtained and that the PVs are distributed over all parts of the data.

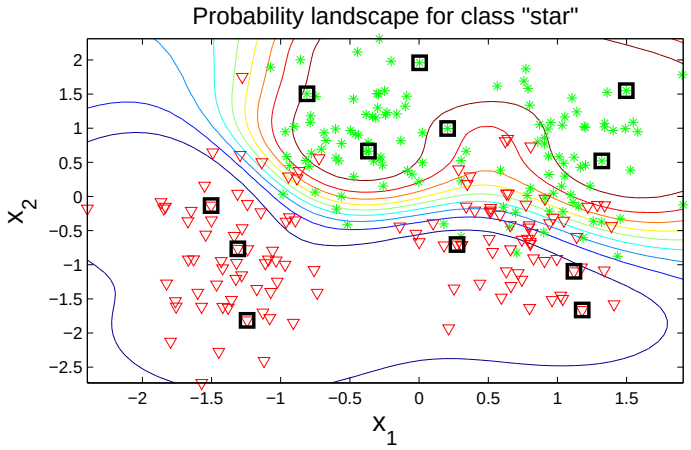


Figure 5.9: The probability "landscape" of SCDP-FSLSSVM (with tuned hyper-parameters) on the `ripley` data set is presented using the method described in Section 3.6 to estimate the Platt's probabilities from the latent functions outputs of SCDP-FSLSSVM. The RBF kernel was used.

5.5.2 Multi-class classification problems

We also evaluated the SCDP algorithm on 3 multi-class data sets using a one-versus-one coding scheme. The pairwise classifiers are combined using the Price (SCDP-FSLSSVM$_{\text{pkpd}}$) method and pairwise coupling (SCDP-FSLSSVM$_{\text{pc}}$) method as was discussed in Section 3.7. These methods require probability outcomes of the methods which we estimate again using the method described in Section 3.6 based on the latent outputs of the individual SCDP-FSLSSVM models. For each experiment the RBF kernel was used and except for attribute standardization no preprocessing was performed. Since probability estimates were already available to combine the different classifiers, we can easily compute the continuous negative log likelihood scores NLL(y, z). We therefore choose to use these scores to select the hyper-parameters (see Section 3.5).

The results are shown in Table 5.5. It is seen that the pairwise coupling method outperforms the simple method of Price for all 3 data sets. Note that there is a

computational cost involved with the pairwise coupling method but results in an improvement in the prediction accuracies as seen. Note that the misclassification rates are slightly better compared to the FS-MKLR results presented in Table 4.4. This is possibly the result of a more thorough search for the optimal model sizes. As a reference we added the results for SVM (using CSA for tuning purposes with $\mathrm{NLL}(y, z)$ as a cost criterion).

Data set	**SCDP-FSLSSVM$_{\mathrm{pc}}$**		**SCDP-FSLSSVM$_{\mathrm{pkpd}}$**		**SVM$_{\mathrm{pc}}$**	
	err (%)	mean(PV)	err (%)	mean(PV)	err (%)	mean(SV)
`satimage`	7.90(±0.27)	298.2(±51.3)	8.06(±0.12)	255.9(±71.1)	8.04(±0.41)	358.3(±67.6)
`usps`	4.05(±0.13)	309.6(±38.9)	4.13(±0.07)	278.3(±35.7)	4.33(±0.05)	154.84(±4.2)
`isolet`	3.52(±0.09)	57.8(±15.2)	3.75(±0.15)	52.4(±19.1)	3.22(±0.1)	80.3(±5.3)

Table 5.5: This table presents the misclassification rates of SCDP-FSLSSVM and SVM models. Hyper-parameters were tuned using a 10-fold cross-validation criterion for each method. Results here are averaged over 5 randomly drawn partitions of the training set (the same partitions are used for the different methods. The multi-class problem is divided into several binary classifiers using a one-versus-one coding scheme. Individual pairwise classifier scores are then combined using the Price (SCDP-FSLSSVM$_{\mathrm{pkpd}}$) method and pairwise coupling (SCDP-FSLSSVM$_{\mathrm{pc}}$) method discussed in Section 3.7.

5.6 Conclusion

In this chapter we investigated a greedy heuristic, called Sparse Conjugate Directions Pursuit, for finding a sparse approximation to a symmetric positive definite linear system. We first explored SCDP on a conceptual level, and relate this to the literature of statistics, signal processing and machine learning. Subsequently, a detailed description of the SCDP implementation is given and its computational complexity is discussed. Then it is explained how SCDP is applied to the LS-SVM framework, resulting in a sparse LS-SVM solution. A main advantage is that a range (starting from 1) of model complexities can be efficiently computed. We carefully examined SCDP's complexity in terms of memory and computational requirements and compared this to other sparse kernel-based methods. Finally, SCDP applied to FS-LSSVM was validated on a number of public available benchmark data sets. This results into the following main conclusions:

- Misclassification scores are similar to those of LS-SVM while training times are much smaller and sparse models are obtained.

- Similar prediction accuracies as those of SVM were obtained, while SCDP usually produces much sparser models.
- SCDPP can improve upon the training speed but υ has to be carefully chosen to avoid deviant (in SCDP sense) results.
- k-center clustering as a preprocessing step to select the initial set of PVs is a valuable tool to speed-up the training process. These PVs need not to be a member of the training set. The initial pool of PVs sets an upper bound on the number of unique PVs in the multi-class setting which has an impact on model evaluation times.
- We indicated that, when applying to FS-LSSVM, SCDP can indeed outperform MP, LARS and LASSO in terms of speed while having a comparable accuracy.
- We validated a multi-class extension based on a probabilistic interpretation of the latent model outcomes and empirically showed that the pairwise coupling method gave consistently better results than the method of Price.

Chapter 6

Segment-based Phone Recognition using Kernel Methods

In this chapter we describe a framework for ASR for integrating the previously explained kernel-based methods. Instead of using frame-based features, which are common in the current state-of-the-art speech recognition systems, features on the phone level are used. Our motivation to use this framework is two-fold: (i) a framework which can handle duration and trajectory information on the phone level has the potential to overcome limitations from the standard HMM, and (ii) including additional features increases the dimensionality of examples which can decrease performance of the traditional HMM systems. Kernel-based methods are known to generalize well in high dimensional spaces. The segment-based framework divides the recognition process into two main tasks namely (i) generating measures of confidence on different segmentations of the full utterance and (ii) phone labeling for a given segmentation. Kernel-based methods are particularly useful for the latter. In Section 6.1 the formalized segment-based speech recognition from Section 2.3 is reviewed. In Section 6.2 models for phone segments are given and in Section 6.3 a simple model which judges different utterance segmentations is formulated. Section 6.4 describes the final segment-based phone recognition process. In Section 6.5 alternative related kernel-based classifier methods in a segment-based ASR context as found in the literature are described. Section 6.6 presents details about the phone classification as well as recognition experiments on the TIMIT

benchmark corpus.

6.1 Introduction

The HMM state-of-the-art techniques typically use a frame-based approach assuming conditional independence of the frames as described in Section 2.2. Adjacent feature vectors, especially those within the same phonetic segment, often exhibit smooth transitions and are highly correlated in general. This violates the conditional independence assumption imposed by most HMM models. However, dependence between features computed in different phonetic segments is weaker. These observations motivate a framework making fewer assumptions about the dependence between observation frames especially for those occurring within a phonetic segment [69]. In order to describe larger contextual windows of speech more features might be needed. Generative techniques, such as the standard HMM approach, need a very large corpus to model the distributions of the data accurately in such high dimensional spaces. Kernel-based classifiers such as SVM, LS-SVM and KLR directly focus on separating the classes and are known to generalize well even in high dimensional feature spaces [159]. Moreover, these kernel methods are based on convex optimization. This motivates why we opted to use kernel-based classifiers in a segment-based framework to tackle automatic speech recognition.

In Section 2.3 the a-posteriori segment-based speech recognition process was formalized as follows:

$$\hat{\underline{\mathrm{W}}} \propto \arg\max_{\underline{\mathrm{W}}} \Pr(\underline{\mathrm{W}}) \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{W}}) \sum_{\underline{\mathrm{S}}} \frac{\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}, \underline{\mathrm{X}}) \Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})}{\Pr(\underline{\mathrm{U}})}. \tag{6.1}$$

Since systems that produce accurate phone sequences are likely to produce good word recognition results, we concentrate on recognizing phone sequences. This can be considered as a special case of word recognition in the sense that the words are the basic units. In this case there is no word level and we have that $\Pr(U) = \Pr(U|W)\Pr(W)$, which reduces the equation (6.1) to

$$\hat{\underline{\mathrm{U}}} = \arg\max_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}, \underline{\mathrm{X}}) \Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}). \tag{6.2}$$

From this formula it is seen that the task of segmenting the full speech utterance into smaller consecutive units emerges in an integrated way with classifying individual segments. The two different dedicated tasks are (i) a unit (phone) component which hypothesizes an utterance given a set of assumably correct segments and (ii)

a segmentation component which judges the segmentation. Within this process we focus on the main first component which corresponds with the unit probability factor $\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}, \underline{\mathrm{X}})$ as defined in Section 2.3. Since in this chapter phones are chosen as units, this factor is termed phone probability factor (generated by the phone model). For the segment probability factor (generated by a segment model), we will use an existing HMM recognizer. During decoding incorrect segments which have no valid interpretation as lexical units (i.e., too long, too short, etc.) might occur. In SVM literature these are called universum data [188], are possible. Since phone models are typically trained only on non-universum data, they might fail on universum data. For this purpose the segment model assigns a low probability to such non-phonetic segments. However, we will empirically indicate that a phone model, trained with both non-universum and universum data, can increase recognition performance.

The standard kernels (e.g. RBF kernel) which were used in the previous chapters operate on fixed-length vectors while phone segments are of variable length. Fortunately, it is quite straightforward to find a fixed-size segmental feature set such as in e.g. [36] which can be utilized by the kernel-based variants.

Our contributions in this part are:

- The integration and validation of SVM, FS-MKLR and SCDP-FSLSSVM in a segment-based speech recognizer.
- The comparison of kernel-based methods in terms of accuracy, probability scores, model complexities, training and evaluation speed on speech data.
- The introduction of a cheap and efficient technique which allows to incorporate information about incorrect segmentations in the unit component without increasing the number of model parameters. We point out experimentally that this improves the final phone recognition rate.

6.2 Modeling the Phone Probability Factor

As in Section 2.1, let $\underline{\mathrm{X}} = (x_1, \ldots, x_{N_x}), \underline{\mathrm{U}} = (u_1, \ldots, u_{N_u})$, and $\underline{\mathrm{S}} = (s_1, \ldots, s_{N_u})$ be the sequence of frames, units and segments respectively. The phone probability factor from (6.2) can be decomposed as follows

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}, \underline{\mathrm{X}}) = \prod_{i=1}^{N_u} \Pr(u_i|u_1, \ldots, u_{i-1}, \underline{\mathrm{S}}, \underline{\mathrm{X}}). \tag{6.3}$$

In case we assume a 2-gram phone language model [83] and use segmental-based features, we can approximate equation (6.3) further to

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}},\underline{\mathrm{X}}) \propto \prod_{i=1}^{N_u} \Pr(u_i|u_{i-1}, f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})), \tag{6.4}$$

where τ_{i-1} and $\tau_i - 1$ respectively indicate the first and last indices of the i-th segment in sequence $\underline{\mathrm{X}}$.

Using Bayes' rule, the fact that features are only selected based on the frames of the current unit, and the assumption that we only condition on u_i and not on u_{i-1}, we can transform the latter approximate factor as follows

$$\begin{aligned}
\Pr(u_i|u_{i-1}, f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})) &= \frac{\Pr(u_i, f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})}{\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})} \\
&= \frac{\Pr(u_i|u_{i-1})\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_i, u_{i-1})}{\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_{i-1})} \\
&\propto \frac{\Pr(u_i|u_{i-1})\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})|u_i)}{\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))} \\
&\propto \frac{\Pr(u_i|u_{i-1})\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}), u_i)}{\Pr(f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))\Pr(u_i)} \\
&\propto \frac{\Pr(u_i|u_{i-1})\Pr(u_i|f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))}{\Pr(u_i)}.
\end{aligned} \tag{6.5}$$

Combining (6.4) and (6.5) gives

$$\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}},\underline{\mathrm{X}}) \propto \prod_{i=1}^{N_u} \frac{\Pr(u_i|u_{i-1})\Pr(u_i|f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1}))}{\Pr(u_i)}. \tag{6.6}$$

The denominator in (6.6) defines 1-gram probabilities. The first factor in the numerator defines a 2-gram language model. Note that this formula can be easily generalized to the N-gram case. In case we intend to model the second factor in the numerator by means of a kernel method using kernels as e.g. RBF kernel, we have to tackle the technical problem that the segments are of different lengths. Fortunately, it is straightforward to find a segmental feature set that yields a

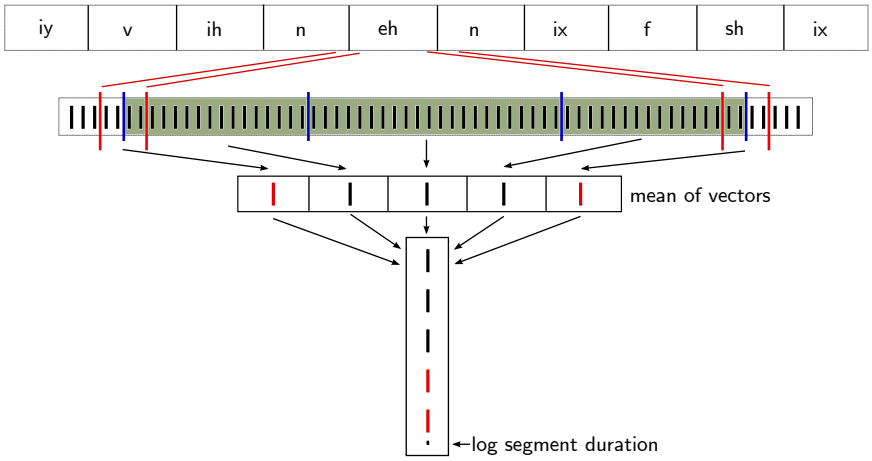


Figure 6.1: Each phone segment is broken into three regions using the ratio 3-4-3. Frames belonging to each of these regions were averaged resulting in three vectors. In addition, frames belonging to a window region of 50 ms width centered at the start and end of the phonetic segments were averaged. These 5 vectors plus a feature indicating the log-duration of the phone segment were stacked to one fixed length vector.

segment classification rate similar to or better than those of HMM phone models. Examples can be found in [68] and [36]. The mapping function f_s outputs a fixed length segmental feature vector which is used to relate segments to phones. In this work we use the technique described in [36]. Here, each phone segment was broken into three regions using the ratio 3-4-3. Different frames belonging to region are averaged resulting in three vectors. Additionally, frames belonging to a window region of 50 ms width centered at the start and end of the phonetic segments were averaged. These 5 vectors as well as a feature indicating the log-duration of the phone segment are stacked into one single fixed length vector [1]. This is displayed schematically in Fig. 6.1.

6.3 Modeling the Segmentation Probability Factor

The segmentation model judges segmentations of a speech signal. Given the segmentation a unit sequence hypothesis can be generated using the phone model.

[1] In case a phone segment contains only 1 frame then 3 copies of the same vector were used to represent the 3 phone regions. In case 2 frames are available they are assigned to the left and right region and their average is used to represent the mid-region.

There are several possibilities for modeling $\Pr(\underline{S}|\underline{X})$ (6.2). A possible solution is to execute a frame-based recognizer, retain the N best paths produced by it, and evaluate only these paths using the segment model [199]. In this case one can ignore the factor $\Pr(\underline{S}|\underline{X})$ by assuming that the segmentations proposed by the frame-based recognizer all have the same segmentation probability. Alternatively, in [114] the authors proposed to calculate the segmentation probability as a product of boundary probabilities, estimated using Multi-Layer Perceptrons (MLP). Instead of making a product in [126] one uses an additional MLP to estimate $\Pr(\underline{S}|\underline{X})$ given the boundary probabilities. A third option is to construct an estimate of $\Pr(\underline{S}|\underline{X})$ as follows

$$\Pr(\underline{S}|\underline{X}) \propto \prod_{i=1}^{N_u} \Pr(s_i|\underline{X}_{\tau_{i-1}}^{\tau_i-1}), \tag{6.7}$$

where independence between different segments is assumed. To train such model one can construct an extra (also called "anti-phone") class with universum data, i.e. data obtained from incorrect segmentations [68]. This extra class can then be used to build a dedicated binary classifier (with the non-universum data as the positive class and the universum data as the negative class) or to extend the phone classifier with an extra class. The universum training examples can be generated artificially over a manually segmented corpus [155] or in a adaptive manner by providing the model with negative examples encountered during (mis)recognition [4].

This work focusses on building a kernel-based phone classifier. Therefore, we choose to use the likelihood probabilities, which we already have from our HMM recognizer, for estimating the segmentation probability factor. Note that other, possibly more suited, segment-based models could be used. We prefer this approach above that of using N best segmentations of the HMM recognizer because in this way the segmentation model as well as the phone model can jointly decide on the segmentation.

In case factor $\Pr(\underline{S}|\underline{X})$ is used in a maximization formulation, it can be replaced by $\Pr(\underline{X}|\underline{S})$ using Bayes' rule since $\Pr(\underline{X})$ is constant and we assume $\Pr(\underline{S})$ to be constant for all permitted $\underline{S}$. A segment duration model in the form of a non-uniform $\Pr(\underline{S})$ could refine the approach.

Using marginalization of the HMM recognizer likelihoods and by again applying Bayes' rule we can formulate

$$\begin{aligned}\Pr(\underline{\mathrm{X}}|\underline{\mathrm{S}}) &= \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}, \underline{\mathrm{U}}|\underline{\mathrm{S}}) \\ &= \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}}) \Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}}) \\ &\propto \sum_{\underline{\mathrm{U}}} \Pr(\underline{\mathrm{X}}|\underline{\mathrm{U}}, \underline{\mathrm{S}}) \Pr(\underline{\mathrm{U}}), \end{aligned} \tag{6.8}$$

where factor $\Pr(\underline{\mathrm{U}}|\underline{\mathrm{S}})$ is approximated as $\Pr(\underline{\mathrm{U}})$. This might be because we do not expect to gain much when conditioning the utterance sequence on the segmentation without knowledge of the acoustic vectors.

With (6.8) and incorporating the standard HMM independence assumptions, we can replace $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ by

$$\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}) \propto \prod_{i=1}^{N_u} \sum_{c=1}^{C} \left(\Pr(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1} | s_i, u_i = u'_c) \Pr(u_i = u'_c) \right). \tag{6.9}$$

Note that $\Pr(\underline{\mathrm{U}})$ does not refer to the hypothesis which is searched for in maximization problem (2.18) but is used as a dummy (prior) variable to obtained the actual segmentation probability. It is therefore motivated to use 1-gram assumptions in this case which also simplifies the implementation of the search algorithm considerably.

6.4 Phone Recognition

Substituting (6.6) and (6.9) into (6.2) gives the formulation of our phone recognition task

$$\begin{aligned}\hat{\underline{\mathrm{U}}} = \arg\max_{\underline{\mathrm{U}}} \sum_{\underline{\mathrm{S}}} \prod_{i=1}^{N_u} \sum_{c=1}^{C} &\left(\Pr(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1} | s_i, u_i = u'_c) \Pr(u_i = u'_c) \right) \\ &\frac{\Pr(u_i | f_{\mathrm{s}}(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i - 1}))}{\Pr(u_i)} \Pr(u_i | u_{i-1}). \end{aligned} \tag{6.10}$$

Note that in case a path-algorithm as Viterbi for extracting the most likely segmentation and corresponding phone sequence, the sum over $\underline{\mathrm{S}}$ in (6.10) is replaced by the max operator over $\underline{\mathrm{S}}$.

In the experimental section the continuous speech recognition performances are presented as phone error rates (PER) defined as

$$PER = \frac{N_I + N_D + N_S}{N_C} \tag{6.11}$$

where N_I is the number of inserted phones, N_D the number of deleted phones, N_S the number of substituted phones in the recognized phone sequence. N_C is the number of phones in the correct sequence.

Remark 24. Notice that in case a 1-gram language model is used, it would cancel out the denominator $\Pr(u_i)$ in (6.10). Hence, using it simplifies our model but deteriorates the PER as is observed in the experimental section.

Since there is information available about incorrect segmentations and since using the learning with a universum technique described in Section 4.2.4 is not too computationally demanding we can easily incorporate the extra unlabeled data while constructing our phone model. Note that the number of phone model parameters is not increased by this technique which means that the time for evaluating new data remains the same as before. Besides discriminating the phones differently the unit probability factor now goes beyond its initial scope and can jointly decide on the segmentation together with the segment probability factor. Note that our approach differs from that of e.g. [68], [155] since we do not consider the universum data as an extra class (which would increase the model complexity) but rather use the universum information to define a criterion which decides where to locate the classification boundary of the phone model. In Section 6.6 we experimentally indicate that adding information of the universum (incorrect segmentations) to build the phone classifier will improve the recognition rate.

6.5 Related work

The primary focus of this thesis was on the development of kernel-based methods. A second, as described in this chapter, is integrating these generic methods into a framework which can handle a labeling procedure on a sequence of speech units of unknown length. In the literature a number of cases are found which also use kernel-based methods in a sequence labeling context. We now briefly discuss such approaches which were applied to segment based speech recognition and state relations and differences.

- In [64], the authors presented a hybrid HMM/SVM approach for continuous speech recognition using the N-best rescoring paradigm. Similar to our

approach they issued a segment-based recognizer. The HMM paradigm is used to generate the most likely segmentations via a dynamic programming approach and the SVM classifier can then postprocess these segmentations to predict a word sequence. They employed a similar simple segmental feature extraction method as ours. Hence, their method addressed both the issue of variable length sequences and the issue of sequence labeling with unknown segmentation. To be able to solve the large scale multi-class problem, they solved the multi-class problem by coupling binary classifiers on pairs of units (phones). The decomposition of the large multi-class problem into several smaller allowed the use of SVM. Our algorithms scale to larger data sets. Moreover, they employed rescoring of an N-best list of HMM hypotheses. In this case sentences with deletion and insertion errors can not be corrected. In this thesis we do not rely on an N-best list. Using an enhanced kernel-based phone recognition system, which incorporates universum data (from non-phone segments) in the kernel-based phone model, we let the phone and segmentation model together decide upon the segmentation.

- Other approaches concentrate on defining a HMM-based feature map which maps a time-series into a fixed-dimensional vector. This was first used in a general context in [87] where a Fisher score mapping was used, i.e. the gradient space of a HMM (Fisher score mapping). The idea was later on applied to speech recognition by e.g. [162], [111]. In [11] the authors propose to use the set of HMM likelihoods (on variable length sequences) as an input to a discriminative logistic regression model and show that this can improve recognition results. Our described recognizer framework could use these HMM based features instead of using the simple segmental feature extraction method.

- Some authors describe kernel functions which can deal with varying length sequences. In [161] the authors introduce the Dynamic Time Alignment Kernel (DTAK) for applications in speech recognition which is a sequence kernel which directly operates on pairs of time-series. The use of their kernel is motivated by the dynamic time warping (DTW) [149] algorithm and defines an alignment score along the optimal alignment path. An enhanced version is the Global Alignment (GA) kernel [38] which sums up the contributions for all possible alignment paths. These kernels have the potential to overcome the weaknesses described in Section 2.2.4, and can be directly applied to the described kernel-based methods.

- Instead of a maximum likelihood Markov model such as common in speech recognition in [23], a Maximum A-Posteriori (MAP) Markov model is

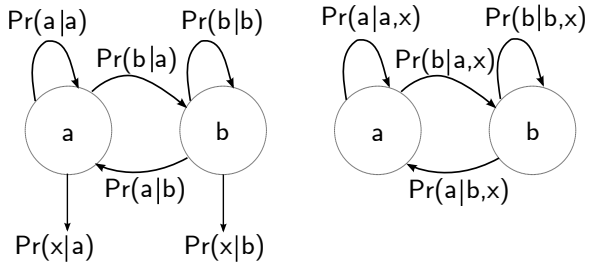


Figure 6.2: The left figure is a two state Markovian model as is often used in speech recognition. The right figure denotes a two state Markovian maximum a-posteriori model, where transition probabilities between states are modulated by the observation vector x.

considered (see Fig. 6.2). All transition probabilities are now conditioned on the observed data. The states, unlike HMMs, directly encode phones in this case. These conditional probabilities can be modeled by discriminative kernel-based models. A forward decoding of the state transition probabilities with the sum-product algorithm directly produces the MAP sequence. The parameters in the probabilistic model are trained using a recursive scheme that maximizes a lower bound on the regularized cross-entropy. Similar to Expectation-Maximization (EM), the recursion performs an expectation step on the outgoing state of the transition probability model, using the a-posteriori probabilities produced by the previous maximization step. While our segmental approach uses an language model which is not conditioned on the observation frames, this method inherently learns phone transition probabilities but a higher computational cost is involved. The full learning problem is decomposed into C, C-class MKLR models ($C = 39$ in their experiments) which for a single training process are iteratively trained until a certain context length is reached. To solve the MKLR problems instead in [23] the authors approximate the cross-entropy by a quadratic formulation resulting in sparse algorithm called GiniSVM which is solved by an online training scheme. Instead of using the GiniSVM approximation our efficient and sparse FS-MKLR algorithm can be directly plugged into their framework. A similar segmental feature extraction as ours was used in their phone recognition experiments on the TIMIT benchmark data set.

- In [102] (chapter 5) the authors also propose a kernel-based learning objective based on full speech examples (utterances). Per example, besides acquiring frame-based acoustical features also segmental-based (on the phone level)

duration and transition features are used. To solve the learning problem an iterative algorithm was proposed but until now this led to slow convergence rates and suboptimal results.

Remark 25. Note that LR as discussed in Section 4.2.1 can be derived from an entropy maximization framework as denoted [10]. In this context the logit models are known as MaxEnt models and entropy maximization is performed using moment constraints. In [192] a method is proposed to additionally formulate distribution constraints which are more suited for continuous features. In order to derive a convenient objective for optimization purposes the authors provide a solution by converting the input data to a higher dimensional space similar to the what is performed by the feature map $\varphi(\cdot)$ in our case. To achieve this, cubic-splines are used instead of kernels. Analogous to choosing a number of PVs, in their work the number of spline knots need to be specified. As in this text this number of knots is selected based on a validation set criterion. Our proposed framework has the advantage that it can be used with any positive semi-definite kernel function.

6.6 Experiments

To test the performance of our devised kernel methods on speech data we used the `timit` database [176] with MIDA-based features (for details about `timit` and the type of features see Appendix B).If we refer to a state-of-the-art HMM phone recognizer it has the following configuration. In total 51 context-independent acoustical phone models (one noise model is included) with 2 to 4 states (left-to-right models) where the number of states dependents on the average phone length. This gives a total of 141 states (from which 13 states are shared between 2 or 3 phones). In total $5,710$ Gaussians (average 131.2/state) were used. In all experiments we used the RBF kernel (3.1) since this kernel outperforms other alternatives as was seen in several previous publications [36], [63], [156]. Since the previous experiments showed that stratified k-center clustering was preferred as a PV selection technique, we used this option for the FS-(M)KLR and SCDP-FSLSSVM algorithms in this section. Note that the size of the `timit` corpus is considered too small to be used for training context-dependent phone models.

6.6.1 Phone classification

At first we report results on phone classification where we assume a given segmentation of speech utterances (note that `timit` corpus has been hand labeled

and -segmented). The objective is then to find a statistical relation between the variable length segments and phone labels. In order to use machine learning techniques such as K-Nearest Neighbors (KNN) [78], Linear Discriminant Analysis (LDA) [78], SVM, FS-(M)KLR and SCDP-FSLSSVM on such variable length segments we must encode the waveform information as a fixed-length vector. We chose the same simple method of encoding the variable length segment information in a vector of fixed length as described in [36], and Section 6.2. This resulted in $142,910$ training examples and $51,681$ test examples. Since the frames we used are composed out of 36 features (see Appendix B) the final phone feature vector is 181-dimensional.

Table 6.1 presents classification accuracies on the `timit` full test set for a number of algorithms. The columns **acc**, $\mathbf{t}_{tr}$, and $\mathbf{t}_{es}$ respectively give the percentage of correctly classified phone segments, the associated training time, and evaluation time. The different algorithms used are: SVM, FS-MKLR which refers to the all-at-once learning algorithm in Chapter 4, SCDP-FSLSSVM as described in Chapter 5, Linear Discriminant Analysis (LDA), K Nearest Neighbors (KNN) and a state-of-the-art HMM phone classifier. Methods SVM, FS-KLR (binary fixed-size KLR), and SCDP-FSLSSVM solve the multi-class problem using a one-versus-one coding scheme. In case such a scheme is used, subscripts $\cdot_{\text{pc}}$, $\cdot_{\text{pkpd}}$, and $\cdot_{\text{vs}}$ respectively indicate whether the pairwise coupling idea, the method by Price as described in Section 3.7, or a voting scheme are utilized. The a-posteriori probabilities for SVMprob and SCDP-FSLSSVM are estimated as described in Section 3.6. SCD-FSLSSVM starts with an initial set of PVs of size $80,000$. For each of the kernel methods an RBF-kernel was used.

In case of multi-class classification using binary coding schemes, one could opt to tune hyper-parameters per binary submodel. In Section 4.4 it was seen that this only led to small improvements while having a larger training and evaluation time. We therefore focus on using 2 shared hyper-parameters (regularization and kernel parameter).

In order to speed-up our automatic tuning process we randomly selected a set of $20,000$ data points which is used for tuning the hyper-parameters using cross-validation. After tuning we used the full train set to train the algorithm using the obtained hyper-parameters. In case of FS-MKLR we ceiled the maximum number of PVs to $8,000$ and training on the full training set was performed using the reduced memory implementation (see 4.3.3) to make computation feasible on our computer (CPU: quad-core 2.66GHz, RAM: 8Gb). The SVM experiments are performed using the LIBSVM toolbox [117] where multi-class problems are encoded in a one-versus-one coding scheme.

A closer look at Table 6.1 gives the following remarks:

- FS-MKLR is much faster in evaluating new observations compared to SVM while having only a slightly worse classification performance. This property will be useful for the phone recognition task explained in the following section.
- All kernel classifiers outperform the state-of-the-art HMM classifier which obtained a classification accuracy of 78.6% on the full test set using a 1-gram language model.
- It becomes apparent that FS-MKLR performs better compared to FS-KLR (using one-versus-one coding scheme) using the same set of PVs, which is consistent with findings in Section 4.4.3.
- Training a FS-MKLR model becomes computational demanding when the number of PVs exceeds $8,000$. The one-versus-one coding scheme method allow a higher number of PVs to be included. In case the number of PVs is increased the classification scores also increase but still are slightly inferior to those reported for SVM.
- The max-wins voting which gives a classification accuracy of 82.9% which is inferior to the pairwise coupling idea described in Section 3.7 which gives an accuracy of 83.2%. A similar conclusion was made in [51]. An analogue conclusion can be drawn for SCDP-FSLSSVM where pairwise coupling gave better results than using the method of Price (see Section 3.7) to combine the probability estimates. However, while being computationally more demanding the FS-KLR pairwise coupling led to no significant improvement in classification performance.
- Compared to FS-KLR$_{\text{pc}}$, SCDP-FSLSSVM$_{\text{pc}}$ has a faster evaluation time due to sparser models. The 741 models built by SCDP-FSLSSVM have $1,184$ PVs on average while those of FS-KLR$_{\text{pc}}$ have $1,723$ PVs on average. Note that SCDP-FSLSSVM$_{\text{pc}}$ requires an additional computation of a sigmoid function for the probability estimates.

Remark 26. When comparing the training and evaluation times it should be noticed that the LIBSVM toolbox is implemented in the programming language C++ while the other approaches are implemented in MATLAB.

When comparing the relationship of the FS-MKLR, SCDP-FSLSSVM$_{\text{pc}}$ and SVMprob$_{\text{pc}}$ outputs to Bayes probabilities, we display in Fig. 6.3 the histogram of FS-MKLR outputs to the fraction of segments correct for each output that is in a

alg	acc(%)	$\mathbf{t}_{tr}$	$\mathbf{t}_{es}$
$\text{SVMprob}_{\text{pc}}$ (89,563 SVs)	83.2	10h	2h
SVM_{vs} (89,563 SVs)	82.9	3h	1h
$\text{SCDP-FSLSSVM}_{\text{pc}}$ (66,001 PVs)	82.7	9.6h	44min
$\text{FS-KLR}_{\text{pc}}$ (80,000 PVs)	82.5	23h	59min
$\text{FS-KLR}_{\text{pkpd}}$ (80,000 PVs)	82.5	23h	50min
FS-MKLR (8,000 PVs)	81.9	86h	0.5min
$\text{FS-KLR}_{\text{pc}}$ (8,000 PVs)	80.5	50min	7min
$\text{FS-KLR}_{\text{pkpd}}$ (8,000 PVs)	80.5	50min	6min
LDA	66.5	4.7min	0.03min
KNN (k=1)	67.8	-	11.5h

Table 6.1: Classification accuracies on the `timit` full test set for different algorithms are shown. The columns **acc**, $\mathbf{t}_{tr}$, and $\mathbf{t}_{es}$ respectively give the percentage of correctly classified phone segments, the associated training time, and evaluation time. FS-MKLR is the all-at-once learning algorithm as in equation (4.1) in Section 4.2.1. FS-KLR couples binary FS-MKLR models using a one-versus-one coding scheme. SCDP-FSLSSVM described in Chapter 5 using a one-versus-one coding scheme. The subscripts $\cdot_{\text{pc}}$ and $\cdot_{\text{pkpd}}$ indicate respectively whether the pairwise coupling idea or the method by Price as described in Section 3.7 are utilized. The SVM entry uses one-versus-one encoding with max-win voting. The a-posteriori probabilities for SVMprob and SCDP-FSLSSVM are estimated as described in Section 3.6. For each of the kernel methods an RBF-kernel was used. Note that the final FS-MKLR model is trained using the reduced memory implementation as defined in 4.3.3.

given bin range. In fact, each data point in Fig. 6.3 represents the average probability of being correct over all the input data for the FS-MKLR outcomes in each of the possible activation bins. This fraction (i.e. the probability of being correct) ideally corresponds to the Bayes probability [53]. A perfect Bayesian classifier should have a probability equal to the probability of being correct, which corresponds to the diagonal in the figure. From the figure it is seen that although SVM is not trained in a probabilistic framework it still approximates the Bayes probabilities rather well. The extremes fit nicely with the desired probabilities. In the middle range the Bayes probabilities are underestimated for the SVM outputs. A similar conclusion can be drawn for SCDP-FSLSSVM. In order to compute the probability scores, SVM and SCDP-FSLSSVM respectively use 89,563 and 66,001 PVs in contrast to the FS-MKLR model which only uses 8,000 PVs and still estimates the Bayes probabilities especially in the low range quite nicely. In [14] a similar experiment is performed using Multi-Layer Perceptrons (MLPs) but instead of

segment classification, frame-wise classification was used on the DARPA `resource management` data set. The a-posteriori MLP probabilities did not approximate the Bayes probabilities as well as those presented in our experiment.

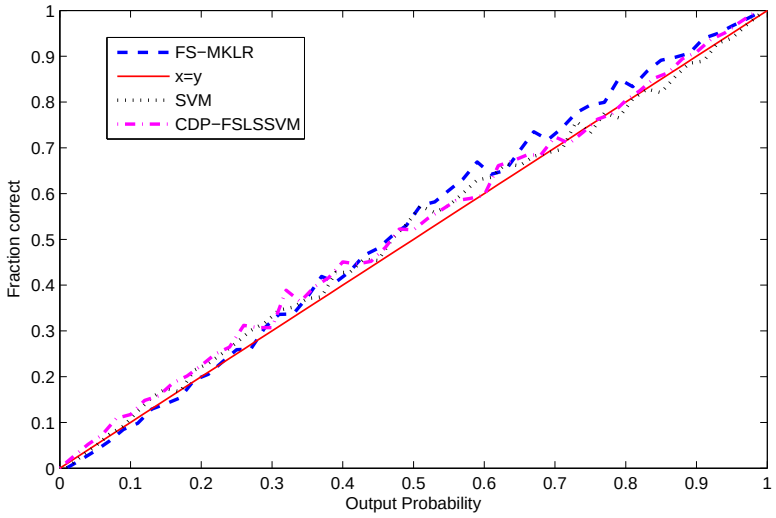


Figure 6.3: Histogram showing the relationship between FS-MKLR (using 8,000 PVs), SCDP-FSLSSVM (using 66,001) and SVM outcome probabilities (using 89,563 PVs) and the percentage correct for each bin. The scores were computed on the independent `timit` full test set.

6.6.2 Phone recognition

Unlike the previous section we will not rely on a known segmentation. In case of phone classification only correct segments are processed, with phone recognition so called universum segments are also considered. These are segments composed out of frames belonging to different adjacent phones when a bad segmentation is employed.

In order to explore some properties of the probability scores we first define the averaged maximum probability score as

$$p^{amps} = \frac{1}{N}\sum_{i=1}^{N} \arg\max_{c}(p_{ci}), \tag{6.12}$$

where N is the number of data samples. In Fig. 6.4(b) we investigate the averaged maximum probability scores for universum segments as a function of the segment

model	p^{amps}
FS-MKLR	0.60
FS-MKLR$_{\mathcal{U}}$	0.27

Table 6.2: The table shows the maximal averaged probability scores for universum data created using the `timit` full test set using the FS-MKLR model from previous section and a tuned FS-MKLR$_{\mathcal{U}}$ using the universum penalization term described in Section 4.2.4. In both cases $8,000$ PVs were used.

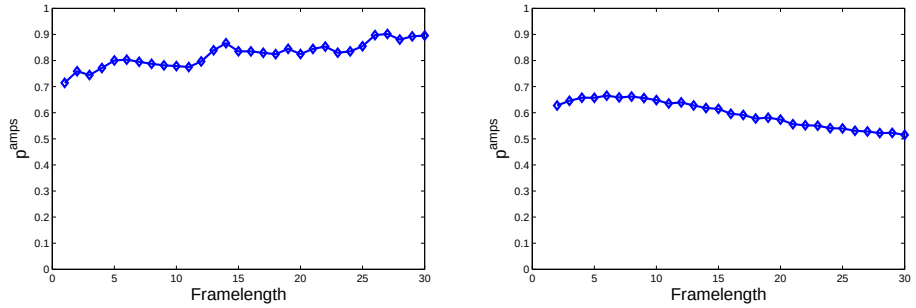
length. As was expected, due to training the model only for classification (with given segmentation) in most cases the universum segment is assigned to a certain class with relatively high probability. From the figure we can also learn that for longer frames the model is less confident about the chosen phone class. The same trend is seen in Fig. 6.4(a) where the averaged maximum probabilities for non-universum data is plotted. Here the model is more certain about the class label in case of larger frame lengths. Intuitively, this seems reasonable because more frames are averaged in the preprocessing step in case of larger segments which possibly reduces the effect of noise in the data.

Next we take universum information into account to build our phone model as was pointed out in Section 4.2.4. For this purpose we generated universum data using the training set (with exception of the examples that are in the validation set). Given a speech utterance, a set of possible segmentations, with minimal segment length set to 2 frames and maximal length equal to 30, is created. All segmentations except those which maximally deviate only 1 frame per phone boundary from the correct one are considered as universum data. From this set a random subset with $150,000$ examples was selected.

We tuned the hyper-parameter $\nu_{\mathcal{U}}$ on a small validation set using Phone Error Rate (PER) as a cost criterion. In Table 6.2 we consider the maximal average phone probabilities for all universum segments for the FS-MKLR as well as the FS-MKLR$_{\mathcal{U}}$ model. As was expected it is seen that this value is reduced in case of using the universum information. Indicating that less probability mass is assigned to universum data.

We proceed as follows

- Given an utterance, a set of possible segmentations, with minimal segment length set to 2 frames and maximal length equal to 30, is created.
- Next, all possible segments are scored by computing (6.10).



(a) FS-MKLR on non-universum data

(b) FS-MKLR on universum data

Figure 6.4: The figures show the averaged maximal probability score p^{amps} in function of the considered phone segment length for non-universum data from the `timit` full test set in subfigure (6.4a) and for universum data created using the `timit` full test set in (6.4b). In both cases $8,000$ PVs were used.

- Obtain the most likely segmentation using a Viterbi decoder.
- Finally, the decoded phone sequence was compared to reference transcriptions using Levenshtein's distance to evaluate different sources of errors.

Remark 27. Since the phone model favors longer phones (see Figure 6.4a) we added an empirically tuned offset factor (tuned on a small validation set selected from the training database) to the log probabilities to compensate for this effect. We also raised the segment probability factor $\Pr(\underline{S}|\underline{X})$ and 1-gram factor $\Pr(\underline{U})$ in (6.10) to an empirically tuned power of 0.02 (using the same small validation set).

Remark 28. Note that using the Kai-Fu Lee mapping to 39 phone classes implies that there are repeated phones in the reference sequence. For instance, consider a phone sequence $(/ey/, /pcl/, /pau/)$ which will be transformed to $(/ey/, /pau/, /pau/)$. Since we choose to train our models (in case of the kernel-based alternatives) on the reduced set of 39 classes (hence there is no model for $/pcl/$), they cannot identify such a segmentation but instead are biased towards recognizing $(/ey/, /pau/)$ which generates a deletion error. We believe that ignoring this fact affects our results negatively in contrast to others which postpone the mapping to just before scoring the sequences (as our HMM recognizer does). When considering the `timit` full test set this situation occurs in 94.2% of the cases for the silence class $/pau/$. It is therefore that multiple runs of identical phones in the decoded as well as the reference sequences were collapsed to single phone instances to reflect true insertion errors.

LM	FS-MKLR		SVM		FS-MKLR$_{\mathcal{U}}$	
	(I,D,S) (%)	PER (%)	(I,D,S) (%)	PER (%)	(I,D,S) (%)	PER (%)
no LM	(4.69, 10.40, 16.02)	31.10	(4.13, 11.92, 15.42)	31.47	(4.68, 8.03, 16.10)	28.81
2-gram	(4.01, 9.14, 16.17)	29.31	(4.16, 9.52, 15.52)	29.20	(4.84, 7.04, 16.15)	28.03
3-gram	(4.09, 8.50, 15.74)	28.32	(4.82, 8.18, 15.14)	28.13	(4.36, 7.27, 15.70)	27.32
4-gram	(4.59, 7.80, 15.73)	28.13	(4.44, 8.60, 14.98)	28.02	(4.52, 7.05, 15.61)	27.17

Table 6.3: Phone recognition rates using FS-MKLR, FS-MKLR$_{\mathcal{U}}$ and SVM as phone probability models were the SVM contains 89,563 PVs and the FS-MKLR models use 8,000 PVs. The column labels I,D,S respectively indicate the number of insertions, deletions and substitutions. The abbreviation LM stands for language model.

In Table 6.3 we present our phone recognition results obtained on the `timit` full test set. We notice that using the SVM phone model we have a PER which is slightly better, but has a much larger evaluation time than that of the FS-MKLR alternative. Without using a language model recognizing the phone sequences in the `timit` full test set using the SVM model required 121 against 2.6 hours in case FS-MKLR using 8,000 PVs Note that the state-of-the-art HMM needed 41 minutes to process the same data). No results for 1-gram language models are shown in Table 6.3 since these only had a negative impact on the recognition rate, consistent with the remark made in Section 6.4. Including universum data into the phone classification model improves the PER significantly. If no language model is used, we have a 2.66% reduction of the PER. In this case we have a recognition rate which is comparable to that of our state-of-the-art frame-based HMM using a 1-gram shown in Table 6.4. However, using N-grams does not result in as much performance improvement as is the case with the HMM architecture. Although the performance is consistently worse, for FS-MKLR and SVM phone models the same PER reduction rate is found as in case of the HMM architecture. Investigating the cause of this phenomenon in case of using universum data into the FS-MKLR model is subject to future research. In case we neglect repeating phones in the reference sequence the scores are consistently 0.4% worse than those presented in Table 6.3. We also remark that the HMM recognizer in total uses 448,118 parameters and the FS-MKLR, without the segmentation probability factor, has 312,039 parameters.

LM	(I,D,S) (%)	PER (%)
no LM	(3.04, 9.23, 17.99)	30.26
1-gram	(2.97, 8.98, 17.01)	28.97
2-gram	(2.70, 7.65, 16.36)	26.70
3-gram	(3.17, 6.50, 15.96)	25.62
4-gram	(3.41, 6.16, 15.71)	25.28

Table 6.4: Phone recognition rates using our state-of-the-art reference HMM phone recognizer. Codes CI,LM,I,D,S and PER respectively signify context independent, language model, number of insertions, number of deletions, number of substitutions and phone error rate.

6.7 Conclusion

We validated some kernel-based methods for their use on speech data, i.e `timit` data set and compared to the state-of-the-art. This leads to the following main conclusions:

- In case of phone classification on the TIMIT corpus, SVM as well as FS-MKLR and SCDP-FSLSSVM outperformed a state-of-the-art HMM classifier. Pairwise coupled one-versus-one encoded SVMs produced the most accurate results. FS-MKLR resulted into very sparse models compared to SVM while still having a competitive classification performance.
- Besides FS-MKLR also SCDP-FSLSSVM and SVM which only indirectly estimate a-posteriori probabilities seem to match the Bayes probability fairly well.
- In case of phone recognition the use of universum data to build the phone model improved the final phone recognition rate without increasing the number of parameters of the phone model.
- Without a language model our segment-based approach has comparable results to that of the frame-based state-of-the-art HMM recognizer using a 1-gram.
- However, when we do include language models our method does not have the same PER reduction as is the case for the HMM recognizer. Investigating this mismatch will be the subject of future research.

We believe there is room left for further improvement of the recognition rates: e.g. instead of using a frame-based segmentation model we could use a segment-based

model incorporating segmental features or features extracted at specific temporal landmarks (e.g. points of oral closure (or release), points of maximal constriction (or opening) in the vocal tract), the simple phone segment mapping to a fixed-length feature vector we employed, looses information due to the averaging operation instead we can use sequence kernels which operate directly on the variable length segments.

Chapter 7

Conclusion

In this chapter the most important results of this dissertation are reviewed and some general conclusions are drawn. Furthermore, some interesting further research paths for future work are described.

7.1 Concluding Remarks

The goal of this thesis was the development of new kernel-based methods and algorithms which allow integration into ASR. The main motivation to use this type of methods are the state-of-the-art results of kernel-based methods reported for many different applications ranging from bio-informatics to financial engineering. These techniques are capable of constructing non-linear decision functions of the examples. As the basis for algorithmic design, the primal-dual formulation originating from the theory of convex optimization in the context of LS-SVMs was used. This also provided insights to define computational feasible approximations to the original schemes.

For the application in a speech recognition system some specific requirements were first identified and later tackled throughout the thesis:

- **Scalability:** Probably the main drawback of using kernel-based methods in ASR is the computational complexity. We therefore presented two practical, and scalable kernel-based algorithms: fixed-size KLR and sparse conjugate direction pursuit based LS-SVM. They both use the primal-dual formulation

to find an approximate primal problem which is more advantageous to solve in case of large-scale problems. Additionally, the trade-off between model accuracy and training (and model) complexity are directly controlled by the user.

- **Multi-class classification:** Typical kernel-based approaches are developed for binary classification but in this thesis several multi-class strategies were compared. As a result it was observed that in terms of training speed a one-versus-one coupling approach is favored over a one-versus-all encoding. In terms of prediction accuracy both gave comparable results while one-versus-one was superior to one-versus-all in case the linear kernel was used. It was further seen that for SVM and sparse LS-SVM, pairwise coupling gives better prediction accuracies compared to the method of Price or voting. In case of one-versus-one coupling of fixed-size KLR models, pairwise coupling proved no further gain over the method of Price which is computationally less demanding. The all-at-once multi-class KLR approach is preferred compared to its binary coupled variants in terms of classification accuracy and model sparsity.

- **Sparseness:** The number of parameters sets the evaluation speed of the model. It is therefore desirable to introduce as much sparsity among the parameters as possible while still having good prediction accuracies. Unfortunately, we empirically found that a tuned SVM model still includes many support vectors and corresponding nonzero parameters. Additionally, the commonly utilized one-versus-one coding scheme additionally increases the global number of parameters. In contrast, the devised methods in this thesis produce significantly sparser models while having comparable classification accuracies on several public domain data sets. Moreover, in case of using the discussed fixed-size approximation on the all-at-once KLR formulation as described in , additional gain in sparsity opposed to one-versus-one approaches is empirically indicated.

- **Probabilistic interpretation:** All considered methods yield, either direct or indirect, probabilistic outcomes. Although theoretically less supported, indirect methods (which use an additional model to estimate probabilities based on the latent model values) give empirically adequate results in this context. They may even give comparable results to a direct approach utilized by KLR, as is seen in some experiments.

- **Variable-length sequence:** For handling sequences of variables lengths a simple and fast mapping to fixed-length vectors [36], based on extracting a

fixed number of features from a number of regions in the considered variable-length sequence was used.

As a showcase a segment-based speech recognition system was built which has the potential advantage to overcome some of the drawbacks known to exist in standard HMM based recognizers such as duration and trajectory modeling. These segments contain larger speech units as is the case in standard systems. This suggests that more features are needed to adequately describe such segment. At this point the theory of kernel-based learners fits nicely since such methods are known to result in accurate classifiers even in high dimensional spaces. Furthermore, it is proposed to include universum data into the training phase through a modified regularization scheme. In case a given segmentation is assumed, it is observed that the kernel-based methods clearly outperform the state-of-the-art HMM system. In case of an unknown segmentation, without a language model the devised segment-based approach supported by kernel methods has comparable results to that of the frame-based state-of-the-art HMM recognizer using a 1-gram. This is promising since there is room left for further improvement (see future work).

7.2 Future Research

Three interesting research paths for future work are:

- **Segmentation model:**

  In the segment-based phone recognition process (formally described in (6.2)), the main focus was on modeling the phone probability factor which performs the labeling process. Therefore less attention was spend on the segmentation probability factor. As already indicated in Chapter 6 instead of using the likelihood scores of a frame-synchronous speech recognizer to account for the segmentation model, one could construct the estimate of $\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}})$ as follows

  $$\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_s} \Pr(s_i|f(\underline{\mathrm{X}}_{\tau_{i-1}}^{\tau_i-1})), \tag{7.1}$$

  where s_i corresponds to a hypothesized speech unit (e.g. phone) and f denotes a function which extracts features from the frames belonging to s_i. These individual segment scores can be directly modeled using the considered kernel methods as binary classifiers which discriminate between correct and incorrect segments. Indeed, in this case a class of universum (badly segmented

units) data has to be provided. Question is whether or not this has a high influence on the recognition performance in case this (or a subset of the) universum data was already used in the training process of the phone model.

Another option would be to first train a kernel-based discriminant function, possibly using other types of features as suggested in e.g. [60], [76], to detect events occurring at the boundaries of segments. And then combine the outputs of the discriminant function on hypothesized boundary data together with additional features extracted from the within segment MFCC frames (as in [126]) to give a score for the considered segmentation

$$\Pr(\underline{\mathrm{S}}|\underline{\mathrm{X}}) \approx \prod_{i=1}^{N_s} \Pr(s_i|z_i), \tag{7.2}$$

where z_i is a fixed length vector of features for the i-th segment. See also [69] where the use of landmarks is successfully integrated in a conditional segment-based speech recognizer.

- **Design of a customized kernel:**

  In most kernel-based research, (static) kernels such as the linear, polynomial, RBF, etc., are used which are designed to work on fixed length vectors. However, kernel methods can be extended in a straightforward manner for the case one works on other objects by means of using customized kernels. To apply kernel methods, it is sufficient in practice to be able to evaluate the kernel scalar outcome for each pair of objects. If roughly speaking the kernel function ensures a symmetric positive semi-definite kernel (Mercer condition, see e.g. [172]) matrix then it can be plugged in into all existing kernel-based methods. In the literature we can find kernels for all kinds of objects such as: string kernels for symbol sequences (e.g. [112]), kernels for graphs for structured objects (e.g. [97]), rational kernels for automata (e.g. [37] [111]), sequence vectors for variable sequences of vectors (e.g. [161]). In [113] an overview of the different *object* kernels is given.

  The approach used in this thesis is to define a new feature space of interest. From each variable length segment we extract an equal number of features as is described in (Section 6.2). From this point we can work with the conventional static vector kernels. Since we have to define our new feature space beforehand, there is a potential risk of loosing information. Future research could concentrate on investigating the use of other kernel functions which score pairs of segments directly. Possible research directions are firstly to use kernels which are based on generative models such as the Fisher kernel [87] and secondly to use kernels with similarity scores based on dynamic

programming. Concerning the latter, which seems to match best with our current context and does not assume a model for the probability density of the features, in the literature examples can be found such as: kernels using the Dynamic Time Warping (DTW) [149] score (e.g. [161] [8]), the Smith Waterman algorithm (e.g. [186]) and the edit distance (e.g. [37]). Although difficulties arise when translating such distances into semi-positive definite kernels, in [38] encouraging results on a speech recognition task are reported.

Note that the computational cost related to these kind of *object* kernels is typically higher than those of the static vector kernels.

- **SCDP-FSMKLR:**

  Recently there was a lot of interest in using the L_1-penalty in the context of logistic regression see e.g. [154], [65], or [160]. This type of regularization adds sparsity to the solution. As the approach discussed in Chapter 5 is very related it might give new insights in the development of training algorithms to obtain sparse approximate (kernel) logistic regression models.

  A concern might be the iterative scheme employed to solve the logistic regression problem. Recall that the logistic regression objective can be restated as an IRLS problem. This means that instead of one, a set of linear systems has to be solved. It is to be investigated in which way the sparsity patterns in different subsequent iterations can be matched such that combined they give a satisfactory sparse approximate solution to the logistic regression problem.

Appendix A

Primal-dual formulation of MKLR

These appendices discuss in more detail the primal-dual formulation for Multi-class Kernel Logistic Regression (MKLR).

A.0.1 Gradient and Hessian

In this appendix a primal-dual formulation of the MKLR algorithm is stated. As will be shown next, MKLR can be solved in the Least Squares Support Vector Machines (LS-SVM) framework. This is achieved by first translating the minimization of objective function (4.6) to an iteratively re-weighted least-squares (IRWLS) equivalent. Then each locally weighted least-squares problem can be kernelized and phrased as a weighted LS-SVM problem. Hence, one can define an iterative scheme of re-weighted LS-SVMs which minimizes the MKLR objective.

By defining $A \in \mathbb{R}^{CN \times C(D+1)}$ as

$$A = \left(\begin{array}{c|c|c|c} A_1 & A_2 & \dots & A_N \end{array} \right)^T, \tag{A.1}$$

where $A_i \in \mathbb{R}^{C \times C(D+1)}$ and

$$A_i = \begin{pmatrix} x_i^T & 0 & & 0 \\ 0 & x_i^T & & 0 \\ & & \ddots & \\ 0 & 0 & & x_i^T \end{pmatrix}. \tag{A.2}$$

we can write the formulas of the gradient and Hessian of the objective function (4.6) a matrix notation which is convenient to reformulate to an IRWLS problem.

In iteration k this gives a gradient defined as

$$g^{(k)} = A^T(\tilde{p}^{(k)} - \tilde{r}) + \nu w^{(k-1)}, \tag{A.3}$$

where notation $\tilde{\cdot}$ denotes a certain component ordering as is explained in the introduction, and

$$\tilde{r}_{ci} = \mathcal{I}(y_i = c), \tag{A.4}$$

with $\mathcal{I}$ the indicator function. The k-th Hessian is given by

$$H^{(k)} = A^T W^{(k)} A + \nu I, \tag{A.5}$$

where $W^{(k)}$ is a block-diagonal weight matrix

$$W^{(k)} = \text{blockdiag}(W_1(k), ..., W_N(k)), \tag{A.6}$$

with the i-th block of a block-diagonal weight matrix $W^{(k)}$ defined as

$$W_i^{(k)} = \begin{pmatrix} \tilde{p}_{1i}^{(k)}(1-\tilde{p}_{2i}^{(k)}) & -\tilde{p}_{1i}^{(k)}\tilde{p}_{2i}^{(k)} & \cdots & -\tilde{p}_{1i}^{(k)}\tilde{p}_{Ci}^{(k)} \\ -\tilde{p}_{2i}^{(k)}\tilde{p}_{1i}^{(k)} & \tilde{p}_{2i}^{(k)}(1-\tilde{p}_{2i}^{(k)}) & & -\tilde{p}_{2i}^{(k)}\tilde{p}_{Ci}^{(k)} \\ & & \ddots & \\ -\tilde{p}_{Ci}^{(k)}\tilde{p}_{1i}^{(k)} & -\tilde{p}_{Ci}^{(k)}\tilde{p}_{2i}^{(k)} & \cdots & \tilde{p}_{Ci}^{(k)}(1-\tilde{p}_{Ci}^{(k)}) \end{pmatrix}. \tag{A.7}$$

Note that this double indexing of a vector is explained Section 4.2.1.

Remark that for each matrix column j we have $\sum_{i=1}^{N} W_{ij}^{(k)} = 1$ which makes the term $A^T W^{(k)} A$ positive semi-definite. Adding a positive constant ν to the diagonal results in a positive definite Hessian.

A.0.2 Iteratively re-weighted least squares

Newton's method computes in each iteration k the optimal step $s^{(k)opt}$ using the second-order Taylor expansion of the objective function ℓ_{LR}^{ν}. This results in the following local optimization step

$$\begin{aligned} s^{(k)opt} = \arg\min_{s^{(k)}} \; & \ell_{LR}(w^{(k-1)}) + (A^T(\tilde{p}^{(k)} - \tilde{r}) + \nu w^{(k-1)})^T s^{(k)} \\ & + \frac{1}{2} s^{(k)T}(A^T W^{(k)} A + \nu I) s^{(k)}. \end{aligned} \tag{A.8}$$

By exchanging terms we can proof that (A.8) can be expressed as an iteratively re-weighted least squares problem (IRWLS) which can be written as

$$\min_{s^{(k)}} \frac{1}{2}||As^{(k)} - \tilde{q}^{(k)}||^2_{W^{(k)}} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}), \tag{A.9}$$

where $\tilde{p}^{(k)}, A, W^{(k)}$ are respectively defined as in (4.7),(A.1),(A.6), and

$$\tilde{q}^{(k)} = (W^{(k)})^{-1}(\tilde{r} - \tilde{p}^{(k)}). \tag{A.10}$$

It turns out that the global regularization term in (4.6) is reflected in each step as a usual regularization term, resulting in a robust algorithm when ν is chosen appropriately.

A.0.3 Iteratively re-weighted least-squares support vector machines

Starting from equation (A.9) we include a feature map and introduce the error variable e, this results in

$$\min_{s^{(k)}, e^{(k)}} \frac{1}{2}{e^{(k)}}^T W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)})$$

$$\text{such that } \tilde{q}^{(k)} = Fs^{(k)} + e^{(k)}, \tag{A.11}$$

where $F \in \mathbb{R}^{CN \times C(D_\varphi + 1)}$ is defined as

$$F = \left(\begin{array}{c|c|c|c} F_1 & F_2 & \ldots & F_N \end{array} \right)^T, \tag{A.12}$$

where

$$F_i = \begin{pmatrix} \varphi(x_i)^T & 0 & & 0 \\ 0 & \varphi(x_i)^T & & 0 \\ & & \ddots & \\ 0 & 0 & & \varphi(x_i)^T \end{pmatrix}. \tag{A.13}$$

In the context of LS-SVMs this is called the primal problem. To conveniently write down a dual formulation of this problem we use the following notation for the multi-class kernel matrix $G = FF^T$.

The Lagrangian of the constrained problem as stated in (A.11) becomes

$$\begin{aligned} \mathcal{L}(s^{(k)}, e^{(k)}; \tilde{\alpha}^{(k)}) =& \frac{1}{2}{e^{(k)}}^T W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}) \\ & - \tilde{\alpha}^{(k)T}(Fs^{(k)} + e^{(k)} - \tilde{q}^{(k)}) \end{aligned}$$

with Lagrange multipliers $\tilde{\alpha}^{(k)} \in \mathbb{R}^{NC}$. The first order conditions for optimality are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial s^{(k)}} = 0 \rightarrow s^{(k)} = \frac{1}{\nu} F^T \tilde{\alpha}^{(k)} - w^{(k-1)} \\ \frac{\partial \mathcal{L}}{\partial e^{(k)}} = 0 \rightarrow \tilde{\alpha}^{(k)} = W^{(k)} e^{(k)} \\ \frac{\partial \mathcal{L}}{\partial \tilde{\alpha}^{(k)}} = 0 \rightarrow F s^{(k)} + e^{(k)} = \tilde{q}^{(k)}. \end{cases} \tag{A.14}$$

This results in the following dual solution at iteration k

$$\left(\frac{1}{\nu} G + (W^{(k)})^{-1} \right) \tilde{\alpha}^{(k)} = \tilde{q}^{(k)} + F w^{(k-1)}. \tag{A.15}$$

The solution $w^{(k')}$ can be expressed in terms of $\tilde{\alpha}^{(k')}$ computed in the last iteration. Given the formula for $s^{(k)}$ in (A.14) and the update rule $w^{(k)} = w^{(k-1)} + s^{(k)}$ employed in the IRWLS scheme, one can write

$$w^{(k')} = \frac{1}{\nu} F^T \tilde{\alpha}^{(k')}. \tag{A.16}$$

The linear system in (A.15) can be solved in each iteration by substituting $w^{(k-1)}$ with $\frac{1}{\nu} F^T \tilde{\alpha}^{(k-1)}$. Hence, $\Pr(\mathbf{Y} = y_* | \mathbf{X} = x^*; w)$ can be predicted by using (4.1) where $f_c(x^*) = w_{c.}^T \varphi(x^*) = \frac{1}{\nu} \sum_{i=1}^{N} \tilde{\alpha}_{ci} K(x_i, x^*)$.

Appendix B

Benchmark data sets

In this appendix, some details of the benchmark data sets on which experiments were carried out, are described. The first part describes the data sets which were solely used in the context of classification. The second part explains some details about the `timit` *benchmark data set and the method to extract features which were used in our phone classification and recognition experiments.*

B.1 Data sets only for classification experiments

Throughout this thesis several publicly available data sets were used. The characteristics for each of the data sets is given in Table B.1. The data set `dna` and the handwritten digits data set `usps` [86] are available at `http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/data`. The remote sensing task `satimage` from the statlog repository is available at `http://www.niaad.liacc.up.pt/old/statlog`. The spoken English alphabet letters data set `isolet` [179] is available at `http://archive.ics.uci.edu/ml`. The Peterson and Barney English vowel data set `pb` [141] is available at `ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/data`, the `adult` and `gamma telescope` data sets can be downloaded from `http://kdd.ics.uci.edu`, and the `ripley` data set can be found as the synthetic two-class problem at `http://www.stats.ox.ac.uk/pub/PRNN/`. Apart from those data sets, we also used modified versions of UCI data sets available from `http://ida.first.fraunhofer.de/~raetsch`. These data sets have a fixed number of partitions for each data set which makes comparison with other references a lot easier. Note that for `adult`, `gamma telescope` no fixed test set was given.

data	N_{tr}	N_{te}	N_{va}	C	D
`adult`	30,148	15,074	0	2	14
`banana`	400	4,900	0	2	2
`breast-cancer`	200	77	0	2	9
`diabetis`	468	300	0	2	8
`dna`	1,400	1,186	600	3	180
`flare-solar`	666	400	0	2	9
`gamma telescope`	12,680	6,340	0	2	10
`german`	700	300	0	2	20
`heart`	170	100	0	2	13
`image`	1,300	1,010	0	2	18
`isolet`	6,240	1,560	0	26	617
`pb`	760	760	0	10	4
`ringnorm`	400	7,000	0	2	20
`ripley`	250	1,000	0	2	2
`satimage`	4,435	2,000	0	6	36
`splice`	1,000	2,175	0	2	60
`thyroid`	140	75	0	2	5
`titanic`	150	2,051	0	2	3
`twonorm`	400	7,000	0	2	20
`usps`	7,291	2,007	0	10	256
`waveform`	400	4,600	0	2	21

Table B.1: Characteristics of the public domain data sets used in this thesis. Where N_{tr} is the number of training data, N_{te} the number of test data, N_{va} the number of validation data, C the number of classes and D the number of dimensions (features).

During our experiments we chose to randomly divide all available data into a training and testing partition at a ratio of respectively 2/3 and 1/3 for several runs.

B.2 `timit` benchmark data set

The `timit` corpus (the name comes from Texas Instruments, where the recordings took place, and MIT, where the transcription and segmentation were performed), is

aimed at low-level acoustic phonetic studies. The material we used [1] (which excludes the two fixed sentences that every speaker read, meant for dialect detection), consists of 450 "phonetically compact" sentences designed at MIT to provide a good coverage of pairs of phones, and 1,890 "phonetically diverse" sentences to add diversity in sentence types and phonetic contexts. The phonetically compact sentences were each read 7 times by different speakers, and the phonetically diverse sentences only once. The training material consists of 3,696 sentences, 2,295 of which are repetitions of 331 sentences, and 1,401 are only said once. In total, the training material is read by 462 different speakers. The test material likewise consists of sentences that were repeated multiple times (836 utterances of the same 122 sentences) and sentences that were spoken only once (508 utterances), giving a total 1,344 utterances. The test material was read by another 168 speakers. No sentences occurred both in training and test sets. The test data has a "core" portion containing 24 speakers (2 male and 1 female from each dialect region), which together produced 192 (unique) sentences. However, our test scores are given for the full test set.

What makes the `timit` corpus rather unique, is that the complete material has been hand labeled and -segmented. This, together with its unusually large number of different speakers, makes it the database of choice for phone classification experiments. The original hand-made phonetic labeling used a phone set of 64 labels, but common practice maps those 64 to a smaller number (typically around 50) for HMM modeling and finally to 39 phones (including silence) for evaluation purposes [120]. For kernel-based modeling we immediately start from the 39 class setting.

Features were always extracted by the preprocessing procedure schematically shown in Fig. B.1. In the following, we list some extra details that are not explicitly shown in the figure:

- All data was recorded at a sample rate of 16 kHz.
- The short time Fourier transform uses Hamming windows.
- The first derivatives are calculated on the basis of 5 frames around the current frame, the second derivatives on the basis of 7 frames.
- A window shift of 10 ms was used.
- The MIDA algorithm [41] selects a 36-dimensional subspace of the 72-dimensional data space given by the 24 Mel-scale filterbank outputs (obtained

[1] Denoted by 'sx' and 'si' prefixes in the nomenclature of the data files.

using a window size of 25ms and shift of 10ms) and their first and second derivatives.

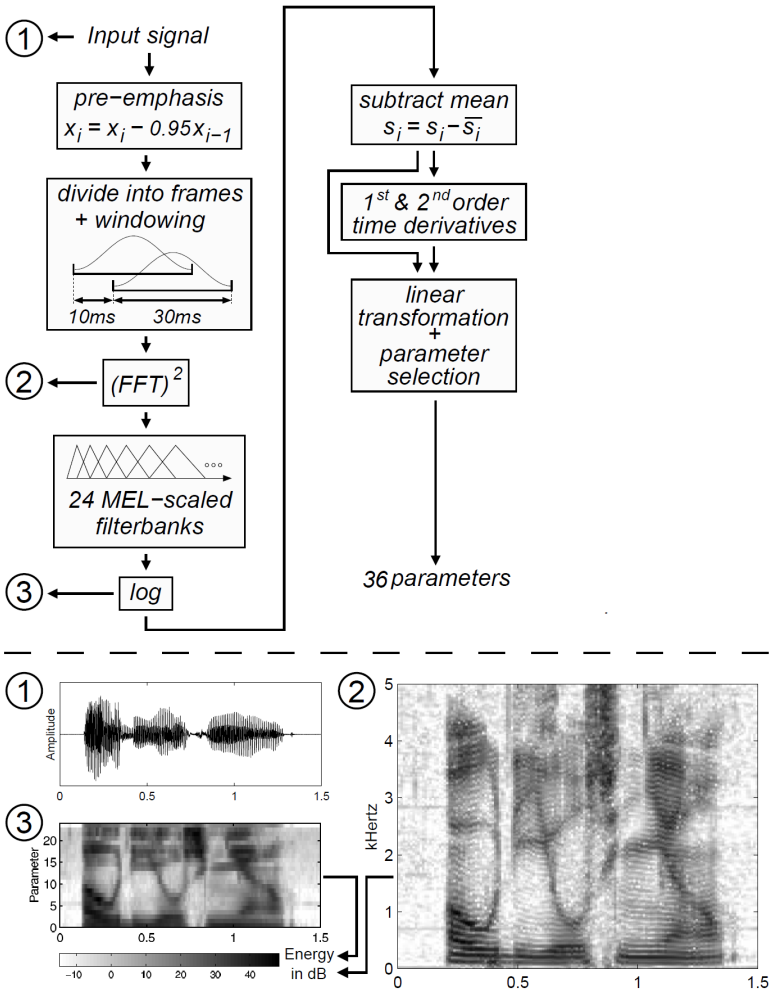


Figure B.1: Default preprocessing setup used for the phone experiments in this dissertation. Adapted from [42] with permission of the author.

Bibliography

[1] D.M. Allan, "The relationship between variable slection and prediction," *Technometrics*, Vol. 16, pp. 125-127, 1974.

[2] C. Alzate, M. Espinoza, B. De Moor, J.A.K. Suykens, "Identifying Customer Profiles in Power Load Time Series Using Spectral Clustering," *in Proc. of the 19th International Conference on Artificial Neural Networks (ICANN 2009)*, Limassol, Cyprus, pp. 315-324, 2009.

[3] J.A. Anderson, V. Blair, "Penalized maximum likelihood estimation in logistic regression and discrimination," *Biometrika*, Vol. 1, pp. 123-136, 1982.

[4] S. Austin, G. Zavaliagkos, J. Makhoul, R. Schwartz, "Speech recognition using segmental neural nets," *In Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, U.S.A., Vol. 1, pp. 625-628, 1992.

[5] M. Aizerman, E. Braverman, L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition," *Automation and Remote Control*, Vol. 25, pp. 821-837, 1964.

[6] L.L. Bahl, F. Jelinek, R.L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5(2), pp. 179-190, 1983.

[7] L.R. Bahl, P.F. Brown, P.V. de Souza, R.L. Mercer, "Speech recognition with continuous-parameter hidden Markov models," *Computer Speech and Language*, Vol. 2(3-4), pp. 219-234, 1987.

[8] C. Bahlmann, B. Haasdonk, and H. Burkhardt, "On-line handwriting recognition with support vector machines - a kernel approach," *In Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 49-54, 2002.

[9] K.P. Bennett, M. Momma, M.J. Embrechts, "MARK: A boosting algorithm for heterogeneous kernel models," *In Proceedings of SIGKDD*, Edmonton, Alberta, Canada, pp. 24-31, 2002.

[10] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra, "A maximum entropy approach to natural language processing", *Computational Linguistics*, Vol. 22(1), pp. 39-71, 1996.

[11] Ø. Birkenes, T. Matsui, K. Tanabe, T.A. Myrvoll, "N-best rescoring for speech recognition using penalized logistic regression machines with garbage class," *In Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, Honolulu, Hawaii, USA, Vol. 4, pp. IV-449-IV-452, 2007.

[12] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[13] T. Blumensath, M.E. Davies, "Gradient Pursuits," *IEEE Transactions on Signal Processing*, Vol. 56 (6), pp. 2370-2382, 2008.

[14] H.A. Bourlard, N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1994.

[15] O. Bousquet, S. Boucheron, G. Lugosi, "Introduction to statistical learning theory," *in Advanced Lectures on Machine Learning Lecture Notes in Artificial Intelligence*, Springer, 2004.

[16] S. Boyd, L. Vandenberghe, *Convex optimitization*, Cambridge University Press, 2002. Available online at `http://www.stanford.edu/~boyd/cvxbook.html`.

[17] P. Bühlmann, B. Yu, "Boosting with the L2-loss: Regression and Classification," *Journal of the American Statistical Association*, Vol. 98, pp. 324-339, 2003.

[18] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, B. De Moor, "Optimized Fixed-Size Kernel Models for Large Data Sets," *Computational Statistics & Data Analysis*, Vol. 54(6), pp. 1484-1504, 2010.

[19] C.J.C. Burges, B. Schölkopf, "Improving the accuracy and speed of support vector learning machines," *In Proceedings of the 9th NIPS Conference*, pp. 375-381, 1997.

[20] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, Vol. 2(2), pp. 121-167, 1998.

[21] P. Burman, "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods," *Biometrika*, Vol. 77, no. 3, pp. 503-514, 1989.

[22] A.M. Bruckstein, D.L. Donoho, M. Elad, "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images," *SIAM Review*, Vol. 51(1), pp. 34-81, 2009.

[23] S. Chakrabartty, G. Cauwenberghs, "Forward Decoding Kernel Machines: A Hybrid HMM/SVM Approach to Sequence Recognition," *Proc. of the First International Workshop on Pattern Recognition with Support Vector Machines*, London, UK, pp. 278-292, 2002.

[24] E. Candès, "Compressive sampling," *in proceedings of International Congress of Mathematics*, 2006.

[25] G.C. Cawley, N.L.C. Talbot, "A greedy training algorithm for sparse least-squares support vector machines," *In International Conference on Artificial Neural Networks*, Madrid, Spain, Vol. 2415, pp. 681-686, 2002.

[26] G.C. Cawley, "Leave-One-Out Cross-Validation Based Model Selection Criteria for Weighted LS-SVMs," *In Proc. of the International Joint Conference on Neural Networks (IJCNN-2006)*, Vancouver, Canada, pp. 1661-1668, 2006.

[27] G.C. Cawley, N.L.C. Talbot, M. Girolami, "Sparse Multinomial Logistic Regression via Bayesian L1 Regularisation," *In Advances in Neural Information Processing Systems*, Vol. 19, pp. 209-216, 2007.

[28] G.C. Cawley, N.L.C. Talbot, "Efficient approximate leave-one-out cross-validation for kernel logistic regression," *Machine Learning*, Vol. 71, pp. 243-264, 2008.

[29] N.V. Chawla, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, "Smote: Synthetic minority oversampling technique," *Journal Of Artificial Intelligence Research*, Vol. 16, pp. 321-357, 2002.

[30] C. Olivier, "Training a Support Vector Machine in the Primal," *Neural Computation*, Vol. 19(5), pp. 1155-1178, 2007.

[31] S. Chen, S. A. Billings and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of Control*, Vol. 50(5), pp. 1873-1896, 1989.

[32] S.S. Chen, D.L. Donoho, M.A. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing*, Vol. 20(1), pp. 33-61, 1999.

[33] S. Chen, X. Hong, B.L. Luk, C.J. Harris, "Orthogonal-least-squares regression: A unified approach for data modelling," *Neurocomputing*, Vol. 72 (10-12), pp. 2670-2681, 2009.

[34] S. Chen, C. Zhang, "Selecting informative universum sample for semi-supervised learning," *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, Pasadena, California, USA, pp. 1016-1021, 2009.

[35] W. Chu, S.S. Keerthi, C. Ong, "An improved conjugate gradient scheme to the solution of least squares SVM," *IEEE Transactions on Neural Networks*, Vol. 16(2), pp. 498-501, 2005.

[36] P. Clarkson, P.J. Moreno, "On the use of support vector machines for phonetic classification," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, Arizona, U.S.A, Vol. 2 , pp. 585-588, 1999.

[37] C. Cortes, P. Haffner, M. Mohri, "Rational kernels : Theory and algorithms," *Journal of Machine Learning Research*, Vol. 5, pp. 1035-1062, 2004.

[38] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," *In IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, U.S.A., pp. 413-416, 2007.

[39] I. Daubechies, R. DeVore, M. Fornasier, S. Gunturk, "Iteratively Re-weighted Least Squares minimization: Proof of faster than linear rate for sparse recovery," *Information Sciences and Systems*, Princeton, NJ, pp. 26-29, 2008.

[40] A.P. Dempster, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, Vol. 39, pp. 1-38, 1977.

[41] K. Demuynck, J. Duchateau, D. Van Compernolle, and P. Wambacq, "An efficient search space representation for large vocabulary continuous speech recognition," *Speech Communication*, Vol. 30(1), pp. 37-53, 2000.

[42] K. Demuynck, *Extracting, Modelling and Combining Information in Speech Recognition*, PhD thesis, Faculty of Engineering, K.U.Leuven, ESAT-PSI, Belgium, 2001.

[43] K. Demuynck, D. Van Compernolle, and P. Wambacq, "Doing away with the viterbi approximation," *In Proc. International Conference on Acoustics, Speech and Signal Processing*, Orlando, U.S.A., Vol. 1, pp. 717-720, 2002.

[44] M. De Wachter, *Example based continuous speech recognition*, PhD thesis, Faculty of Engineering, K.U.Leuven, ESAT-PSI, Belgium, 2007.

[45] J. Duchateau, "HMM Based Acoustic Modelling in Large Vocabulary Speech Recognition," *PhD thesis*, K.U.Leuven, ESAT, 1998.

[46] J. Duchateau, K. Demuynck, and D. Van Compernolle, "Fast and accurate acoustic modelling with semi-continuous HMMs," *Speech Communication*, Vol. 24(1), pp. 5-17, 1998.

[47] D.L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52(4), pp. 1289-1306, 2006.

[48] D.L. Donoho, Y. Tsaig, I. Drori, J. Starck, "Sparse Solution of Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit," *Technical report*, Stanford University, 2006.

[49] D.L. Donoho, Y. Tsaig, "Fast Solution of L1-norm Minimization Problems When the Solution May be Sparse," *Technical report*, Stanford University, 2006.

[50] T. Downs, K.E. Gates, A. Masters, "Exact simplification of support vector solutions," *Journal of Machine Learning Research*, Vol. 2, pp. 293-297, 2001.

[51] K. Duan, S.S Keerthi, "Which Is the Best Multiclass SVM Method? An Empirical Study," *In Proc. of the Sixth International Workshop on Multiple Classifier Systems*, Seaside, Oregon, U.S.A., pp. 278-285, 2005.

[52] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification (2nd ed.)*, John Wiley and Sons, 2001.

[53] R.O. Duda, P.E. Hart, *Pattern Recognition and Scene Analysis*, Wiley, 1973.

[54] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, "Least Angle regression," *Annals of Statistics (with discussion)*, Vol. 32(2), pp. 407-499, 2004.

[55] P.P.B. Eggermont, V.N. LaRiccia, *Maximum Penalized Likelihood Estimation. Volume II: Regression*, Springer, 2001.

[56] M. Espinoza, J.A.K. Suykens, R. Belmans, B. De Moor, "Electric Load Forecasting - Using kernel based modeling for nonlinear system identification," *IEEE Control Systems Magazine, Special Issue on Applications of System Identification*, vol. 27(5), pp. 43-57, 2007.

[57] R.-E. Fan, P.-H. Chen, C.-J. Lin, "Working set selection using second order information for training SVM," *Journal of Machine Learning Research*, Vol. 6, pp. 1889-1918, 2005.

[58] T. Feder, D. Greene, "Optimal algorithms for approximate clustering". *In Proc. 20th ACM Symposium Theory of Computing*, Chicago, Illinois, U.S.A, pp. 434-444, 1988.

[59] M.A.T. Figueiredo, R.D. Nowak, S.J. Wright, "Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 1(4), pp. 586-597, 2007.

[60] G. Flammia, P. Dalsgaard, O. Andersen, B. Lindberg, "Segment based variable frame rate speech analysis and recogntiont using a spectral variation function," *In Proc. ICSLP*, pp. 983-986, 1992.

[61] E. Frank, S. Kramer, "Ensembles of nested dichotomies for multi-class problems," *In Proc. of the twenty-first international conference on Machine learning*, Banff, Alberta, Canada, Vol. 69, pp. 305-312, 2004.

[62] J.H. Friedman, "Another approach to polychotomous classification," *technical report*, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, 1996.

[63] A. Ganapathiraju, *Support Vector Machines For Speech Recognition*, PhD thesis, Mississippi State University, 2001.

[64] A. Ganapathiraju, J. Hamaker, and J. Picone, "Applications of support vector machines to speech recognition," *IEEE Trans. Signal Processing*, Vol. 52(8), pp. 2348-2355, 2004.

[65] A. Genkin, D. Lewis, D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, Vol. 49, pp. 291-304, 2007.

[66] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, Vol. 4(1), pp. 1-58, 1992.

[67] M. Girolami, "Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem," *Neural Computation*, Vol. 14(3), pp. 669-688, 2003.

[68] J.R. Glass, "A probabilistic framework for feature-based speech recognition," In Proc. of ICSLP, Philadelphia, Pennsylvania, U.S.A., pp. 2277-2280, 1996.

[69] J.R. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, Vol. 17 (2-3), pp. 137-152, 2003.

[70] G.H. Golub, C.F. Van Loan, *Matrix Computations*, Baltimore MD: Johns Hopkins University Press, 3th edition, 1996.

[71] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, Vol. 38, pp. 293-306, 1985.

[72] R. Gribonval, P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," *IEEE Transactions on Information Theory*, Vol. 52(1), pp. 255-261, 2006.

[73] I. Guyon, J. Weston, S. Barnhill, N. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, Vol. 46(1-3), pp. 389-422, 2002.

[74] I. Guyon, A. Saffari, G. Dror, G. and Cawley, "Analysis of the IJCNN 2007 agnostic learning versus prior knowledge challenge, *Neural Networks*, Vol. 21(2-3), pp. 544-550, 2008.

[75] B. Hamers, *Kernel Models for Large Scale Applications*, PhD thesis, Faculty of Engineering, K.U.Leuven, Leuven, 2004.

[76] M. Hasegawa-Johnson, J. Baker, S. Borys, K. Chen, E. Coogan, S. Greenberg, A. Juneja, K. Kirchhoff, K. Livescu, S. Mohan, J. Muller, K. Sommez, T. Wong, "Landmark-based speech recognition: Report of the 2004 Johns Hopkins summer workshop," *in International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, USA, pp. 213-216, 2005.

[77] T. Hastie, R. Tibshirani, "Classification by pairwise coupling," *Advances in Neural Information Processing Systems*, Vol. 10, 1998.

[78] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2001.

[79] X. He, L. Deng, *Discriminative Learning for Speech Recognition: Theory and Practice*, Morgan & Claypool Publishers, 2008.

[80] D.W. Hosmer, S. Lemeshow, *Applied Logistic Regression*, Wiley, 1989.

[81] C.W. Hsu, C.J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks*, Vol. 13, pp. 415-425, 2002.

[82] X. Huang, M.A. Jack, "Semi-continuous hidden Markov models for speech signals," *Computer Speech and Language*, Vol. 3(3), pp. 239-252, 1989.

[83] X. Huang, A. Acero, H.W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, 2001.

[84] R. Hérault, Y. Grandvalet, "Sparse Probabilistic Classifiers," *In Proc. of the 24-th International Conference on Machine Learning*, Corvallis, Oregon, U.S.A., pp. 337-344, 2007.

[85] M.R. Hestenes, E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *J. Res. Natl. Bur. Stand.*, vol. 49, 409-436 (1952).

[86] J.J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16(5), pp. 550-554, 1994.

[87] T. Jaakkola, D. Haussler, "Exploiting generative models in disciminative classifiers," *Advances in Neural Information Processing Systems*, Vol. 11, 1998.

[88] T. Jaakkola, D. Haussler, "Probabilistic kernel regression models," *In Seventh International Workshop on Artificial Intelligence and Statistics*," Fort Lauderdale, Florida, U.S.A., 1999.

[89] F. Jelinek *Statistical Methods for Speech Recognition*, The MIT Press, 1998.

[90] L. Jiao, L. Bo, L. Wang, "Fast Sparse Approximation for Least Squares Support Vector Machine," *IEEE Transactions on Neural Networks*, Vol. 18(3), pp. 685-697, 2007.

[91] B.-H. Juang, W. Chou, C.-H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 5(3), pp. 257-265, 1997.

[92] J.C. Junqua, "The Lombard reflex and its role on human listeners and automatic speech recognizers," *Journal of Acoustical Society of America* , Vol. 93(1), pp. 510-24, 2003.

[93] P. Karsmakers, K. Pelckmans, J. A. K. Suykens, "Multi-class kernel logistic regression: a fixed-size implementation," *In Proc. of the international joint conference in neural networks (IJCNN)*, Orlando, Florida, U.S.A, pp. 1756-1761, 2007.

[94] P. Karsmakers, K. Pelckmans, J. A. K. Suykens, H. Van hamme, "Fixed-Size Kernel Logistic Regression for Phoneme Classification," *In Proc. of INTERSPEECH*, Antwerpen, Belgium, pp. 78-81, 2007.

[95] P. Karsmakers, K. Pelckmans, H. Van hamme, J.A.K Suykens, "Large-Scale Kernel Logistic Regression for Segment-Based Phoneme Recognition", *Internal Report 09-174*, ESAT-SISTA, K.U.Leuven, Leuven, 2009, submitted for publication.

[96] P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van hamme H., J.A.K Suykens, "Sparse Conjugate Directions Pursuit with Application to Fixed-size Kernel Models", *Internal Report 10-63*, ESAT-SISTA, K.U.Leuven, Leuven, 2010, submitted for publication.

[97] H. Kashima, K. Tsuda, A. Inokuchi, *Kernel Methods in Computational Biology, volume 7, chapter Kernels for Graphs*, MIT Press, pp. 155Ű-170, 2004.

[98] M. Katz, M. Schafföner, E. Andelic, S. Krüger, A. Wendemuth "Sparse Kernel Logistic Regression for Phoneme Classification," *In Proc. of 10th International Conference on Speech and Computer (SPECOM)*, Patras, Greece, Vol. 2, pp. 523-526, 2005.

[99] S.S. Keerthi, S.K. Shevade, "SMO algorithnm for least-squares SVM formulations," *Neural Computation*, Vol. 15(2), pp. 487-507, 2003.

[100] S.S. Keerthi, K.B. Duan, S.K. Shevade, A.N. Poo, "A Fast Dual Algorithm for Kernel Logistic Regression," *Machine Learning*, Vol. 61, pp.151-165, 2005.

[101] S.S. Keerthi, O. Chapelle, D. Decoste, P. Bennett, E. Parrado-hernández, "Building Support Vector Machines with Reduced Classifier Complexity," *Journal of Machine Learning Research*, Vol. 7, pp. 1493-1515, 2006.

[102] J. Keshet, S. Bengio (Eds.), *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, John Wiley & Sons, United Kingdom, 2009.

[103] Y. Kim, S. Kwon, S.H. Song, "Multiclass sparse logistic regression for classification of multiple cancer types using gene expression data," *Computational Statistics & Data Analysis*, Elsevier, 2006.

[104] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, "An Interior-Point Method for Large-Scale l1-Regularized Least Squares," *IEEE Journal on Selected Topics in Signal Processing*, Vol. 1(4), pp. 606-617, 2007.

[105] G. Kimeldorf, G. Wahba, "Some results on Tchebycheffian spline functions," *Journal of Mathematics Analysis and Applications*, Vol. 33, pp. 82-95, 1971.

[106] K. Koh, S. Kim, and S. Boyd, "An interior-point method for large-scale l1-regularized logistic regression," *Journal of Machine Learning Research*, Vol. 8, pp. 1519-1555, 2007.

[107] P. Komarek, A. Moore, "Making Logistic Regression A Core Data Mining Tool With TR-IRLS," *In Proc. of the 5th International Conference on Data Mining*, Houston, Texas, U.S.A., pp. 685-688, 2005.

[108] B.J. de Kruif, T.J.A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Transactions on Neural Networks*, Vol. 14 (3), pp. 696-702, 2003.

[109] N.D. Lawrence, M. Seeger, R. Herbrich, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," *Advances in Neural Information Processing Systems*, Vol. 15, pp. 609-616, 2003.

[110] F. Lauer, Y.S. Ching, G. Blocha, A trainable feature extractor for handwritten digit recognition *Pattern Recognition*, Vol. 40(6), pp. 1816-1824, 2007.

[111] M. Layton, M. Gales, "Acoustic Modelling Using Continuous Rational Kernels," *Journal of VLSI Signal Processing Systems*, Vol. 48(1-2), pp. 67-82, 2007.

[112] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, Vol. 2, pp. 419-444, 2002.

[113] J. Louradour, "Noyaux de séquences pour la vérification du locuteur par Machines à Vecteurs de Support," Phd thesis, Université Paul Sabatier, Toulouse, France, 2007.

[114] H.C. Leung, I.L. Hetherington, V.W. Zue, "Speech recognition using stochastic segment neural networks," *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, U.S.A, Vol. 1, pp. 613-616, 1992.

[115] K.-F. Lee, "*Automatic Speech Recognition - The Development of the SPHINX System*," Kluwer Academic Publishers, 1989.

[116] Y.J. Lee and O.L. Mangasarian, "RSVM: Reduced support vector machines," *In Proceedings of the SIAM International Conference on Data Mining*, Philadelphia, 2001.

[117] C.C. Chang, C.J. Lin, "LIBSVM : a library for support vector machines," *Software available at* `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[118] H.-T. Lin, C.-J.Lin, R.-C. Weng, A note on Platt's probabilistic outputs for support vector machines, *Machine learning*, Vol. 68(3), pp. 267-276, 2007.

[119] C.-J. Lin, R.C. Weng, S.S. Keerthi, Trust Region Newton Method for Large-Scale Logistic Regression, *Journal of Machine Learning Research*, Vol. 9, pp. 623-646, 2008.

[120] K.F. Lee and H.W. Hon, "Speaker-independent phone recognition using hidden Markov models," *In Proc. IEEE Transactions on Acoustics, Speech, and Signal Processing*, New York, New York, U.S.A., Vol. 37, no. 11, pp. 1641-1648, 1988.

[121] R.W. Lutz and P. Bühlmann, "Conjugate direction boosting," *Journal of Computational and Graphical Statistics*, Vol. 15(2), pp. 287-311, 2006.

[122] D.J.C. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, Vol. 4, pp. 698-714, 1992.

[123] D.J.C. MacKay, "Bayesian methods for backpropagation networks," *Models of Neural Networks III*, pp. 211-254, 1994.

[124] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE*, Vol. 73(11), pp. 1551-1588, 1985.

[125] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic press, 1999.

[126] J.-P Martens, "A connectionist approach to continuous speech recognition," *In Proc. FORWISS/CRIM ESPRIT workshop*," Munich, Germany, pp. 26-33, 1994.

[127] P. McCullagh, J.A. Nelder, *Generalized Linear Models, 2nd edition*, Chapman and Hall, London, 1989.

[128] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. Smola, and K.-R. Müller, "Invariant feature extraction and classification in feature spaces," *Advances in Neural Information Processing Systems*, Vol. 12, pp. 526-532, 2000.

[129] M.F. Möller, "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks*, Vol. 6, pp. 525-533, 1993.

[130] A. Nadas, R.L. Mercer, L.R. Bahl, R. Bakis, P.S. Cohen, A.G. Cole, F. Jelinek, B.L. Lewis, "Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering," *In Proc. International Conference on Acoustics, Speech and Signal Processing*, Atlanta, Georgia, pp. 1153-1155, 1981.

[131] J. A. Nelder, R. Mead, "A simplex method for function minimization," *Computer Journal*, Vol. 7, pp. 308-313, 1965.

[132] J. Nocedal, S. J. Wright, *Numerical Optimization, second edition*, Springer, 2006.

[133] W.H. Press, B.P.F. Cornell, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing 2nd Edition*, Press Harvard University, 1993.

[134] E.J. Nyström, "Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben," *Acta Mathematica*, Vol. 54, pp. 185-204, 1930.

[135] M. R. Osborne, B. Presnell and B. A. Turlach, "A new approach to variable selection in least squares problems," *IMA J. Numerical Analysis*, Vol. 20, pp. 389-403, 2000.

[136] E. Osuna, R. Freund, F. Girosi, "Support vector machines: Training and applications," *AI Memo 1602*, Massachusetts Institute of Technology, 1997.

[137] E. Osuna, F. Girosi, "Reducing the run-time complexity of support vector machines," *In Proceedings of the International Conference on Pattern Recognition*, 1998.

[138] M. Ostendorf, S. Roukos, "A stochastic segment model for phoneme-based continuous speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processsing*, Vol. 37, pp. 1857-1869, 1989.

[139] M. Ostendorf, V.V. Digalakis, O. A. Kimball, "From HMM's to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 4(5), pp. 360-378, 1996.

[140] Y. C. Pati, R. Rezaifar and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet

decomposition," *in Proc. of the 27-th Annual Asilomar Conference on Signals, Systems, and Computers*, pp. 40-44, 1993.

[141] G.E. Peterson, "Control methods used in a study of the vowels," *Journal of the Acoustical Society of America*, Vol. 24, pp. 175-184, 1952.

[142] K. Pelckmans, "*Primal-Dual Kernel Machines*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), 2005.

[143] F. Pérez-Cruz and C. Bousoño-Calzón and A. Artés-Rodríguez, "Convergence of the IRWLS Procedure to the Support Vector Machine Solution," *Neural Computation*, Vol. 17, pp. 7-18, 2005.

[144] P.J. Phillips, "Support Vector Machines Applied to Face Recognition," *Advances in Neural Information Processing Systems*, Vol. 11, 803-809, 1999.

[145] J.C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods: Support Vector Learning*, MIT Press, pp. 185-208, 1999.

[146] J.C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, MIT Press, pp. 61-74, 1999.

[147] V. Popovici, S. Bengio, J.-P. Thiran, "Kernel Matching Pursuit for Large Datasets," *Pattern Recognition*, Vol. 38(12), pp. 2385-2390, 2005.

[148] D. Price, S. Knerr, L. Personnaz, G. Dreyfus, "Pairwise neural network classifiers with probabilistic outputs," *Neural Information Processing Systems*, Vol. 7, pp. 1109-1116, 1995.

[149] L.R. Rabiner, B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[150] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition, *Proc. IEEE*, Vol. 77(2), pp. 257-286, 1989.

[151] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[152] A. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks*, Vol. 5(2),pp. 298-305, 1994.

[153] V. Roth, "Probabilistic Discriminative Kernel Classifiers for Multi-class Problems," *In Proc. of the 23rd DAGM-Symposium on Pattern Recognition*, Munich, Germany, pp. 246-253, Springer, 2001.

[154] V. Roth, "The generalized LASSO," *IEEE Transactions on Neural Networks*, Vol. 15, pp. 16-28, 2004.

[155] L. Tóth, A. Kocsor, "Telephone speech recognition via the combination of knowledge sources in a segmental speech model," *Acta Cybernetica*, Vol. 16, pp. 643-657, 2004.

[156] M. Schafföner, S.E. Krüger, E. Andelic, M. Katz, A. Wendemuth, "Limited training data robust speech recognition using kernel-based acoustic models", *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toulouse, France, Vol. 1, pp. 1137-1140, 2006.

[157] B. Schölkopf, K. Tsuda, J-P. Vert, *Kernel methods in computational biology*, MIT Press, 2004

[158] M. Seeger, "Cross-Validation Optimization for Large Scale Structured Classification Kernel Methods," *Journal of Machine Learning Research*, Vol. 9, pp. 1147-1178, 2008.

[159] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[160] S. Shevade, S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, Vol. 19, pp. 2246-2253, 2003.

[161] H. Shimodaira, K. Noma, M. Nakai, S. Sagayama, "Dynamic time-alignment kernel in support vector machine," *In Advances in Neural Information Processing Systems (NIPS)*, Vol. 14, 2002.

[162] N. Smith, M. Gales, "Speech recognition using SVMs," *Advances in Neural Information Processing Systems*, Vol. 14, pp. 1197-1204, 2002.

[163] A.J. Smola, P.L. Bartlett, "Sparse greedy Gaussian process regression," *in Proc. Neural Information Processing Systems*, Vol. 13, pp. 619-625, 2001.

[164] S. Sonnenburg, *Machine Learning for Genomic Sequence Analysis*, PhD thesis, Fraunhofer Institute FIRST, 2008.

[165] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, 2002.

[166] F.H. Sinz, O. Chapelle, C. Santa Clara, A. Agarwal, B. Scholkopf, "An Analysis of Inference with the Universum," *In Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1Ű8, 2008.

[167] T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization," *SIAM Journal on Numerical Analysis*, Vol. 20, pp. 626-637, 1983.

[168] J.A.K. Suykens, J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, Vol. 9(3), pp. 293-300, 1999.

[169] J.A.K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, J. Vandewalle, "Least squares support vector machine classifiers: a large scale algorithm," *in Proc. of the European Conference on Circuit Theory and Design*, pp. 839-842, 1999.

[170] J.A.K Suykens, L. Lukas, J. Vandewalle, "Sparse approximation using least squares support vector machines," *In IEEE Proc. Int. Symp. Circuits Syst.*, Geneva, Switzerland, pp. 757-760, 2000.

[171] J.A.K Suykens, J. Vandewalle, B. De Moor, "Intelligence and cooperative search by coupled local minimizers," *International Journal of Bifurcation and Chaos*, Vol. 11(8), pp. 2133-2144, 2001.

[172] 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.

[173] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, Vol. 48(1-4), pp. 85-105, 2002.

[174] J. Tebelskis, "*Speech Recognition using Neural Networks*," PhD thesis, Carnegie Mellon University, Pittsburgh, USA, 1995.

[175] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal. Statist. Soc B.*, vol. 58(1), 1996.

[176] TIMIT Acoustic-Phonetic Continuous Speech Corpus, *National Institute of Standards and Technology Speech Disc 1 -1.1*, NTIS Order No. PB91 -5050651996, 1990.

[177] M.E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, Vol. 1, pp. 211-244, 2001.

[178] J.A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, Vol. 50(10), pp. 2231-2242, 2004.

[179] C.J. Merz, P.M. Murphy, "UCI repository of machine learning databases," *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1998.

[180] B. Van Calster, J. Luts, J.A.K. Suykens, G. Condous, T. Bourne, D. Timmerman, S. Van Huffel, "Comparing methods for multi-class probabilities in medical decision making using LS-SVMs and kernel logistic regression," *in Proc. of the 17th International Conference on Artificial Neural Networks*, Porto, Portugal, vol. 4669, pp. 139-148, 2007.

[181] T. Van Gestel, J.A.K Suykens, D.-E Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, J. Vandewalle, "Financial Time Series Prediction Using Least Squares Support Vector Machines Within the Evidence Framework," *IEEE Transactions on Neural Networks*, Vol. 12 (4), pp. 809-821, 2001.

[182] T. Van Gestel, J.A.K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, J. Vandewalle, "Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel Fisher discriminant analysis," *Neural Computation*, Vol. 14(5), pp. 1115-1147, 2002.

[183] V. Vapnik, "Transductive inference and semisupervised learning," *Semi-Supervised Learning*, pp. 454-472, 2006.

[184] V.N. Vapnik, *Statistical learning theory*, Wiley, New York, 1998.

[185] J. Verhasselt, I. Illina, J.-P. Martens, Y. Gong, J.-P. Haton, "Assessing the importance of the segmentation probability in segment-based speech recognition," *Speech Communications*, Vol. 24 (1), pp. 51-72, 1998.

[186] J.-P. Vert, H. Saigo, T. Akutsu, "Local alignment kernels for biological sequences," *Kernel Methods in Computational Biology*, pp. 131-154, 2004.

[187] P. Vincent, Y. Bengio, "Kernel Matching Pursuit," *Machine Learning*, Vol. 48, pp. 165-187, 2002.

[188] J. Weston, R. Collobert, F. Sinz, L. Bottou and V. Vapnik, "Inference with the Universum," *In Proc. of International Conference on Machine Learning*, Pittsburgh, Pennsylvania, U.S.A., pp. 1009-1016, 2006.

[189] C.K.I. Williams, M. Seeger, "Using the Nyström Method to Speed Up Kernel Machines," *Advances in Neural Information Processing Systems*, Vol 13., pp. 682-688, 2001.

[190] M. Wu, B. Schölkopf, and G. Bakir, "Building sparse large margin classifiers," *In Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, pp. 996-1003, 2005.

[191] T.-F. Wu, C.J. Lin, R.C. Weng, "Probability estimates for multi-class classification by pairwise coupling.," *Journal of Machine Learning Research*, Vol. 5, pp. 975-1005, 2004.

[192] D. Yu, L. Deng, A. Acero, "Using Continuous Features in the Maximum Entropy Model", *Pattern Recognition Letters*, Vol. 30 (8), 2009.

[193] D. Yu, L. Deng, A. Acero, "Hidden Conditional Random Field with Distribution Constraints for Phone Classification", *Interspeech*, Brighton, UK, pp. 676-679, 2009.

[194] D. Young, *Iterative Solution of Large Linear Systems*, Courier Dover Publications, 2003.

[195] S. Xavier de Souza, J.A.K. Suykens, J. Vandewalle, D. Bollé, "Cooperative behavior in coupled simulated annealing processes with variance control," *In Proc. of the International Symposium on Nonlinear Theory and its Applications (NOLTA2006)*, Bologna, Italy, pp. 114-119, 2006.

[196] S. Xavier de Souza, *Optimization and robustness of cellular neural networks*, PhD thesis, Faculty of Engineering, K.U.Leuven, ESAT-SCD, Belgium, 2007.

[197] S. Xavier de Souza, J.A.K. Suykens, J. Vandewalle, D. Bollé, "Coupled Simulated Annealing," *IEEE Transactions on Systems, Man and Cybernetics - Part B*, Vol. 40(2), pp. 320-335, 2010.

[198] B. Zadrozny, C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," *In Proc. of International Joint Conference on Neural Networks*, Edmonton, Alberta, Canada, pp. 2970-2977, 2006.

[199] G. Zavaliagkos, J. Zhao, R. Schwartz, J. Makhoul, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition" *IEEE Trans. Speech and Audio Proc.*, Vol. 2(1), pp. 151-159, 1994.

[200] X.Y. Zeng, X.W. Chen, "SMO-based pruning methods for sparse least squares support vector machines," *IEEE Transactions in Neural Networks*, Vol. 16(6), pp. 1541-1546, 2005.

[201] K. Zhang, I.W. Tsang, J.T. Kwok, "Improved Nyström low-rank approximation and error analysis," *In Proc. of the 25th International Conference on Machine Learning*, Helsinki, Finland, pp. 1232-1239, 2008.

[202] J. Zhu, T. Hastie, "Kernel logistic regression and the import vector machine," *Advances in Neural Information Processing Systems*, Vol. 14, pp. 1081-1088, 2001.

[203] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, "1-Norm support vector machines," *Advances in Neural Information Processing Systems*, MIT Press, 2003.

[204] J. Zhu and T. Hastie, "Classification of gene microarrays by penalized logistic regression," *Biostatistics*, Vol. 5, pp. 427-444, 2004.

[205] X. Zhu, "Semi-supervised learning literature survey," *Technical report*, `http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf` Computer Sciences, University of Wisconsin-Madison, 2007.

[206] V. Zue, J. Glass, M. Phillips, S. Seneff, "Acoustic segmentation and phonetic classification in the SUMMIT system," *In Proc. International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, UK, Vol. 1, pp. 389-392, 1989.

Biography

Peter Karsmakers was born on April 14, 1979. He received a M.Sc. degree ("industrieel ingenieur") in electronics-ICT from the Katholieke Hogeschool Kempen (K.H.Kempen) in 2001. From that moment on he started working there with a main assignment of teaching courses in the context of electronics and informatics. In 2004 he received the degree of Master in Artificial Intelligence from the Katholieke Universiteit Leuven (K.U.Leuven). He currently pursues a Ph.D. at the K.U.Leuven in the faculty of Applied Sciences, department of Electrical Engineering in the SCD/SISTA research division. His research mainly focusses on designing machine learning methods in the context of speech recognition.

Publications by the author

Submitted journal papers

- P. Karsmakers, K. Pelckmans, H. Van hamme, J.A.K. Suykens, "Large-Scale Kernel Logistic Regression for Segment-Based Phoneme Recognition",*Internal Report 09-174*, ESAT-SISTA, K.U.Leuven, Leuven, 2009, submitted for publication.
- P. Karsmakers, K. Pelckmans, K. De Brabanter, H. Van hamme, J.A.K. Suykens, "Sparse Conjugate Directions Pursuit with Application to Fixed-size Kernel Models",*Internal Report 10-63*, ESAT-SISTA, K.U.Leuven, Leuven, 2010, submitted for publication.

Conference papers

- K. Pelckmans, P. Karsmakers, J.A.K. Suykens, B. De Moor, "Ordinal Least Squares Support Vector Machines - a Discriminant Analysis Approach," *in Proc. of the Machine Learning for Signal Processing (MLSP 2006)*, Maynooth, Ireland, 2006.
- P. Karsmakers, K. Pelckmans, J.A.K. Suykens, "Multi-class kernel logistic regression: a fixed-size implementation," *In Proc. of the international joint conference in neural networks (IJCNN)*, Orlando, Florida, U.S.A, pp. 1756-1761, 2007.
- P. Karsmakers, K. Pelckmans, J.A.K. Suykens, H. Van hamme, "Fixed-Size Kernel Logistic Regression for Phoneme Classification,"*In Proc. of INTERSPEECH*, Antwerpen, Belgium, pp. 78-81, 2007.

- P. Karsmakers, P. Leroux, J. De Brabanter, J.A.K. Suykens,"Least Squares Support Vector Machines for Modelling Electronic Devices," *in Proc. of the European Conference on the Use of Modern Information and Communication Technologies (ECUMICT)*, Ghent, Belgium, pp. 237-248, 2008.

- T. Wu, P. Karsmakers, H. Van hamme, D. Van Compernolle, "Comparison of Variable Selection Methods and Classifiers for Native Accent Identification," *in In Proc. International Conference on Spoken Language Processing*, Brisbane, Australia, pp. 305-308, 2008.

- K. De Brabanter, P. Dreesen, P. Karsmakers, K. Pelckmans, J. De Brabanter, J.A.K. Suykens, B. De Moor, "Fixed-Size LS-SVM Applied to the Wiener-Hammerstein Benchmark," *in Proc. of the 15th IFAC Symposium on System Identification (SYSID 2009)*, Saint-Malo, France, pp. 826-831, 2009.

- K. De Brabanter, S. Sahhaf, P. Karsmakers, J. De Brabanter, J.A.K. Suykens, B. De Moor,"Nonparametric Comparison of Densities Based on Statistical Bootstrap," *Accepted for publication in proceedings of the Fourth European Conference on the Use of Modern Information and Communication Technologies- ECUMICT 2010*, 2009.

Internal reports

- P. Karsmakers, B. Gierlichs, K. Pelckmans, K. De Cock, J.A.K. Suykens, B. Preneel, B. De Moor, "Side channel attacks on cryptographic devices as a classification problem," *Internal Report 07-36*, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2007.

Arenberg Doctoral School of Science, Engineering & Technology
Faculteit ingenieurswetenschappen
Department elektrotechniek
Onderzoeksgroep SCD/SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Belgium)



