Deep Restricted Kernel Machines using Conjugate Feature Duality

Johan A.K. Suykens
KU Leuven, ESAT-STADIUS
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
Email: johan.suykens@esat.kuleuven.be

Neural Computation, in press
(accepted: March 16, 2017; first version: March 7, 2016)

Abstract

The aim of this paper is to propose a theory of *Deep Restricted Kernel Machines* offering new foundations for deep learning with kernel machines. From the viewpoint of deep learning it is partially related to restricted Boltzmann machines, which are characterized by visible and hidden units in a bipartite graph without hidden-to-hidden connections, and deep learning extensions as deep belief networks and deep Boltzmann machines. From the viewpoint of kernel machines it includes least squares support vector machines for classification and regression, kernel PCA, matrix SVD, and Parzen-type models. A key element in this paper is to first characterize these kernel machines in terms of so-called *conjugate feature duality*, yielding a representation with visible and hidden units. It is shown how this is related to the energy form in restricted Boltzmann machines, with continuous variables in a nonprobabilistic setting. In this new framework of so-called *Restricted Kernel Machine* (RKM) representations the dual variables correspond to hidden features. Deep Restricted Kernel Machines (Deep RKM) are obtained by coupling the Restricted Kernel Machines. The method is illustrated for Deep Restricted Kernel Machines consisting of three levels with a LS-SVM regression level and two kernel PCA levels. In its primal form also deep feedforward neural networks can be trained within this framework.

Keywords. deep learning, kernel methods, least squares support vector machines, kernel PCA, restricted Boltzmann machines, feedforward neural networks, duality.

1 Introduction

Deep learning has become an important method of choice in several research areas such as computer vision, speech recognition, language processing and others [25]. Among the existing techniques in deep learning are deep belief networks, deep Boltzmann machines, convolutional neural networks, stacked autoencoders with pretraining and finetuning, and others [4, 15, 17, 18, 25, 26, 35, 38, 45, 7, 20, 42, 60]. On the other hand also support vector machines (SVM) and kernel based methods have made a large impact in a wide range of application fields, together with finding strong foundations in optimization and learning theory [5, 9, 31, 41, 50, 56, 57]. Therefore one can pose the question: which synergies or common foundations could be developed between these different directions? One has started already to explore such synergies e.g. in kernel methods for deep learning [8], deep Gaussian processes [10, 34], convolutional kernel networks [27], multi-layer support vector machines [59], mathematics of the neural response [43], and others.

In this paper we intend to present a new theory of *Deep Restricted Kernel Machines* (Deep RKM) offering foundations for deep learning with kernel machines. It partially relates to restricted Boltzmann machines which are used within deep belief networks [17, 18]. In restricted Boltzmann machines (RBM) one considers a specific type of Markov random field, characterized by a bipartite graph consisting of a layer of visible units and another layer of hidden units [4, 12, 18, 35]. In RBMs, which are related to harmoniums [44, 58], there are no connections between the hidden units [17], and often also no visible to visible connections. In deep belief networks the hidden units of a layer are mapped to a next layer in order to create a deep architecture. In RBM one considers stochastic binary variables [1, 16], and extensions have been made to Gaussian-Bernoulli variants [35]. On the other hand Hopfield networks [19] take continuous values and a class of Hamiltonian neural networks has been studied in [11]. Also discriminative RBMs have been studied where the class labels are considered at the level of visible units [12, 22]. In all these methods the energy function plays an important role, also in energy based learning methods [24].

Representation learning issues are considered to be important in deep learning [3]. The method that is proposed in this paper will make a link to restricted Boltzmann machines by characterizing several kernel machines by means of so-called *conjugate feature duality*. Duality is important in the context of support vector machines [5, 9, 56, 50, 52], optimization [6, 32] and in mathematics and physics in general. Here we will consider hidden features conjugated to part of the unknown variables. This part of the formulation is linked to a restricted Boltzmann machine energy expression, though with continuous variables in a nonprobabilistic setting. In this way one can express a model both in its primal representation and in its dual representation, and give an interpretation in terms of visible and hidden units, in analogy with RBM. The primal representation contains the feature map, while the dual model representation is expressed in terms of the kernel function and

the conjugated features. The class of kernel machines discussed in this paper includes least squares support vector machines (LS-SVM) for classification and regression, kernel principal component analysis (kernel PCA), matrix singular value decomposition (matrix SVD), and Parzen-type models. These have been previously conceived within a primal and Lagrange dual setting in [49, 50, 51, 53, 54]. Other examples are kernel spectral clustering [2, 28], kernel canonical correlation analysis [50] and several others, which will not be addressed in this paper but can be the subject of future work. In this paper we will give a different characterization for these models. It is based on a property of quadratic forms, which can be verified through the Schur complement form. The property relates to a specific case of Legendre-Fenchel duality [32]. Also note that in classical mechanics converting a Lagrangian into Hamiltonian formulation is by Legendre transformation [13].

The kernel machines with conjugate feature representations are used then as building blocks to obtain the Deep RKM by coupling the RKMs. The Deep RKM becomes unrestricted after coupling the RKMs. The approach is explained for a model with 3 levels, consisting of two kernel PCA levels and a level with LS-SVM classification or regression. The conjugate features of level 1 are taken as input of level 2, and subsequently the features of level 2 as input for level 3. The objective of the Deep RKM is the sum of the objectives of the RKMs in the different levels. The characterization of the stationary points leads to solving a set of nonlinear equations in the unknowns, which is computationally expensive. However, for the case of linear kernels in part of the levels it reveals how kernel fusion is taking place over the different levels. For this case a heuristic algorithm is obtained with level-wise solving. For the general nonlinear case a reduced set algorithm with estimation in the primal is proposed.

In this paper we make a distinction between *levels* and *layers*. We use the terminology of *levels* to indicate the depth of the model. The terminology of *layers* is used here in connection to the feature map. It was shown in [48] how a multilayer perceptron can be trained by a support vector machine method. It is done by defining the hidden layer to be equal to the feature map. In this way the hidden layer is treated at the feature map and at the kernel parameters level. In [50] it has been explained that in support vector machine and least squares support vector machine models, one can have a neural networks interpretation both in the primal and in the dual. The number of hidden units in the primal equals the dimension of the feature space, while in the dual representation it equals the number of support vectors. In this way it provides a setting to work with parametric models in the primal and kernel-based models in the dual. Therefore, we will also illustrate in this paper how deep multilayer feedforward neural networks can be trained within the Deep RKM framework. While in the classical backpropagation [33] one typically learns the model by specifying a single objective (unless e.g. imposing additional stability constraints to obtain stable multilayer recurrent networks with dynamic backpropagation [47], in the Deep RKM the objective function consists of the different objectives related to the different

levels.

In summary, we aim at contributing to the following challenging questions in this paper:

- can we find new synergies and foundations between support vector machines & kernel methods and deep learning architectures?
- can we extend primal and dual model representations, as occuring in SVM and LS-SVM models, from shallow to deep architectures?
- can we handle deep feedforward neural networks and deep kernel machines within a common setting?

In order to address these questions, this paper is organized as follows. In Section 2 the context of this paper is outlined with a brief introductory part on restricted Boltzmann machines, support vector machines, least squares support vector machines, kernel PCA and singular value decomposition. In Section 3 we explain how these kernel machines can be characterized by conjugate feature duality with visible and hidden units. In Section 4 deep restricted kernel machines are explained for 3 levels consisting of a LS-SVM regression level and two additional kernel PCA levels. In Section 5 different algorithms are proposed for solving either in the primal or in the dual, where the former will be related to deep feedfoward neural networks and the latter to kernel-based models. Illustrations with numerical examples are given in Section 6. Finally, Section 7 concludes the paper.

2 Preliminaries and Context

In this section we explain basic principles of restricted Boltzmann machines, support vector machines, least squares support vector machines, and related formulations for kernel principal component analysis and singular value decomposition. These are basic ingredients needed before introducing restricted kernel machines in Section 3.

2.1 Restricted Boltzmann machines (RBM)

A restricted Boltzmann machine is a specific type of Markov random field, characterized by a bipartite graph consisting of a layer of visible units and another layer of hidden units [4, 12, 18, 35], without hidden-to-hidden connections. Both the visible and hidden variables, denoted by v and h respectively, have stochastic binary units with value 0 or 1. A joint state $\{v, h\}$ is defined for these visible and hidden variables with energy (see Figure 1)

$$E(v, h; \theta) = -v^T W h - c^T v - a^T h \tag{1}$$

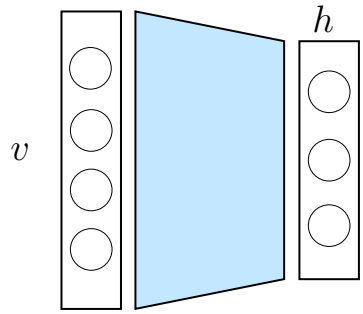

Figure 1: Restricted Boltzmann machine (RBM) consisting of a layer of visible units v and a layer of hidden units h. They are interconnected through the interaction matrix W, depicted in blue color.

where $\theta = \{W, c, a\}$ are the model parameters, W is an interaction weight matrix and c, a contain bias terms.

One obtains then the joint distribution over the visible and hidden units as

$$P(v, h; \theta) = \frac{1}{Z(\theta)} \exp(-E(v, h; \theta)) \tag{2}$$

with the partition function

$$Z(\theta) = \sum_v \sum_h \exp(-E(v, h; \theta))$$

for normalization.

Thanks to the specific bipartite structure one can obtain an explicit expression for the marginalization $P(v; \theta) = \frac{1}{Z(\theta)} \sum_h \exp(-E(v, h; \theta))$. The conditional distributions are obtained as

$$\begin{aligned} P(h|v; \theta) &= \prod_j p(h_{(j)}|v) \\ P(v|h; \theta) &= \prod_i p(v_{(i)}|h) \end{aligned} \tag{3}$$

where $p(h_{(j)} = 1|v) = \sigma(\sum_i W_{ij} v_{(i)} + a_j)$ and $p(v_{(i)} = 1|h) = \sigma(\sum_j W_{ij} h_{(j)} + d_i)$ with $\sigma(x) = 1/(1 + \exp(-x))$ the logistic function. Here $v_{(i)}$ and $h_{(j)}$ denote the i-th visible unit and the j-th hidden unit, respectively.

Because exact maximum likelihood for this model is intractable a contrastive divergence algorithm is used with the following update equation for the weights

$$\Delta W = \alpha(\mathbb{E}_{P_{\text{data}}}(vh^T) - \mathbb{E}_{P_T}(vh^T)) \tag{4}$$

with learning rate α and $\mathbb{E}_{P_{\text{data}}}$ the expectation wrt the data distribution $P_{\text{data}}(h, v; \theta) = P(h|v; \theta) P_{\text{data}}(v)$ where $P_{\text{data}}(v)$ denotes the empirical distribution. Furthermore $\mathbb{E}_{P_T}$ is a

distribution defined by running a Gibbs chain for T steps initialized at the data. Often one takes $T = 1$, while $T \to \infty$ recovers the maximum likelihood approach [35].

In Boltzmann machines there are besides visible-to-hidden, also visible-to-visible, and hidden-to-hidden interaction terms with

$$E(v, h; \theta) = -v^T W h - \frac{1}{2} v^T L v - \frac{1}{2} h^T G h \tag{5}$$

and $\theta = \{W, L, G\}$ as explained in [36].

In Section 3 we will make a connection between the energy expression (1) and a new representation of least squares support vector machines and related kernel machines, which will be made in terms of visible and hidden units. Therefore, we will now briefly review basics of support vector machines, least squares support vector machines, kernel principal component analysis and singular value decomposition.

2.2 Least Squares Support Vector Machines and related Kernel Machines

2.2.1 SVM and LS-SVM

Assume a binary classification problem with training data $\{(x_i, y_i)\}_{i=1}^N$ with input data $x_i \in \mathbb{R}^d$ and corresponding class labels $y_i \in \{-1, 1\}$. A support vector machine classifier takes the form

$$\hat{y} = \text{sign}[w^T \varphi(x) + b]$$

where the feature map $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{n_f}$ maps the data from the input space to a high dimensional feature space and $\hat{y}$ is the estimated class label for a given input point $x \in \mathbb{R}^d$. The training problem for this SVM classifier [5, 9, 56] is

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{2} w^T w + \mathsf{c} \sum\limits_{i=1}^N \xi_i \\ \text{subject to} & y_i[w^T \varphi(x_i) + b] \geq 1 - \xi_i, \quad i = 1, ..., N \\ & \xi_i \geq 0, \quad i = 1, ..., N \end{array} \tag{6}$$

where the objective function makes a trade-off between minimization of the regularization term (corresponding to maximization of the margin $2/\|w\|_2$) and the amount of misclassifications, controlled by the regularization constant $\mathsf{c} > 0$. The slack variables ξ_i are needed to tolerate misclassifications on the training data, in order to avoid that one would overfit the data. The following dual problem in the Lagrange multipliers α_i is obtained, related

to the first set of constraints:

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum\limits_{i,j=1}^{N} y_i y_j\, K(x_i,x_j)\,\alpha_i\alpha_j + \sum\limits_{j=1}^{N}\alpha_j \\ \text{subject to} & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \mathsf{c},\ i=1,...,N. \end{array} \tag{7}$$

Here a positive definite kernel K is used with $K(x,z) = \varphi(x)^T\varphi(z) = \sum_{j=1}^{n_f}\varphi_j(x)\varphi_j(z)$. The SVM classifier is expressed in the dual as

$$\hat{y} = \text{sign}[\sum_{i\in\mathcal{S}_{SV}} \alpha_i\, y_i\, K(x_i,x) + b], \tag{8}$$

where $\mathcal{S}_{SV}$ denotes the set of support vectors, corresponding to the non-zero α_i values. Common choices are e.g. to take a linear $K(x_i,x_j) = x_i^T x_j$, polynomial $K(x_i,x_j) = (\nu + x_i^T x_j)^d$ with $\nu \geq 0$ or Gaussian radial basis function kernel $K(x_i,x_j) = \exp(-\|x_i - x_j\|_2^2/\sigma^2)$.

The LS-SVM classifier [49] is a modification to it

$$\begin{array}{ll} \min\limits_{w,b,e_i} & \frac{1}{2}w^T w + \gamma\,\frac{1}{2}\sum\limits_{i=1}^{N} e_i^2 \\ \text{subject to} & y_i[w^T\varphi(x_i) + b] = 1 - e_i, \quad i=1,...,N, \end{array} \tag{9}$$

where the value 1 in the constraints is taken as a target value, instead of a threshold value. This implicitly corresponds to a regression on the class labels ± 1. From the Lagrangian $\mathcal{L}(w,b,e;\alpha) = \frac{1}{2}w^T w + \gamma\,\frac{1}{2}\sum_{i=1}^{N} e_i^2 - \sum_{i=1}^{N}\alpha_i\{y_i[w^T\varphi(x_i) + b] - 1 + e_i\}$ one takes the conditions for optimality $\partial\mathcal{L}/\partial w = 0$, $\partial\mathcal{L}/\partial b = 0$, $\partial\mathcal{L}/\partial e_i = 0$, $\partial\mathcal{L}/\partial\alpha_i = 0$. Writing the solution in α, b gives the square linear system

$$\left[\begin{array}{c|c} \Omega + I/\gamma & y_{1:N} \\ \hline y_{1:N}^T & 0 \end{array}\right]\left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} 1_N \\ \hline 0 \end{array}\right] \tag{10}$$

where $\Omega_{ij} = y_i y_j\,\varphi(x_i)^T\varphi(x_j) = y_i y_j\, K(x_i,x_j)$ and $y_{1:N} = [y_1;...;y_N]$, $1_N = [1;...;1]$ with as classifier in the dual

$$\hat{y} = \text{sign}[\sum_{i=1}^{N}\alpha_i y_i K(x_i,x) + b]. \tag{11}$$

This formulation has also been extended to multiclass problems in [50].

In the LS-SVM regression formulation [50] one performs ridge regression in the feature space with an additional bias term b

$$\begin{array}{ll} \min\limits_{w,b,e_i} & \frac{1}{2}w^T w + \gamma\,\frac{1}{2}\sum\limits_{i=1}^{N} e_i^2 \\ \text{subject to} & y_i = w^T\varphi(x_i) + b + e_i, \quad i=1,...,N \end{array} \tag{12}$$

which gives

$$\left[\begin{array}{c|c} K + I/\gamma & 1_N \\ \hline 1_N^T & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline b \end{array}\right] = \left[\begin{array}{c} y_{1:N} \\ \hline 0 \end{array}\right] \tag{13}$$

with the predicted output

$$\hat{y} = \sum_{i=1}^{N} \alpha_i K(x_i, x) + b \tag{14}$$

where $K_{ij} = K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$. The classifier formulation can also be transformed into the regression formulation, by multiplying the constraints in (9) by the class labels and considering new error variables [50]. In the zero bias term case this corresponds to kernel ridge regression ([37]), which is also related to function estimation in reproducing kernel Hilbert spaces, regularization networks and Gaussian processes, within a different setting [30, 57, 31, 50].

2.2.2 Kernel PCA and matrix SVD

Within the setting of using equality constraints and the L_2 loss function, typical for LS-SVMs, one can characterize the kernel PCA problem [39] as follows, as shown in [50, 51]

$$\begin{array}{ll} \min\limits_{w,b,e_i} & \frac{1}{2} w^T w - \gamma \frac{1}{2} \sum\limits_{i=1}^{N} e_i^2 \\ \text{subject to} & e_i = w^T \varphi(x_i) + b, \quad i = 1, ..., N. \end{array} \tag{15}$$

From the KKT conditions one obtains the following in the Lagrange multipliers α_i

$$K^{(c)} \alpha = \lambda \alpha \quad \text{with} \quad \lambda = 1/\gamma \tag{16}$$

where $K_{ij}^{(c)} = (\varphi(x_i) - \hat{\mu}_\varphi)^T (\varphi(x_j) - \hat{\mu}_\varphi)$ are the elements of the centered kernel matrix $K^{(c)}$, $\hat{\mu}_\varphi = (1/N) \sum_{i=1}^{N} \varphi(x_i)$ and $\alpha = [\alpha_1; ....; \alpha_N]$. In (15) maximizing instead of minimizing also leads to (16). The centering of the kernel matrix is obtained as a result of taking a bias term b in the model. The γ value is treated at a selection level and is chosen so as to correspond to $\lambda = 1/\gamma$ where λ are eigenvalues of $K^{(c)}$. In the zero bias term case $K^{(c)}$ becomes the kernel matrix $K = [\varphi(x_i)^T \varphi(x_j)]$. Also kernel spectral clustering [2] was obtained in this setting by considering a weighted version of the L_2 loss part, weighted by the inverse of the degree matrix of the graph in the clustering problem.

It has been recently shown in [54] that matrix SVD can be obtained from the following primal problem:

$$\begin{array}{ll} \min\limits_{w,v,e,r} & -w^T v + \gamma \frac{1}{2} \sum\limits_{i=1}^{N} e_i^2 + \gamma \frac{1}{2} \sum\limits_{j=1}^{M} r_j^2 \\ \text{subject to} & e_i = w^T \varphi(x_i), \ i = 1, ..., N \\ & r_j = v^T \psi(z_j), \ j = 1, ..., M \end{array} \tag{17}$$

where $\{x_i\}_{i=1}^N$ and $\{z_j\}_{j=1}^M$ are data sets related to two data sources, which in the matrix SVD [14, 46] case correspond to the sets of rows and columns of the given matrix. Here one has two feature maps $\varphi(\cdot)$ and $\psi(\cdot)$. After taking the Lagrangian and the necessary conditions for optimality, the dual problem in the Lagrange multipliers α_i and β_j, related to the first and second set of constraints, results into

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{18}$$

where $A = [\varphi(x_i)^T \psi(z_j)]$ denotes the matrix with ij-th entry $\varphi(x_i)^T \psi(z_j)$, $\lambda = 1/\gamma$ correspond to non-zero eigenvalues and $\alpha = [\alpha_1; ....; \alpha_N]$, $\beta = [\beta_1; ....; \beta_M]$. For a given matrix A, by choosing the linear feature maps $\varphi(x_i) = C^T x_i$, $\psi(z_j) = z_j$ with a compatibility matrix C that satisfies $ACA = A$, this eigenvalue problem corresponds to the SVD of matrix A [54] in connection to Lanczos' decomposition theorem. One can also see that for a symmetric matrix the two data sources coincide and the objective of (17) reduces then to the kernel PCA objective (15) [54], involving only one feature map instead of two feature maps.

3 Restricted Kernel Machines (RKM) and Conjugate Feature Duality

3.1 LS-SVM regression as a restricted kernel machine: linear case

A training data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ is assumed to be given with input data $x_i \in \mathbb{R}^d$ and output data $y_i \in \mathbb{R}^p$ (now with p outputs), where the data are assumed to be i.i.d. and drawn from an unknown but fixed underlying distribution $P(x, y)$, which is a common assumption made in statistical learning theory [56].

We will explain now how LS-SVM regression can be linked to the energy form expression of a restricted Boltzmann machine, with an interpretation in terms of hidden and visible units. In view of these connections with RBMs and the fact that there will be no hidden-to-hidden connections, we will call it a Restricted Kernel Machine (RKM) representation, when this particular interpretation of the model is made. For LS-SVM regression, the part in the RKM interpretation that will take a similar form as the RBM energy function is

$$\begin{aligned} R_{\text{RKM}}(v, h) &= -v^T \tilde{W} h \\ &= -(x^T W h + b^T h - y^T h) \\ &= e^T h \end{aligned} \tag{19}$$

with a vector of hidden units $h \in \mathbb{R}^p$ and a vector of visible units $v \in \mathbb{R}^{n_v}$ with $n_v = d+1+p$ equal to

$$v = \begin{bmatrix} x \\ 1 \\ -y \end{bmatrix} \text{ and } \tilde{W} = \begin{bmatrix} W \\ b^T \\ I_p \end{bmatrix} \tag{20}$$

and $e = y - \hat{y}$ with $\hat{y} = W^T x + b$ the estimated output vector for a given input vector x where $e, y, \hat{y} \in \mathbb{R}^p$, $W \in \mathbb{R}^{d \times p}, b \in \mathbb{R}^p$. Note that b is treated as part of the interconnection matrix by adding a constant 1 within the vector v, which is also frequently done in the area of neural networks [47]. While in RBM the units are binary valued, in the RKM they are continuous valued. The notation "R" in $R_{\text{RKM}}(v,h)$ refers to the fact that the expression is restricted, i.e. there are no hidden-to-hidden connections.

For the training problem, the sum is taken over the training data $\{(x_i, y_i)\}_{i=1}^N$ with

$$\begin{aligned} R_{\text{RKM}}^{\text{train}} &= \sum_{i=1}^N R_{\text{RKM}}(v_i, h_i) \\ &= -\sum_{i=1}^N (x_i^T W h_i + b^T h_i - y_i^T h_i) \\ &= \sum_{i=1}^N e_i^T h_i. \end{aligned} \tag{21}$$

Note that we will adopt the following notation $h_{(j),i}$ to denote the value of the j-th unit for the i-th data point, and $e_i, h_i \in \mathbb{R}^p$ for $i = 1, ..., N$.

We start now from the LS-SVM regression training problem (12) but for the multiple outputs case. We express the objective in terms of $R_{\text{RKM}}^{\text{train}}$ and show how the hidden units can be introduced. Defining $\lambda = 1/\gamma > 0$ we obtain

$$\begin{aligned} J &= \frac{\eta}{2}\text{Tr}(W^T W) + \frac{1}{2\lambda}\sum_{i=1}^N e_i^T e_i \text{ s.t. } e_i = y_i - W^T x_i - b, \forall i \\ &\geq \sum_{i=1}^N e_i^T h_i - \frac{\lambda}{2}\sum_{i=1}^N h_i^T h_i + \frac{\eta}{2}\text{Tr}(W^T W) \text{ s.t. } e_i = y_i - W^T x_i - b, \forall i \\ &= \sum_{i=1}^N (y_i^T - x_i^T W - b^T) h_i - \frac{\lambda}{2}\sum_{i=1}^N h_i^T h_i + \frac{\eta}{2}\text{Tr}(W^T W) \triangleq \underline{J} \\ &= R_{\text{RKM}}^{\text{train}} - \frac{\lambda}{2}\sum_{i=1}^N h_i^T h_i + \frac{\eta}{2}\text{Tr}(W^T W) \end{aligned} \tag{22}$$

where λ, η are positive regularization constants and the first term corresponds to $R_{\text{RKM}}^{\text{train}}$. $\underline{J}$ denotes the lower bound on J. [1] This is based on the property that for two arbitrary

[1] Note that also the term $-\frac{\lambda}{2}\sum_{i=1}^N h_i^T h_i$ appears. This would in a Boltzmann machine energy correspond to matrix G equal to the identity matrix. The term $\frac{\eta}{2}\text{Tr}(W^T W)$ is an additional regularization term.

vectors e, h one has

$$\frac{1}{2\lambda}e^T e \geq e^T h - \frac{\lambda}{2}h^T h, \ \forall e, h \in \mathbb{R}^p. \tag{23}$$

The maximal value of the rhs in (23) is obtained for $h = e/\lambda$ which follows from $\partial(e^T h - \frac{\lambda}{2}h^T h)/\partial h = 0$ and $\partial^2(e^T h - \frac{\lambda}{2}h^T h)/\partial h^2 = -\lambda I < 0$. The maximal value that can be obtained for the rhs equals the lhs $\frac{1}{2\lambda}e^T e$. The property (23) can also be verified by writing it in the quadratic form

$$\frac{1}{2}\begin{bmatrix} e^T & h^T \end{bmatrix}\begin{bmatrix} \frac{1}{\lambda}I & I \\ I & \lambda I \end{bmatrix}\begin{bmatrix} e \\ h \end{bmatrix} \geq 0, \ \forall e, h \in \mathbb{R}^p, \tag{24}$$

which holds. This follows immediately from the Schur complement form[2] which results into the condition $\frac{1}{2}(\lambda I - I(\lambda I)I \geq 0)$, which holds. Writing (23) as

$$\frac{1}{2\lambda}e^T e + \frac{\lambda}{2}h^T h \geq e^T h \tag{25}$$

gives a property that is also known in Legendre-Fenchel duality for the case of a quadratic function [32]. Furthermore it also follows from (23) that

$$\frac{1}{2\lambda}e^T e = \max_h (e^T h - \frac{\lambda}{2}h^T h). \tag{26}$$

We will call the method of introducing the hidden features h_i into (22) *conjugate feature duality*, where the hidden features h_i are conjugated to the e_i. Here $R^{\text{train}}_{\text{RKM}} = \sum_i e_i^T h_i$ will be called an *inner pairing* between the e_i and the hidden features h_i (see Figure 2).

We proceed now by looking at the stationary points of $\underline{J}(h_i, W, b)$: [3]

$$\begin{cases} \frac{\partial \underline{J}}{\partial h_i} = 0 & \Rightarrow \quad y_i = W^T x_i + b + \lambda h_i, \ \forall i \\ \frac{\partial \underline{J}}{\partial W} = 0 & \Rightarrow \quad W = \frac{1}{\eta}\sum_i x_i h_i^T \\ \frac{\partial \underline{J}}{\partial b} = 0 & \Rightarrow \quad \sum_i h_i = 0. \end{cases} \tag{27}$$

The first condition yields $h_i = e_i/\lambda$ which means that the maximal value of $\underline{J}$ is reached. Therefore $y_i = \hat{y}_i + e_i = \hat{y}_i + \lambda h_i$. Also note the similarity between the condition $W =$

[2]This states that for a matrix $Q = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$, one has $Q \geq 0$ if and only if $A > 0$ and the Schur complement $C - B^T A^{-1} B \geq 0$. ([6])

[3]The following properties are used throughout this paper: $\frac{\partial \text{Tr}(X^T B X)}{\partial X} = BX + B^T X, \frac{\partial a^T X b}{\partial X} = ab^T, \frac{\partial a^T X^T b}{\partial X} = ba^T, \frac{\partial \text{Tr}(X^T A)}{\partial X} = A, \frac{\partial \text{Tr}(AX^T)}{\partial X} = A, \frac{\partial \text{Tr}(XA)}{\partial X} = A^T, \frac{\partial x^T a}{\partial x} = \frac{\partial a^T x}{\partial x} = a, a^T a = \text{Tr}(aa^T)$ for matrices A, B, X and vectors a, b. ([29])

$\frac{1}{\eta}\sum_i x_i h_i^T$ and the expression (4) in the contrastive divergence algorithm. Elimination of h_i from this set of conditions gives the solution in W, b:

$$\left[\begin{array}{c|c} \sum_i x_i x_i^T + \lambda\eta I_d & \sum_i x_i \\ \hline \sum_i x_i^T & N \end{array}\right]\left[\begin{array}{c} W \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} \sum_i x_i y_i^T \\ \hline \sum_i y_i^T \end{array}\right]. \tag{28}$$

Elimination of W from the set of conditions gives the solution in h_i, b:

$$\left[\begin{array}{c|c} \frac{1}{\eta}[x_i^T x_j] + \lambda I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right]\left[\begin{array}{c} H^T \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} Y^T \\ \hline 0 \end{array}\right] \tag{29}$$

with $[x_i^T x_j]$ denoting the matrix with ij-entry $x_i^T x_j$, $H = [h_1 ... h_N] \in \mathbb{R}^{p\times N}$, $Y = [y_1 ... y_N] \in \mathbb{R}^{p\times N}$. From this square linear system one can solve $\{h_i\}$ and b. 1_N denotes a vector of all ones of size N, I_N the identity matrix of size $N \times N$.

It is remarkable to see here that the hidden features h_i take the same role as the Lagrange dual variables α_i in the LS-SVM formulation based on Lagrange duality (13) when taking $\eta = 1$ and $p = 1$. For the estimated values $\hat{y}_i$ on the training data one can express the model then in terms of W, b or in terms of h_i, b. In the restricted kernel machine interpretation of the LS-SVM regression one has the following primal and dual model representations:

$$\begin{array}{ccc} & \nearrow & (P)_{\text{RKM}}: \quad \hat{y} = W^T x + b \\ \mathcal{M} & & \\ & \searrow & (D)_{\text{RKM}}: \quad \hat{y} = \frac{1}{\eta}\sum_j h_j x_j^T x + b \end{array} \tag{30}$$

evaluated at a point x, where the primal representation is in terms of W, b and the dual representation is in the hidden features h_i. The primal representation is suitable for handling the "large N, small d" case, while the dual representation for "small N, large d" [50].

3.2 Nonlinear case

The extension to the general nonlinear case goes by replacing x_i by $\varphi(x_i)$ where $\varphi(x_i) : \mathbb{R}^d \to \mathbb{R}^{n_f}$ denotes the feature map, with n_f the dimension of the feature space. Therefore the objective function for the RKM interpretation becomes

$$\underline{J} \;=\; \sum_{i=1}^{N}(y_i^T - \varphi(x_i)^T W - b^T)h_i - \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\text{Tr}(W^T W) \tag{31}$$

with vector of visible units $v \in \mathbb{R}^{n_v}$ with $n_v = n_f + 1 + p$ equal to

$$v = \begin{bmatrix} \varphi(x) \\ 1 \\ -y \end{bmatrix}. \tag{32}$$

Following the same approach as in the linear case, one obtains then as solution in the primal

$$\left[\begin{array}{c|c} \sum_j \varphi(x_j)\varphi(x_j)^T + \lambda\eta I_{n_f} & \sum_j \varphi(x_j) \\ \hline \sum_j \varphi(x_j)^T & N \end{array}\right] \left[\begin{array}{c} W \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} \sum_j \varphi(x_j) y_j^T \\ \hline \sum_j y_j^T \end{array}\right]. \tag{33}$$

In the conjugate feature dual one obtains the same linear system as (29) but with the positive definite kernel $K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$ instead of the linear kernel $x_i^T x_j$:

$$\left[\begin{array}{c|c} \frac{1}{\eta}K + \lambda I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right] \left[\begin{array}{c} H^T \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} Y^T \\ \hline 0 \end{array}\right]. \tag{34}$$

We will also employ the notation $[K(x_i, x_j)]$ to denote the kernel matrix K with ij-th entry equal to $K(x_i, x_j)$.

The primal and dual model representations are expressed in terms of the feature map and kernel function, respectively

$$\begin{array}{ccc} & & (P)_{\mathrm{RKM}}: \quad \hat{y} = W^T\varphi(x) + b \\ & \nearrow & \\ \mathcal{M} & & \\ & \searrow & \\ & & (D)_{\mathrm{RKM}}: \quad \hat{y} = \dfrac{1}{\eta}\displaystyle\sum_j h_j K(x_j, x) + b. \end{array} \tag{35}$$

One can define the feature map either in an implicit or in an explicit way. When employing a positive definite kernel function $K(\cdot,\cdot)$, according to the Mercer theorem, there exists a feature map φ such that $K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$ holds. On the other hand one could also explicitly define an expression for φ and construct the kernel function according to $K(x_i, x_j) := \varphi(x_i)^T\varphi(x_j)$. For multilayer perceptrons it was shown in [48] that the hidden layer can be chosen as the feature map. We can let it correspond to

$$\varphi_{\mathrm{FF}}(x) = \sigma(U_q\sigma(...\sigma(U_2\sigma(U_1 x + \beta_1) + \beta_2)...) + \beta_q) \tag{36}$$

related to a feedforward (FF) neural network with multi layers, with hidden layer matrices $U_1, U_2, ..., U_q$ and bias term vectors $\beta_1, \beta_2, ..., \beta_q$. By construction one obtains then $K_{\mathrm{FF}}(x_i, x_j) := \varphi_{\mathrm{FF}}(x_i)^T\varphi_{\mathrm{FF}}(x_j)$. Note that the activation function σ might be different also

for each of the hidden layers. A common choice is a sigmoid or hyperbolic tangent function. Within the context of this paper $U_1, ..., U_q, \beta_1, ..., \beta_q$ are treated then at the feature map level and the kernel parameter level.

As explained in [50] one can also give a neural network interpretation both to the primal and to the dual representation, with a number of hidden units equal to the dimension of the feature space for the primal representation and the number of support vectors in the dual representation, respectively. For the case of a Gaussian RBF kernel one has a one hidden layer interpretation then with an infinite amount of hidden units in the primal, while in the dual the number of hidden units equals the number of support vectors.

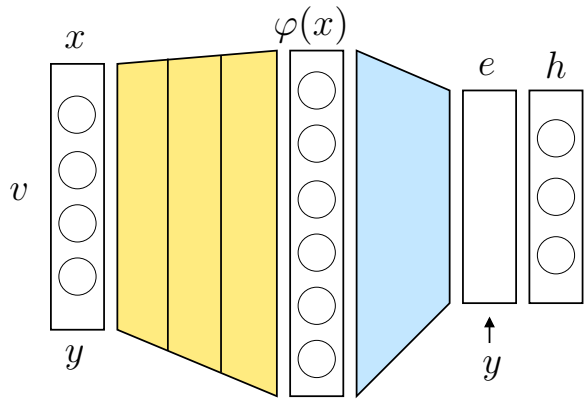

Figure 2: Restricted kernel machine (RKM) representation for regression: the feature map $\varphi(x)$ maps the input vector x to a feature space (possibly by multi layers, depicted in yellow color) and the hidden features are obtained through an inner pairing $e^T h$ where $e = y - \hat{y}$ compares the given output vector y with the predictive model output vector $\hat{y} = W^T \varphi(x) + b$, where the interconnection matrix W is depicted in blue color.

3.3 Classifier formulation

In the multi-class case the LS-SVM classifier constraints are

$$D_{y_i}(W^T \varphi(x_i) + b) = 1_p - e_i, i = 1, ..., N \tag{37}$$

where $y_i \in \{-1, 1\}^p, e_i \in \mathbb{R}^p$ with p outputs encoding the classes and diagonal matrix $D_{y_i} = \text{diag}\{ y_{(1),i}, ..., y_{(p),i}\}$.

In this case, starting from the LS-SVM classifier objective, one obtains

$$\begin{aligned} J &= \frac{\eta}{2}\mathrm{Tr}(W^TW) + \frac{1}{2\lambda}\sum_{i=1}^{N} e_i^T e_i \text{ s.t. } e_i = 1_p - D_{y_i}(W^T\varphi(x_i)+b), \forall i \\ &\geq \sum_{i=1}^{N} e_i^T h_i - \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \text{ s.t. } e_i = 1_p - D_{y_i}(W^T\varphi(x_i)+b), \forall i \\ &= \sum_{i=1}^{N}\left(1_p^T - (\varphi(x_i)^TW + b^T)D_{y_i}\right)h_i - \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \triangleq \underline{J}. \end{aligned} \tag{38}$$

The stationary points of $\underline{J}(h_i, W, b)$ are given by

$$\begin{cases} \frac{\partial \underline{J}}{\partial h_i} = 0 & \Rightarrow \quad 1_p = D_{y_i}(W^T\varphi(x_i)+b) + \lambda h_i, \ \forall i \\ \frac{\partial \underline{J}}{\partial W} = 0 & \Rightarrow \quad W = \frac{1}{\eta}\sum_i \varphi(x_i)h_i^T D_{y_i} \\ \frac{\partial \underline{J}}{\partial b} = 0 & \Rightarrow \quad \sum_i D_{y_i}h_i = 0. \end{cases} \tag{39}$$

The solution in the conjugate features follows then from the linear system:

$$\left[\begin{array}{c|c} \frac{1}{\eta}K + \lambda I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right]\left[\begin{array}{c} H_D^T \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} Y^T \\ \hline 0 \end{array}\right] \tag{40}$$

with $H_D = [D_{y_1}h_1 ... D_{y_N}h_N]$.

The primal and dual model representations are expressed in terms of the feature map and the kernel function, respectively

$$\mathcal{M} \begin{cases} \nearrow & (P)_{\mathrm{RKM}}: \quad \hat{y} = \mathrm{sign}[W^T\varphi(x) + b] \\ \searrow & (D)_{\mathrm{RKM}}: \quad \hat{y} = \mathrm{sign}[\frac{1}{\eta}\sum_j D_{y_j}h_j K(x_j, x) + b]. \end{cases} \tag{41}$$

3.4 Kernel PCA

In the kernel PCA case we start from the objective in (15) and introduce the conjugate hidden features:

$$\begin{aligned} J &= \frac{\eta}{2}\mathrm{Tr}(W^TW) - \frac{1}{2\lambda}\sum_{i=1}^{N} e_i^T e_i \text{ s.t. } e_i = W^T\varphi(x_i), \forall i \\ &\leq -\sum_{i=1}^{N} e_i^T h_i + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \text{ s.t. } e_i = W^T\varphi(x_i), \forall i \\ &= -\sum_{i=1}^{N} \varphi(x_i)^T W h_i + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \triangleq \overline{J} \\ &= -R_{\mathrm{RKM}}^{\mathrm{train}} + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\mathrm{Tr}(W^TW) \end{aligned} \tag{42}$$

where the upper bound $\overline{J}$ is introduced now by relying on the same property as used in the regression/classification case: $\frac{1}{2\lambda}e^Te + \frac{\lambda}{2}h^Th \geq e^Th$, but employed in a different way. Note that

$$-\frac{1}{2\lambda}e^Te = \min_h(-e^Th + \frac{\lambda}{2}h^Th). \tag{43}$$

The minimal value for the rhs is obtained for $h = e/\lambda$ which equals the lhs in that case.

We then proceed by characterizing the stationary points of $\overline{J}(h_i, W)$:

$$\begin{cases} \frac{\partial \overline{J}}{\partial h_i} = 0 \;\; \Rightarrow \;\; W^T\varphi(x_i) = \lambda h_i, \;\; \forall i \\ \frac{\partial \overline{J}}{\partial W} = 0 \;\; \Rightarrow \;\; W = \frac{1}{\eta}\sum_i \varphi(x_i)h_i^T. \end{cases} \tag{44}$$

Note that the first condition yields $h_i = e_i/\lambda$. Therefore the minimum value of $\overline{J}$ is reached. Elimination of W gives the following solution in the conjugated features

$$\frac{1}{\eta}KH^T = H^T\Lambda \tag{45}$$

where $H = [h_1...h_N] \in \mathbb{R}^{s\times N}$ and $\Lambda = \mathrm{diag}\{\lambda_1, ..., \lambda_s\}$ with $s \leq N$ the number of selected components. One can verify that the solutions corresponding to the different eigenvectors h_i and their corresponding eigenvalues λ_i all lead to the value $J = 0$.

The primal and dual model representations are

$$\begin{array}{ccc} & & (P)_{\mathrm{RKM}}: \quad \hat{e} = W^T\varphi(x) \\ & \nearrow & \\ \mathcal{M} & & \\ & \searrow & \\ & & (D)_{\mathrm{RKM}}: \quad \hat{e} = \frac{1}{\eta}\sum_j h_j K(x_j, x). \end{array} \tag{46}$$

Here the number of hidden units equals s with $h \in \mathbb{R}^s$ and the visible units $v \in \mathbb{R}^{n_f}$ with $v = \varphi(x)$, and $R_{\mathrm{RKM}}(v,h) = -v^T W h$.

3.5 Singular value decomposition

For the SVD case we start from the objective in (17) and introduce the conjugated hidden features. The model is characterized now by matrices W, V:

$$\begin{aligned} J &= -\frac{\eta}{2}\mathrm{Tr}(V^T W) + \frac{1}{2\lambda}\sum_{i=1}^{N} e_i^T e_i + \frac{1}{2\lambda}\sum_{j=1}^{M} r_j^T r_j \\ &\quad \text{s.t. } e_i = W^T\varphi(x_i), \forall i \ \&\ r_j = V^T\psi(z_j), \forall j \\ &\geq \sum_{i=1}^{N} e_i^T h_{e_i} - \frac{\lambda}{2}\sum_{i=1}^{N} h_{e_i}^T h_{e_i} + \sum_{j=1}^{M} r_j^T h_{r_j} - \frac{\lambda}{2}\sum_{j=1}^{M} h_{r_j}^T h_{r_j} - \frac{\eta}{2}\mathrm{Tr}(V^T W) \\ &\quad \text{s.t. } e_i = W^T\varphi(x_i), \forall i \ \&\ r_j = V^T\psi(z_j), \forall j \\ &= \sum_{i=1}^{N} \varphi(x_i)^T W h_{e_i} - \frac{\lambda}{2}\sum_{i=1}^{N} h_{e_i}^T h_{e_i} + \sum_{j=1}^{M} \psi(z_j)^T V h_{r_j} - \frac{\lambda}{2}\sum_{j=1}^{M} h_{r_j}^T h_{r_j} - \frac{\eta}{2}\mathrm{Tr}(V^T W) \triangleq \underline{J}. \end{aligned} \tag{47}$$

In this case $R_{\mathrm{RKM}}^{\mathrm{train}} = \sum_{i=1}^{N} \varphi(x_i)^T W h_{e_i} + \sum_{j=1}^{M} \psi(z_j)^T V h_{r_j}$. The stationary points of $\underline{J}(h_{e_i}, W, h_{r_j}, V)$ are given by:

$$\begin{cases} \dfrac{\partial \underline{J}}{\partial h_{e_i}} = 0 \ \Rightarrow \ W^T\varphi(x_i) = \lambda h_{e_i}, \ \forall i \\ \dfrac{\partial \underline{J}}{\partial W} = 0 \ \Rightarrow \ V = \dfrac{1}{\eta}\displaystyle\sum_i \varphi(x_i) h_{e_i}^T \\ \dfrac{\partial \underline{J}}{\partial h_{r_j}} = 0 \ \Rightarrow \ V^T\psi(z_j) = \lambda h_{r_j}, \ \forall j \\ \dfrac{\partial \underline{J}}{\partial V} = 0 \ \Rightarrow \ W = \dfrac{1}{\eta}\displaystyle\sum_j \psi(z_j) h_{r_j}^T. \end{cases} \tag{48}$$

Elimination of W, V gives the solution in the conjugated dual features h_{e_i}, h_{r_j}:

$$\left[\begin{array}{c|c} 0 & \frac{1}{\eta}[\varphi(x_i)^T\psi(z_j)] \\ \hline \frac{1}{\eta}[\psi(z_j)^T\varphi(x_i)] & 0 \end{array}\right] \left[\begin{array}{c} H_e^T \\ \hline H_r^T \end{array}\right] = \left[\begin{array}{c} H_e^T \\ \hline H_r^T \end{array}\right] \Lambda \tag{49}$$

with $H_e = [h_{e_1} ... h_{e_N}] \in \mathbb{R}^{s \times N}$, $H_r = [h_{r_1} ... h_{r_M}] \in \mathbb{R}^{s \times M}$ and $\Lambda = \mathrm{diag}\{\lambda_1, ..., \lambda_s\}$ with $s \leq N + M$ a specified number of non-zero eigenvalues.

The primal and dual model representations are

$$\begin{array}{rll} & (P)_{\text{RKM}}: & \hat{e} = W^T\varphi(x) \\ \nearrow & & \hat{r} = V^T\psi(z) \\ \mathcal{M} & & \\ \searrow & & \\ & (D)_{\text{RKM}}: & \hat{e} = \frac{1}{\eta}\sum_j h_{r_j}\psi(z_j)^T\varphi(x) \\ & & \hat{r} = \frac{1}{\eta}\sum_i h_{e_i}\varphi(x_i)^T\psi(z) \end{array} \tag{50}$$

which corresponds to matrix SVD in the case of linear compatible feature maps and if an additional compatibility condition holds [54].

3.6 Kernel pmf

For the case of kernel probability mass function (kernel pmf) estimation [53], we start from the objective

$$J = \sum_{i=1}^{N}(p_i - \varphi(x_i)^T w)h_i - \sum_{i=1}^{N} p_i + \frac{\eta}{2}w^T w \tag{51}$$

in the unknowns $w \in \mathbb{R}^{n_f}$, $p_i \in \mathbb{R}$ and $h_i \in \mathbb{R}$. In [53] it was explained how a similar formulation is related to the probability rule in quantum measurement for a complex valued model.

The stationary points are characterized by

$$\begin{cases} \frac{\partial J}{\partial h_j} = 0 & \Rightarrow \quad p_i = w^T\varphi(x_i),\ \forall i \\ \frac{\partial J}{\partial w} = 0 & \Rightarrow \quad w = \frac{1}{\eta}\sum_i \varphi(x_i)h_i \\ \frac{\partial J}{\partial p_i} = 0 & \Rightarrow \quad h_i = 1,\ \forall i. \end{cases} \tag{52}$$

The regularization constant η can be chosen to normalize $\sum_i p_i = 1$ ($p_i \geq 0$ is achieved by the choice of an appropriate kernel function), which gives then the kernel pmf obtained in [53]. This results in the representations

$$\begin{array}{rll} & (P)_{\text{RKM}}: & p_i = w^T\varphi(x_i) \\ \nearrow & & \\ \mathcal{M} & & \\ \searrow & & \\ & (D)_{\text{RKM}}: & p_i = \frac{1}{\eta}\sum_j K(x_j, x_i). \end{array} \tag{53}$$

4 Deep Restricted Kernel Machines

In this section we couple different restricted kernel machines within a deep architecture. Several coupling configurations are possible at this point. We illustrate deep restricted kernel machines here for an architecture consisting of three levels. We will discuss two configurations:

1. Two kernel PCA levels followed by an LS-SVM regression level

2. LS-SVM regression level followed by two kernel PCA levels.

In the first architecture the first two levels extract features which are used within the last level for classification or regression. Related type of architectures are stacked autoencoders [4] where a pre-training phase provides a good initialization for training the deep neural network in the fine-tuning phase. The deep RKM on the other hand will consider an objective function jointly related to the kernel PCA feature extractions and to the classification or regression. It will be explained how the insights of the RKM kernel PCA representations can be employed for combined supervised training and feature selection. A difference with other methods is also that conjugated features are used within the layered architecture.

In the second architecture one starts with regression and then let two kernel PCA levels further act on the residuals. In this case connections will be shown with deep Boltzmann machines [35, 36] when considering the special case of linear feature maps, though for the RKMs in a non-probabilistic setting.

4.1 Two kernel PCA levels followed by regression level

We focus here on a Deep RKM architecture consisting of three levels where

- Level 1 consists of kernel PCA with given input data x_i and is characterized by conjugated features $h_i^{(1)}$;

- Level 2 consists of kernel PCA by taking $h_i^{(1)}$ as input and is characterized by conjugated features $h_i^{(2)}$;

- Level 3 consists of LS-SVM regression on $h_i^{(2)}$ with output data y_i and is characterized by conjugated features $h_i^{(3)}$.

As predictive model is taken

$$\begin{aligned} \hat{e}^{(1)} &= W_1^T \varphi_1(x) \\ \hat{e}^{(2)} &= W_2^T \varphi_2(\Lambda_1^{-1}\hat{e}^{(1)}) \\ \hat{y} &= W_3^T \varphi_3(\Lambda_2^{-1}\hat{e}^{(2)}) + b \end{aligned} \tag{54}$$

evaluated at point $x \in \mathbb{R}^d$. The level 1 part has feature map $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{n_{f_1}}$, $W_1 \in \mathbb{R}^{n_{f_1} \times s^{(1)}}$, the level 2 part $\varphi_2 : \mathbb{R}^{n_{f_1}} \to \mathbb{R}^{n_{f_2}}$, $W_2 \in \mathbb{R}^{n_{f_2} \times s^{(2)}}$, and the level 3 part $\varphi_3 : \mathbb{R}^{n_{f_2}} \to \mathbb{R}^{n_{f_3}}$, $W_3 \in \mathbb{R}^{n_{f_3} \times p}$. Note that $\Lambda_1^{-1}\hat{e}^{(1)}$ and $\Lambda_2^{-1}\hat{e}^{(2)}$ (with $\hat{e}^{(1)} \in \mathbb{R}^{s^{(1)}}, \hat{e}^{(2)} \in \mathbb{R}^{s^{(2)}}$) are taken as input for level 2 and 3 respectively, where Λ_1, Λ_2 denote the diagonal matrices with the corresponding eigenvalues. The latter is inspired by the property that for the uncoupled kernel PCA levels the property $h_i = e_i/\lambda$ holds on the training data according to (44), which is further extended then to the out-of-sample case in (54).

The objective function in the primal is

$$J_{\text{deep,P}} = J_1 + J_2 + J_3 \tag{55}$$

with

$$\begin{aligned} J_1 &= -\tfrac{1}{2\lambda_1}\textstyle\sum_{i=1}^N {e_i^{(1)}}^T e_i^{(1)} + \tfrac{\eta_1}{2}\mathrm{Tr}(W_1^T W_1) \\ J_2 &= -\tfrac{1}{2\lambda_2}\textstyle\sum_{i=1}^N {e_i^{(2)}}^T e_i^{(2)} + \tfrac{\eta_2}{2}\mathrm{Tr}(W_2^T W_2) \\ J_3 &= \tfrac{1}{2\lambda_3}\textstyle\sum_{i=1}^N {e_i^{(3)}}^T e_i^{(3)} + \tfrac{\eta_3}{2}\mathrm{Tr}(W_3^T W_3) \end{aligned} \tag{56}$$

with $\hat{e_i}^{(1)} = W_1^T\varphi_1(x_i)$, $\hat{e_i}^{(2)} = W_2^T\varphi_2(\Lambda_1^{-1}\hat{e_i}^{(1)})$, $\hat{y_i} = W_3^T\varphi_3(\Lambda_2^{-1}\hat{e_i}^{(2)}) + b$. However, this objective function is not directly usable for minimization due to the minus sign terms $-\frac{1}{2\lambda_1}\sum_{i=1}^N {e_i^{(1)}}^T e_i^{(1)}$ and $-\frac{1}{2\lambda_2}\sum_{i=1}^N {e_i^{(2)}}^T e_i^{(2)}$. For direct minimization of an objective in the primal we will use the following stabilized version

$$J_{\text{deep,P}_{\text{stab}}} = J_1 + J_2 + J_3 + \frac{1}{2}c_{\text{stab}}(J_1^2 + J_2^2) \tag{57}$$

with c_{stab} a positive constant. The role of this stabilization term for the kernel PCA levels is further explained in Appendix. While in stacked autoencoders one has an unsupervised pretraining and a supervised finetuning phase [4], here we train the whole network at once.

For a characterization of the Deep RKM in terms of the conjugated features $h_i^{(1)}, h_i^{(2)}, h_i^{(3)}$ (Figure 3) we will study the stationary points of

$$J_{\text{deep}} = \overline{J}_1 + \overline{J}_2 + \underline{J}_3 \tag{58}$$

where the objective $J_{\text{deep}}(h_i^{(1)}, W_1, h_i^{(2)}, W_2, h_i^{(3)}, W_3, b)$ for the Deep RKM consists of the sum of the objectives of levels 1,2,3 given by $\overline{J}_1$, $\overline{J}_2$, $\underline{J}_3$, respectively.

This becomes

$$\begin{aligned} J_{\text{deep}} &= -\sum_{i=1}^N \varphi_1(x_i)^T W_1 h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^N {h_i^{(1)}}^T h_i^{(1)} + \frac{\eta_1}{2}\mathrm{Tr}(W_1^T W_1) \\ &\quad -\sum_{i=1}^N \varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} + \frac{\lambda_2}{2}\sum_{i=1}^N {h_i^{(2)}}^T h_i^{(2)} + \frac{\eta_2}{2}\mathrm{Tr}(W_2^T W_2) \\ &\quad +\sum_{i=1}^N (y_i^T - \varphi_3(h_i^{(2)})^T W_3 - b^T) h_i^{(3)} - \frac{\lambda_3}{2}\sum_{i=1}^N {h_i^{(3)}}^T h_i^{(3)} + \frac{\eta_3}{2}\mathrm{Tr}(W_3^T W_3), \end{aligned} \tag{59}$$

with the following inner pairings at the three levels

$$\begin{aligned}
&\text{Level 1}: \quad \sum_{i=1}^{N} {e_i^{(1)}}^T h_i^{(1)} = \sum_{i=1}^{N} \varphi_1(x_i)^T W_1 h_i^{(1)} \\
&\text{Level 2}: \quad \sum_{i=1}^{N} {e_i^{(2)}}^T h_i^{(2)} = \sum_{i=1}^{N} \varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} \\
&\text{Level 3}: \quad \sum_{i=1}^{N} {e_i^{(3)}}^T h_i^{(3)} = \sum_{i=1}^{N} (y_i^T - \varphi_3(h_i^{(2)})^T W_3 - b^T) h_i^{(3)}.
\end{aligned} \tag{60}$$

The stationary points of $J_{\text{deep}}(h_i^{(1)}, W_1, h_i^{(2)}, W_2, h_i^{(3)}, W_3, b)$ are given by

$$\begin{cases}
\frac{\partial J_{\text{deep}}}{\partial h_i^{(1)}} = 0 \;\Rightarrow\; W_1^T \varphi_1(x_i) = \lambda_1 h_i^{(1)} - \frac{\partial}{\partial h_i^{(1)}}[\varphi_2(h_i^{(1)})^T W_2 h_i^{(2)}], \;\forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_1} = 0 \;\Rightarrow\; W_1 = \frac{1}{\eta_1} \sum_i \varphi_1(x_i) {h_i^{(1)}}^T \\
\frac{\partial J_{\text{deep}}}{\partial h_i^{(2)}} = 0 \;\Rightarrow\; W_2^T \varphi_2(h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{\partial}{\partial h_i^{(2)}}[\varphi_3(h_i^{(2)})^T W_3 h_i^{(3)}], \;\forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_2} = 0 \;\Rightarrow\; W_2 = \frac{1}{\eta_2} \sum_i \varphi_2(h_i^{(1)}) {h_i^{(2)}}^T \\
\frac{\partial J_{\text{deep}}}{\partial h_i^{(3)}} = 0 \;\Rightarrow\; y_i - W_3^T \varphi_3(h_i^{(2)}) - b = \lambda_3 h_i^{(3)}, \;\forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_3} = 0 \;\Rightarrow\; W_3 = \frac{1}{\eta_3} \sum_i \varphi_3(h_i^{(2)}) {h_i^{(3)}}^T \\
\frac{\partial J_{\text{deep}}}{\partial b} = 0 \;\Rightarrow\; \sum_i h_i^{(3)} = 0.
\end{cases} \tag{61}$$

The primal and dual model representations for the Deep RKM are then

$$\mathcal{M} \begin{cases}
\nearrow & (P)_{\text{DeepRKM}}: \begin{aligned} \hat{e}^{(1)} &= W_1^T \varphi_1(x) \\ \hat{e}^{(2)} &= W_2^T \varphi_2(\Lambda_1^{-1} \hat{e}^{(1)}) \\ \hat{y} &= W_3^T \varphi_3(\Lambda_2^{-1} \hat{e}^{(2)}) + b \end{aligned} \\
\searrow & (D)_{\text{DeepRKM}}: \begin{aligned} \hat{e}^{(1)} &= \tfrac{1}{\eta_1} \textstyle\sum_j h_j^{(1)} K_1(x_j, x) \\ \hat{e}^{(2)} &= \tfrac{1}{\eta_2} \textstyle\sum_j h_j^{(2)} K_2(h_j^{(1)}, \Lambda_1^{-1} \hat{e}^{(1)}) \\ \hat{y} &= \tfrac{1}{\eta_3} \textstyle\sum_j h_j^{(3)} K_3(h_j^{(2)}, \Lambda_2^{-1} \hat{e}^{(2)}) + b. \end{aligned}
\end{cases} \tag{62}$$

By elimination of W_1, W_2, W_3 one obtains the following set of nonlinear equations in the conjugated features $h_i^{(1)}, h_i^{(2)}, h_i^{(3)}$ and b:

$$\begin{cases} \frac{1}{\eta_1}\sum_j h_j^{(1)} K_1(x_j, x_i) = \lambda_1 h_i^{(1)} - \frac{1}{\eta_2}\sum_j \frac{\partial K_2(h_i^{(1)}, h_j^{(1)})}{\partial h_i^{(1)}} h_j^{(2)^T} h_i^{(2)}, \ \forall i \\ \frac{1}{\eta_2}\sum_j h_j^{(2)} K_2(h_j^{(1)}, h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{1}{\eta_3}\sum_j \frac{\partial K_3(h_i^{(2)}, h_j^{(2)})}{\partial h_i^{(2)}} h_j^{(3)^T} h_i^{(3)}, \ \forall i \\ y_i = \frac{1}{\eta_3}\sum_j h_j^{(3)} K_3(h_j^{(2)}, h_i^{(2)}) + b + \lambda_3 h_i^{(3)}, \ \forall i \\ \sum_i h_i^{(3)} = 0. \end{cases} \tag{63}$$

Solving this set of nonlinear equations is computationally expensive. However, for the case of taking linear kernels K_2 and K_3 (and $K_{\text{lin}}, K_{2,\text{lin}}, K_{3,\text{lin}}$ denoting linear kernels) eq. (63) simplifies to

$$\begin{cases} \text{Level 1}: & \left(\frac{1}{\eta_1}[K_1(x_j, x_i)] + \frac{1}{\eta_2}[K_{\text{lin}}(h_j^{(2)}, h_i^{(2)})]\right) H_1^T = H_1^T \Lambda_1 \\ \text{Level 2}: & \left(\frac{1}{\eta_2}[K_{2,\text{lin}}(h_j^{(1)}, h_i^{(1)})] + \frac{1}{\eta_3}[K_{\text{lin}}(h_j^{(3)}, h_i^{(3)})]\right) H_2^T = H_2^T \Lambda_2 \\ \text{Level 3}: & \left[\begin{array}{c|c} \frac{1}{\eta_3}[K_{3,\text{lin}}(h_j^{(2)}, h_i^{(2)})] + \lambda_3 I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right] \left[\begin{array}{c} H_3^T \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} Y^T \\ \hline 0 \end{array}\right]. \end{cases} \tag{64}$$

Here we denote $H_1 = [h_1^{(1)} ... h_N^{(1)}]$, $H_2 = [h_1^{(2)} ... h_N^{(2)}]$, $H_3 = [h_1^{(3)} ... h_N^{(3)}]$. One sees that at level 1 and 2 a data fusion is taking place between K_1 and K_{lin} and between $K_{2,\text{lin}}$ and K_{lin}, where $\frac{1}{\eta_1}, \frac{1}{\eta_2}, \frac{1}{\eta_3}$ are specifying the relative weight given to each of these kernels. In this way one can choose for emphasizing or de-emphasizing the levels with respect to each other.

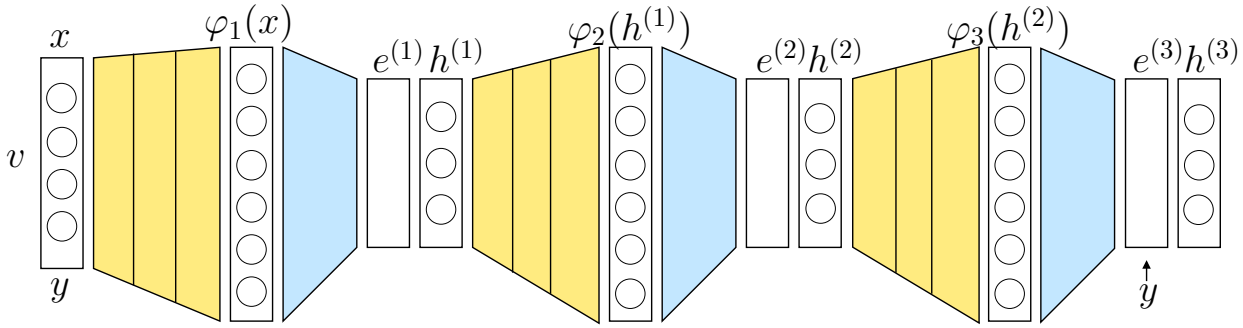

Figure 3: Example of a Deep restricted kernel machine (Deep RKM) consisting of 3 levels with kernel PCA in level 1 and 2 and LS-SVM regression in level 3.

4.2 Regression level followed by two kernel PCA levels

In this case we consider a Deep RKM architecture with the following three levels:

- Level 1 consists of LS-SVM regression with given input data x_i and output data y_i and is characterized by conjugated features $h_i^{(1)}$;
- Level 2 consists of kernel PCA by taking $h_i^{(1)}$ as input and is characterized by conjugated features $h_i^{(2)}$;
- Level 3 consists of kernel PCA by taking $h_i^{(2)}$ as input and is characterized by conjugated features $h_i^{(3)}$.

We look then for the stationary points of

$$J_{\text{deep}} = \underline{J}_1 + \overline{J}_2 + \overline{J}_3 \tag{65}$$

where the objective $J_{\text{deep}}(h_i^{(1)}, W_1, b, h_i^{(2)}, W_2, h_i^{(3)}, W_3)$ for the Deep RKM consists of the sum of the objectives of levels 1,2,3 given by $\underline{J}_1$, $\overline{J}_2$, $\overline{J}_3$, respectively. Deep RKM consists of coupling the RKMs.

This becomes

$$\begin{aligned} J_{\text{deep}} \;=\; & \sum_{i=1}^{N}(y_i^T - \varphi_1(x_i)^T W_1 - b^T)h_i^{(1)} - \frac{\lambda_1}{2}\sum_{i=1}^{N} {h_i^{(1)}}^T h_i^{(1)} + \frac{\eta_1}{2}\mathrm{Tr}(W_1^T W_1) \\ & -\sum_{i=1}^{N}\varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} + \frac{\lambda_2}{2}\sum_{i=1}^{N} {h_i^{(2)}}^T h_i^{(2)} + \frac{\eta_2}{2}\mathrm{Tr}(W_2^T W_2) \\ & -\sum_{i=1}^{N}\varphi_3(h_i^{(2)})^T W_3 h_i^{(3)} + \frac{\lambda_3}{2}\sum_{i=1}^{N} {h_i^{(3)}}^T h_i^{(3)} + \frac{\eta_3}{2}\mathrm{Tr}(W_3^T W_3) \end{aligned} \tag{66}$$

with $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{n_{f_1}}$, $W_1 \in \mathbb{R}^{n_{f_1} \times p}$, the level 2 part $\varphi_2 : \mathbb{R}^p \to \mathbb{R}^{n_{f_2}}$, $W_2 \in \mathbb{R}^{n_{f_2} \times s^{(2)}}$, and the level 3 part $\varphi_3 : \mathbb{R}^{s^{(2)}} \to \mathbb{R}^{n_{f_3}}$, $W_3 \in \mathbb{R}^{n_{f_3} \times s^{(3)}}$. Note that in J_{deep} the sum of the three inner pairing terms is similar to the energy in deep Boltzmann machines [35, 36] for the particular case of linear feature maps $\varphi_1, \varphi_2, \varphi_3$ and symmetric interaction terms. For the special case of linear feature maps one has

$$U_{\text{deep}} = -v^T \tilde{W}_1 h^{(1)} - {h^{(1)}}^T W_2 h^{(2)} - {h^{(2)}}^T W_3 h^{(3)} \tag{67}$$

which takes a same form as eq.(29) in [35], with $\tilde{W}_1$ defined in the sense of (19). The "U" in U_{deep} refers to the fact that the Deep RKM is unrestricted after coupling because of the hidden-to-hidden connections between layer 1 and 2 and between layer 2 and 3, while the uncoupled RKMs are restricted.

The stationary points of $J_{\text{deep}}(h_i^{(1)}, W_1, b, h_i^{(2)}, W_2, h_i^{(3)}, W_3)$ are given by

$$\left\{\begin{array}{rcl}
\frac{\partial J_{\text{deep}}}{\partial h_i^{(1)}} = 0 & \Rightarrow & y_i - W_1^T\varphi_1(x_i) - b = \lambda_1 h_i^{(1)} + \frac{\partial}{\partial h_i^{(1)}}[\varphi_2(h_i^{(1)})^T W_2 h_i^{(2)}],\ \forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_1} = 0 & \Rightarrow & W_1 = \frac{1}{\eta_1}\sum_i \varphi_1(x_i){h_i^{(1)}}^T \\
\frac{\partial J_{\text{deep}}}{\partial b} = 0 & \Rightarrow & \sum_i h_i^{(1)} = 0 \\
\frac{\partial J_{\text{deep}}}{\partial h_i^{(2)}} = 0 & \Rightarrow & W_2^T\varphi_2(h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{\partial}{\partial h_i^{(2)}}[\varphi_3(h_i^{(2)})^T W_3 h_i^{(3)}],\ \forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_2} = 0 & \Rightarrow & W_2 = \frac{1}{\eta_2}\sum_i \varphi_2(h_i^{(1)}){h_i^{(2)}}^T \\
\frac{\partial J_{\text{deep}}}{\partial h_i^{(3)}} = 0 & \Rightarrow & W_3^T\varphi_3(h_i^{(2)}) = \lambda_3 h_i^{(3)},\ \forall i \\
\frac{\partial J_{\text{deep}}}{\partial W_3} = 0 & \Rightarrow & W_3 = \frac{1}{\eta_3}\sum_i \varphi_3(h_i^{(2)}){h_i^{(3)}}^T.
\end{array}\right. \tag{68}$$

As predictive model for this Deep RKM case, we have

$$\begin{array}{rcl}
 & \nearrow & (P)_{\text{DeepRKM}}: \quad \hat{y} = W_1^T\varphi_1(x) + b \\
\mathcal{M} & & \\
 & \searrow & (D)_{\text{DeepRKM}}: \quad \hat{y} = \frac{1}{\eta_1}\sum_j h_j^{(1)} K_1(x_j, x) + b.
\end{array} \tag{69}$$

By elimination of W_1, W_2, W_3 one obtains the following set of nonlinear equations in the conjugated features $h_i^{(1)}, h_i^{(2)}, h_i^{(3)}$ and b:

$$\left\{\begin{array}{l}
y_i = \frac{1}{\eta_1}\sum_j h_j^{(1)} K_1(x_j, x_i) + b + \lambda_1 h_i^{(1)} + \frac{1}{\eta_2}\sum_j \frac{\partial K_2(h_i^{(1)}, h_j^{(1)})}{\partial h_i^{(1)}} {h_j^{(2)}}^T h_i^{(2)},\ \forall i \\
\sum_i h_i^{(1)} = 0 \\
\frac{1}{\eta_2}\sum_j h_j^{(2)} K_2(h_j^{(1)}, h_i^{(1)}) = \lambda_2 h_i^{(2)} - \frac{1}{\eta_3}\sum_j \frac{\partial K_3(h_i^{(2)}, h_j^{(2)})}{\partial h_i^{(2)}} {h_j^{(3)}}^T h_i^{(3)},\ \forall i \\
\frac{1}{\eta_3}\sum_j h_j^{(3)} K_3(h_j^{(2)}, h_i^{(2)}) = \lambda_3 h_i^{(3)},\ \forall i.
\end{array}\right. \tag{70}$$

When taking linear kernels K_2 and K_3, the set of nonlinear equations simplifies to

$$\begin{cases} \text{Level } 1: & \left[\begin{array}{c|c} \frac{1}{\eta_1}[K_1(x_j,x_i)] + \frac{1}{\eta_2}[K_{\text{lin}}(h_j^{(2)},h_i^{(2)})] + \lambda_1 I_N & 1_N \\ \hline 1_N^T & 0 \end{array}\right] \left[\begin{array}{c} H_1^T \\ \hline b^T \end{array}\right] = \left[\begin{array}{c} Y^T \\ \hline 0 \end{array}\right] \\ \text{Level } 2: & \left(\frac{1}{\eta_2}[K_{2,\text{lin}}(h_j^{(1)},h_i^{(1)})] + \frac{1}{\eta_3}[K_{\text{lin}}(h_j^{(3)},h_i^{(3)})]\right) H_2^T = H_2^T \Lambda_2 \\ \text{Level } 3: & \left(\frac{1}{\eta_3}[K_{3,\text{lin}}(h_j^{(2)},h_i^{(2)})]\right) H_3^T = H_3^T \Lambda_3 \end{cases} \tag{71}$$

with a similar data fusion interpretation as explained in the previous subsection.

5 Algorithms for Deep RKM

The characterization of the stationary points for the objective functions in the different deep RKM models will typically lead to solving large sets of nonlinear equations in the unknown variables, especially for large given data sets. Therefore in this section we will outline a number of different approaches and algorithms for working with the kernel-based models (either in the primal or in the dual). We also outline algorithms for training deep feedforward neural networks in a parametric way in the primal within the deep RKM setting. The algorithms proposed in subsections 5.2 and 5.3 are applicable also towards large data sets.

5.1 Levelwise solving for kernel-based models

For the case of linear kernels in level 2 and 3 in (64) and (71) we propose a heuristic algorithm that consists of levelwise solving linear systems and eigenvalue decompositions by alternating fixing different unknown variables.

For the case (71), in order to solve level 1 as a linear system one needs the input/output data $X = [x_1 ... x_N] \in \mathbb{R}^{d\times N}, Y = [y_1 ... y_N] \in \mathbb{R}^{p\times N}$, but one would also need the knowledge of $h_i^{(2)}$. Therefore an initialization phase is required. One can initialize $h_i^{(2)}$ as zero or at random at level 1, obtain then H_1 and propagate it to level 2. At level 2, after initializing H_3, one finds H_2, which is then propagated to level 3 where one computes H_3. After this *forward phase* one can go backward from level 3 to level 1 in a *backward phase*.

Schematically this gives the following heuristic algorithm:

Forward phase (level 1 $\rightarrow$ level 3)

$$\begin{array}{ll} & H_2, H_3 \text{ initialization} \\ \text{Level } 1: & H_1 := f_1(X, Y, H_2) \text{ (for (71))} \ \text{ or } \ H_1 := f_1(X, H_2) \text{ (for (64))} \\ \text{Level } 2: & H_2 := f_2(H_1, H_3) \\ \text{Level } 3: & H_3 := f_3(H_2) \text{ (for (71))} \ \text{ or } \ H_3 := f_1(Y, H_2) \text{ (for (64))} \end{array} \tag{72}$$

Backward phase (level 3 $\rightarrow$ level 1)

$$\begin{aligned}\text{Level 2}: &\quad H_2 := f_2(H_1, H_3)\\ \text{Level 1}: &\quad H_1 := f_1(X, Y, H_2)\,(\text{for (71)}) \;\text{ or }\; H_1 := f_1(X, H_2)\,(\text{for (64)}).\end{aligned}$$

One can further repeat the Forward and Backward phases a number of times, without the initialization step. Alternatively, one could also apply an algorithm with **Forward only phases** which can then be applied a number of times after each other.

5.2 Deep reduced set kernel-based models with estimation in primal

In the following approach, approximations $\tilde{W}_1, \tilde{W}_2, \tilde{W}_3$ are made to W_1, W_2, W_3

$$\begin{array}{rcl} W_1 & = & \frac{1}{\eta_1}\sum_{i=1}^{N}\varphi_1(x_i)h_i^{(1)^T} \simeq \tilde{W}_1 = \frac{1}{\eta_1}\sum_{j=1}^{M}\varphi_1(\tilde{x}_j)\tilde{h}_j^{(1)T} \\ \tilde{W}_2 & = & \frac{1}{\eta_2}\sum_{j=1}^{M}\varphi_2(\tilde{h}_j^{(1)})\tilde{h}_j^{(2)T} \\ \tilde{W}_3 & = & \frac{1}{\eta_3}\sum_{j=1}^{M}\varphi_2(\tilde{h}_j^{(2)})\tilde{h}_j^{(3)T} \end{array} \tag{73}$$

where a subset of the training data set $\{\tilde{x}_j\}_{j=1}^{M} \subset \{x_i\}_{i=1}^{N}$ is considered with $M \ll N$. This approximation corresponds to a reduced set technique in kernel methods [40]. In order to have a good representation of the data distribution one can take a fixed-size algorithm with subset selection according to quadratic Renyi entropy [50], or a random subset as a simpler scheme.

We proceed then with a primal estimation scheme by taking stabilization terms for the kernel PCA levels. In the case of two kernel PCA levels followed by LS-SVM regression we minimize the following objective

$$\begin{aligned}\min_{\tilde{h}_j^{(1)},\tilde{h}_j^{(2)},\tilde{h}_j^{(3)},b,\Lambda_1,\Lambda_2} J_{\text{deep,P}_{\text{stab}}} = &-\frac{1}{2}\sum_{j=1}^{M} e_j^{(1)^T}\Lambda_1^{-1}e_j^{(1)} + \frac{\eta_1}{2}\text{Tr}(\tilde{W}_1^T\tilde{W}_1) - \tfrac{1}{2}\textstyle\sum_{j=1}^{M} e_j^{(2)^T}\Lambda_2^{-1}e_j^{(2)} \\ &+\frac{\eta_2}{2}\text{Tr}(\tilde{W}_2^T\tilde{W}_2) + \frac{1}{2\lambda_3}\sum_{j=1}^{M} e_j^{(3)^T}e_j^{(3)} + \frac{\eta_3}{2}\text{Tr}(\tilde{W}_3^T\tilde{W}_3) \\ &+\frac{1}{2}c_{\text{stab}}(-\frac{1}{2}\sum_{j=1}^{M} e_j^{(1)^T}\Lambda_1^{-1}e_j^{(1)} + \frac{\eta_1}{2}\text{Tr}(\tilde{W}_1^T\tilde{W}_1))^2 \\ &+\frac{1}{2}c_{\text{stab}}(-\frac{1}{2}\sum_{j=1}^{M} e_j^{(2)^T}\Lambda_2^{-1}e_j^{(2)} + \frac{\eta_2}{2}\text{Tr}(\tilde{W}_2^T\tilde{W}_2))^2.\end{aligned} \tag{74}$$

The predictive model becomes then

$$\begin{aligned} \hat{e}^{(1)} &= \frac{1}{\eta_1}\sum_{j=1}^{M}\tilde{h}_j^{(1)}K_1(\tilde{x}_j,x) \\ \hat{e}^{(2)} &= \frac{1}{\eta_2}\sum_{j=1}^{M}\tilde{h}_j^{(2)}K_2(\tilde{h}_j^{(1)},\Lambda_1^{-1}\hat{e}^{(1)}) \\ \hat{y} &= \frac{1}{\eta_3}\sum_{j=1}^{M}\tilde{h}_j^{(3)}K_3(\tilde{h}_j^{(2)},\Lambda_2^{-1}\hat{e}^{(2)})+b. \end{aligned} \tag{75}$$

The number of unknowns in this case is $M\times(s^{(1)}+s^{(2)}+p)+s^{(1)}+s^{(2)}+1$. Alternatively, instead of the regularization terms $\mathrm{Tr}(\tilde{W}_l^T\tilde{W}_l)$ one could also take $\mathrm{Tr}(\tilde{H}^{(l)}\tilde{H}^{(l)T})$ where $\tilde{H}^{(l)}=[\tilde{h}_1^{(l)}...\tilde{h}_M^{(l)}]$ for $l=1,2,3$.

One can additionally also maximize $\mathrm{Tr}(\Lambda_1)+\mathrm{Tr}(\Lambda_2)$ by adding a term $-c_0(\mathrm{Tr}(\Lambda_1)+\mathrm{Tr}(\Lambda_2))$ to the objective (74), with c_0 a positive constant. Note that the components of $\tilde{H}^{(1)},\tilde{H}^{(2)}$ in level 1 and 2 don't possess an orthogonality property, unless this is imposed as additional constraints to the objective function.

5.3 Training deep feedforward neural networks within the Deep RKM framework

For training of deep feedforward neural networks within this deep RKM setting one minimizes $J_{\mathrm{deep,P_{stab}}}$ in the unknown interconnection matrices of the different levels. In case one takes one hidden layer per level, the following objective is minimized

$$\begin{aligned} \min_{W_{1,2,3},U_{1,2,3},\beta_{1,2,3},b,\Lambda_1,\Lambda_2} J_{\mathrm{deep,P_{stab}}} &= -\frac{1}{2}\sum_{j=1}^{M}{e_j^{(1)}}^T\Lambda_1^{-1}e_j^{(1)}+\frac{\eta_1}{2}\mathrm{Tr}(W_1^TW_1)-\tfrac{1}{2}\textstyle\sum_{j=1}^{M}{e_j^{(2)}}^T\Lambda_2^{-1}e_j^{(2)} \\ &+\frac{\eta_2}{2}\mathrm{Tr}(W_2^TW_2)+\frac{1}{2\lambda_3}\sum_{j=1}^{M}{e_j^{(3)}}^Te_j^{(3)}+\frac{\eta_3}{2}\mathrm{Tr}(W_3^TW_3) \\ &+\frac{1}{2}c_{\mathrm{stab}}(-\frac{1}{2}\sum_{j=1}^{M}{e_j^{(1)}}^T\Lambda_1^{-1}e_j^{(1)}+\frac{\eta_1}{2}\mathrm{Tr}(W_1^TW_1))^2 \\ &+\frac{1}{2}c_{\mathrm{stab}}(-\frac{1}{2}\sum_{j=1}^{M}{e_j^{(2)}}^T\Lambda_2^{-1}e_j^{(2)}+\frac{\eta_2}{2}\mathrm{Tr}(W_2^TW_2))^2 \end{aligned} \tag{76}$$

for the model

$$\begin{aligned} \hat{e}^{(1)} &= W_1^T\sigma(U_1x+\beta_1) \\ \hat{e}^{(2)} &= W_2^T\sigma(U_2\Lambda_1^{-1}\hat{e}^{(1)}+\beta_2) \\ \hat{y} &= W_3^T\sigma(U_3\Lambda_2^{-1}\hat{e}^{(2)}+\beta_3)+b. \end{aligned} \tag{77}$$

Alternatively, one can take additional nonlinearities on $\Lambda_1^{-1}\hat{e}^{(1)},\Lambda_2^{-1}\hat{e}^{(2)}$ which results in the model $\hat{e}^{(1)}=W_1^T\sigma(U_1x+\beta_1)$, $\hat{e}^{(2)}=W_2^T\sigma(U_2\sigma(\Lambda_1^{-1}\hat{e}^{(1)})+\beta_2)$, $\hat{y}=W_3^T\sigma(U_3\sigma(\ \Lambda_2^{-1}\ \hat{e}^{(2)})+\beta_3)+b$. The number of unknowns is $n_{h_1}\times(s^{(1)}+d+1)+n_{h_2}\times(s^{(2)}+s^{(1)}+1)+n_{h_3}\times(p+s^{(2)}+1)+s^{(1)}+s^{(2)}+1$, where $n_{h_{1,2,3}}$ denote the number of hidden units.

In order to further reduce the number of unknowns, and partially inspired by convolutional operations in convolutional neural networks [23], we further consider also the case where U_1 and U_2 are Toeplitz matrices. For a matrix $U \in \mathbb{R}^{n_1 \times n_2}$ the number of unknowns is reduced then from $n_1 n_2$ to $n_1 + n_2 - 1$.

6 Numerical examples

6.1 Two kernel PCA levels followed by regression level: examples

We define the following models and methods for comparison:

- $[\mathcal{M}_1]$: *Deep reduced set kernel-based models* (with RBF kernel) with estimation in the primal according to (74) with the following choices:
  - $[\mathcal{M}_{1,a}]$: with additional term $-c_0(\mathrm{Tr}(\Lambda_1)+\mathrm{Tr}(\Lambda_2))$ and $\mathrm{Tr}(\tilde{H}^{(l)}\tilde{H}^{(l)T})$ $(l = 1, 2, 3)$ regularization terms.
  - $[\mathcal{M}_{1,b}]$: without additional term $-c_0(\mathrm{Tr}(\Lambda_1) + \mathrm{Tr}(\Lambda_2))$
  - $[\mathcal{M}_{1,c}]$: with objective function $\frac{1}{2\lambda_3}\sum_{j=1}^{M} e_j^{(3)^T} e_j^{(3)} + \frac{\eta_3}{2}\mathrm{Tr}(W_3^T W_3)$ i.e. only the level 3 regression objective.
- $[\mathcal{M}_2]$: *Deep feedforward neural networks* with estimation in the primal according to (76) with the same choices in $[\mathcal{M}_{2,a}]$, $[\mathcal{M}_{2,b}]$, $[\mathcal{M}_{2,c}]$ as above in $[\mathcal{M}_1]$. In the model $[\mathcal{M}_{2,b,T}]$ Toeplitz matrices are taken for the U matrices in all levels, except for the last level.

We test and compare the proposed algorithms on a number of UCI datasets: Pima indians diabetes (pid) $[d = 8, p = 1, N = 400, N_{\mathrm{val}} = 112, N_{\mathrm{test}} = 256]$, Bupa liver disorder (bld) $[d = 6, p = 1, N = 170, N_{\mathrm{val}} = 60, N_{\mathrm{test}} = 115]$, Johns Hopkins university ionosphere (ion) $[d = 34, p = 1, N = 170, N_{\mathrm{val}} = 64, N_{\mathrm{test}} = 117]$, adult (adu) $[d = 14, p = 1, N = 22000, N_{\mathrm{val}} = 11000, N_{\mathrm{test}} = 12222]$ datasets, where the number of inputs (d), outputs (p), training (N), validation (N_{val}) and test data (N_{test}) are indicated. These numbers correspond to previous benchmarking studies in [55]. In Table 1 *bestbmark* indicates the best result obtained in the benchmarking study of [55] from different classifiers including SVM and LS-SVM classifiers with linear, polynomial and RBF kernel, linear and quadratic discriminant analysis, decision tree algorithm C4.5, logistic regression, one-rule classifier, instance based learners and Naive Bayes.

The tuning parameters, selected at the validation level, are

- pid: For $\mathcal{M}_1$: $s^{(1),(2),(3)} = 2, 2, p$; $M = 20$; $\lambda_3 = 10^{-2}$; $c_{\text{stab}} = 10$ ($\mathcal{M}_{1,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{1,a}$). For $\mathcal{M}_2$: $s^{(1),(2),(3)} = 4, 2, p$; $n_{h_{1,2,3}} = 3, 3, 3$; $\lambda_3 = 10^{-2}$; $c_{\text{stab}} = 10$ ($\mathcal{M}_{2,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{2,a}$). For $\mathcal{M}_{2,b,T}$: $s^{(1),(2),(3)} = 4, 4, p$; $n_{h_{1,2,3}} = 3, 3, 3$; $\lambda_3 = 10^{-2}$; $c_{\text{stab}} = 1$.

- bld: For $\mathcal{M}_1$: $s^{(1),(2),(3)} = 3, 2, p$; $M = 20$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 100$ ($\mathcal{M}_{1,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{1,a}$). For $\mathcal{M}_2$: $s^{(1),(2),(3)} = 4, 2, p$; $n_{h_{1,2,3}} = 3, 3, 5$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 1000$ ($\mathcal{M}_{2,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{2,a}$) For $\mathcal{M}_{2,b,T}$: $s^{(1),(2),(3)} = 4, 2, p$; $n_{h_{1,2,3}} = 3, 3, 5$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 1000$.

- ion: For $\mathcal{M}_1$: $s^{(1),(2),(3)} = 3, 2, p$; $M = 30$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 100$ ($\mathcal{M}_{1,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{1,a}$). For $\mathcal{M}_2$: $s^{(1),(2),(3)} = 3, 3, p$; $n_{h_{1,2,3}} = 3, 3, 3$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 1000$ ($\mathcal{M}_{2,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{2,a}$) For $\mathcal{M}_{2,b,T}$: $s^{(1),(2),(3)} = 3, 3, p$; $n_{h_{1,2,3}} = 3, 3, 3$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 1000$.

- adu: For $\mathcal{M}_1$: $s^{(1),(2),(3)} = 20, 10, p$; $M = 15$; $\lambda_3 = 10^{-3}$; $c_{\text{stab}} = 10^{-4}$ ($\mathcal{M}_{1,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{1,a}$). For $\mathcal{M}_2$: $s^{(1),(2),(3)} = 5, 2, p$; $n_{h_{1,2,3}} = 10, 5, 3$; $\lambda_3 = 10^{-7}$; $c_{\text{stab}} = 0.1$ ($\mathcal{M}_{2,a,b}$); $c_0 = 0.1$ ($\mathcal{M}_{2,a}$). For $\mathcal{M}_{2,b,T}$: $s^{(1),(2),(3)} = 5, 2, p$; $n_{h_{1,2,3}} = 10, 5, 3$; $\lambda_3 = 10^{-7}$; $c_{\text{stab}} = 10$, $\eta_{1,2,3} = 1, 1, 1$.

The other tuning parameters were selected as $\eta_{1,2,3} = 1, 1, 1$ for pid, bld, ion and $\eta_{1,2} = 10^3$, $\eta_3 = 10^{-3}$ for adu, unless specified differently above. In the $\mathcal{M}_1$ and $\mathcal{M}_2$ models the $\tilde{H}^{(1),(2),(3)}$ matrices and the interconnection matrices were initialized at random according to a normal distribution with zero mean and standard deviation 0.1 (100, 20, 10 and 3 initializations for pid, bld, ion, adu, respectively), the diagonal matrices $\Lambda_{1,2}$ by the identity matrix and $\sigma_{1,2,3} = 1$ for the RBF kernel models in $\mathcal{M}_1$. For the training a quasi-Newton method was used with fminunc in Matlab.

General observations from the experiments shown in Table 1 are that

- having the additional terms with kernel PCA objectives in level 1 and 2, as opposed to the level 3 objective only, gives improved results on all tried datasets.

- the best selected value for c_{stab} varies among the datasets. In case this value is large then the value of the objective function terms related to the kernel PCA parts are close to zero.

- the use of Toeplitz matrices for the U matrices in the deep feeforward neural networks leads to competitive performance results and greatly reduces the number of unknowns.

In Figure 4 the evolution of the objective function (in logarithmic scale) during training on the ion data set, for different values of c_{stab} and in comparison with a level 3 objective function only.

	pid	bld	ion	adu
$\mathcal{M}_{1,a}$	19.53 [20.02(1.53)]	26.09 [30.96(3.34)]	**0** [0.68(1.60)]	16.99 [17.46(0.65)]
$\mathcal{M}_{1,b}$	**18.75** [19.39(0.89)]	**25.22** [31.48(4.11)]	**0** [5.38(12.0)]	17.08 [17.48(0.56)]
$\mathcal{M}_{1,c}$	21.88 [24.73(5.91)]	28.69 [32.39(3.48)]	**0** [8.21(6.07)]	17.83 [21.21(4.78)]
$\mathcal{M}_{2,a}$	21.09 [20.20(1.51)]	27.83 [28.86(2.83)]	1.71 [5.68(2.22)]	15.07 [15.15(0.15)]
$\mathcal{M}_{2,b}$	**18.75** [20.33(2.75)]	28.69 [28.38(2.80)]	10.23 [6.92(3.69)]	14.91 [15.08 (0.15)]
$\mathcal{M}_{2,b,T}$	19.03 [19.16(1.10)]	26.08 [27.74(9.40)]	6.83 [6.50(8.31)]	15.71 [15.97(0.07)]
$\mathcal{M}_{2,c}$	24.61 [22.34(1.95)]	32.17 [27.61(3.69)]	3.42 [9.66(6.74)]	15.21 [15.19(0.08)]
bestbmark	22.7(2.2)	29.6(3.7)	4.0(2.1)	14.4(0.3)

Table 1: Comparison of test error (%) of models $\mathcal{M}_1$ and $\mathcal{M}_2$ on UCI data sets. Shown is first the test error corresponding to the selected model with minimal validation error from the different random initializations. Between square brackets the mean and standard deviation of the test errors related to all initializations is shown. The lowest test error is shown in bold.

6.2 Regression level followed by two kernel PCA levels: examples

Regression example on synthetic data set: In this example we compare a basic LS-SVM regression with Deep RKM consisting of 3 levels with LSSVM + KPCA + KPCA, where a Gaussian RBF kernel $K(x_i, x_j) = \exp(-\|x_i - x_j\|_2^2/\sigma^2)$ is used in the LSSVM level and linear kernels in the KPCA levels. Training, validation and test data sets are generated from the following true underlying function

$$f(x) = \sin(0.3x) + \cos(0.5x) + \sin(2x) \tag{78}$$

where zero mean Gaussian noise with standard deviation 0.1, 0.5, 1 and 2 is added to the function values for the different data sets. In this example we have a single input and single output $d = p = 1$. Training data (with noise) are generated in the interval $[-10, 10]$ with steps 0.1, validation data (with noise) in $[-9.77, 9.87]$ with steps 0.11 and test data (noiseless) in $[-9.99, 9.99]$ with steps 0.07. In the experiments 100 realizations for the noise are made, for which the mean and standard deviation of the results are shown in Table 3. The tuning parameters are selected based on the validation set, which are σ, γ for the RBF kernel in the basic LS-SVM model and σ, λ_1, η_2, η_3 ($\eta_1 = 1$ has been chosen) for the complete Deep RKM. The number of forward-backward passes in the Deep RKM is chosen equal to 10. For Deep RKM we take the following two choices for the number of components $s^{(2)}, s^{(3)}$ in the KPCA levels: 1 and 1, 7 and 2 for level 2 and level 3 respectively. For Deep RKM the optimal values for $\frac{1}{\eta_2}, \frac{1}{\eta_3}$ are 10^5 which means that the level 2 and 3 kernel PCA levels receive higher weight in the kernel fusion terms. As

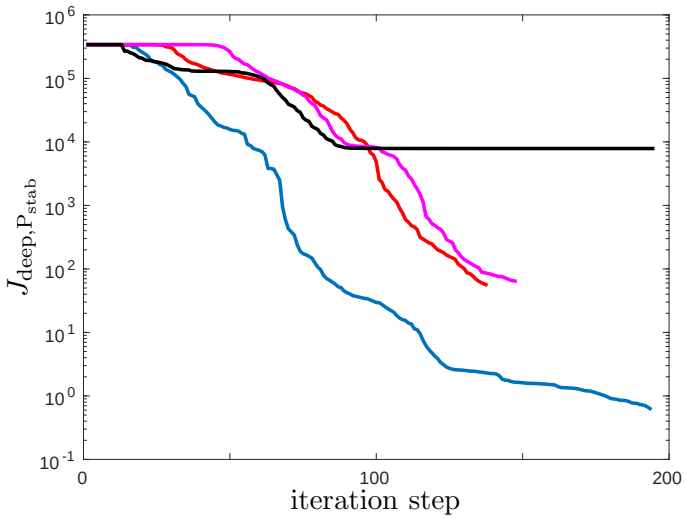

Figure 4: Illustration of the evolution of the objective function (logarithmic scale) during training on the ion data set. Shown are training curves for the model $\mathcal{M}_{2,a}$ for different choices of c_{stab} (equal to 1, 10, 100 in blue, red, magenta color, respectively) in comparison with $\mathcal{M}_{2,c}$ (level 3 objective only, in black color), for the same initialization.

seen in this table Deep RKM improves over the basic LS-SVM regression in this example. The optimal values for (σ, λ_1) are $(1, 0.001)$ for noise level 0.1, $(1, 0.01)$ for noise level 0.5, $(1, 0.4)$ for noise level 1. $(1, 1)$ for noise level 2.

noise	basic	deep (1+1)	deep (7+2)
0.1	$0.0019 \pm 4.3\,10^{-4}$	$0.0018 \pm 4.2\,10^{-4}$	$0.0019 \pm 4.4\,10^{-4}$
0.5	0.0403 ± 0.0098	0.0374 ± 0.0403	0.0397 ± 0.0089
1	0.1037 ± 0.0289	0.0934 ± 0.0269	0.0994 ± 0.0301
2	0.3368 ± 0.0992	0.2902 ± 0.0875	0.3080 ± 0.0954

Table 2: Comparison between basic LS-SVM regression and deep RKM on the synthetic data set, for different noise levels.

Multiclass example USPS: In this example the USPS Handwritten Digits data set is taken, with Matlab data set as prepared at http://www.cs.nyu.edu/∼roweis/data.html. It contains 8-bit grayscale images of digit "0" through "9" with 1100 examples of each class. These data are used without additional scaling or preprocessing. We compare a basic LS-SVM model (with primal representation $\hat{y} = W^T\varphi(x) + b$ and $W \in \mathbb{R}^{n_f \times p}, b \in \mathbb{R}^p$ with $p = 10$, i.e. one output per class, and RBF kernel) with Deep RKM consisting of LSSVM + KPCA

+ KPCA with RBF kernel in level 1 and linear kernels in level 2 and 3 (with number of selected components $s^{(2)}$, $s^{(3)}$ in level 2 and 3). In level 1 of Deep RKM the same type of model is taken as in the basic LS-SVM model. In this way we intend to study the effect of the two additional KPCA layers. The dimensionality of the input data is $d = 256$. Two training set sizes were taken ($N = 2000$ and $N = 4000$ data points, i.e. 200 and 400 examples per class), 2000 data points (200 per class) for validation and 5000 data (500 per class) for testing. The tuning parameters are selected based on the validation set, which are σ, γ for the RBF kernel in the basic LS-SVM model and σ, λ_1, η_2, η_3 ($\eta_1 = 1$ has been chosen) for Deep RKM. The number of forward-backward passes in the Deep RKM is chosen equal to 2. The results are shown for the case of 2000 training data in Figure 5, showing the results on training, validation and test data with the predicted class labels and the predicted output values for the different classes. For the case $N = 2000$ the selected values were $\sigma^2 = 45$, $s^{(2)} = 10$, $s^{(3)} = 1$, $\lambda_1 = 10^{-6}$, $\frac{1}{\eta_2} = \frac{1}{\eta_3} = 10^6$. The misclassification error on the test data set is 3.18% for the Deep RKM, while 3.26% for the basic LS-SVM (with $\sigma^2 = 45$ and $\gamma = 1/\lambda_1$). For the case $N = 4000$ the selected values were $\sigma^2 = 45$, $s^{(2)} = 1$, $s^{(3)} = 1$, $\lambda_1 = 10^{-6}$, $\frac{1}{\eta_2} = \frac{1}{\eta_3} = 10^6$. The misclassification error on the test data set is 2.12% for the Deep RKM, while 2.14% for the basic LS-SVM (with $\sigma^2 = 45$ and $\gamma = 1/\lambda_1$). This illustrates that for Deep RKM the level 2 and 3 are given high relative importance through the selection of large $\frac{1}{\eta_2}$, $\frac{1}{\eta_3}$ values.

Multiclass example MNIST: The data set, which is used without additional scaling or preprocessing, is taken from http://www.cs.nyu.edu/∼roweis/data.html. The dimensionality of the input data is $d = 784$ (images of size 28x28 for each of the 10 classes). In this case we take an ensemble approach where the training set ($N = 50000$ with 10 classes) has been partitioned into small non-overlapping subsets of size 50 (i.e. 5 data points per class). The choice for this subset size was resulting from taking the last 10000 point of this data set as validation data with use of 40000 data for training in that case. Other tuning parameters were selected in a similar way. The 1000 resulting submodels have been linearly combined after applying the tanh function to their outputs. The linear combination is determined by solving an overdetermined linear system with ridge regression, following a similar approach as discussed in section 6.4 of [50]. For the submodels Deep RKMs consisting of LSSVM + KPCA + KPCA with RBF kernel in level 1 and linear kernels in level 2 and 3, are taken. The selected tuning parameters are $\sigma^2 = 49$, $s^{(2)} = 1$, $s^{(3)} = 1$, $\lambda_1 = 10^{-6}$, $\eta_1 = 1$, $\frac{1}{\eta_2} = \frac{1}{\eta_3} = 10^{-6}$. The number of forward-backward passes in the Deep RKM is chosen equal to 2. The training data set has been extended with another 50000 training data consisting of the same data points but corrupted with noise (random perturbations with zero mean and standard deviation 0.5, truncated to the range [0,1]), which is related to the method with random perturbations in [21]. The misclassification error on the test data set (10000 data points) is 1.28%, which is comparable in performance to deep belief networks (1.2%) and in between the reported test performances of deep Boltzmann machines (0.95, 1.01%) and

SVM with Gaussian kernel (1.4%) [35] (see http://yann.lecun.com/exdb/mnist/ for an overview and comparison of performances obtained by different methods).

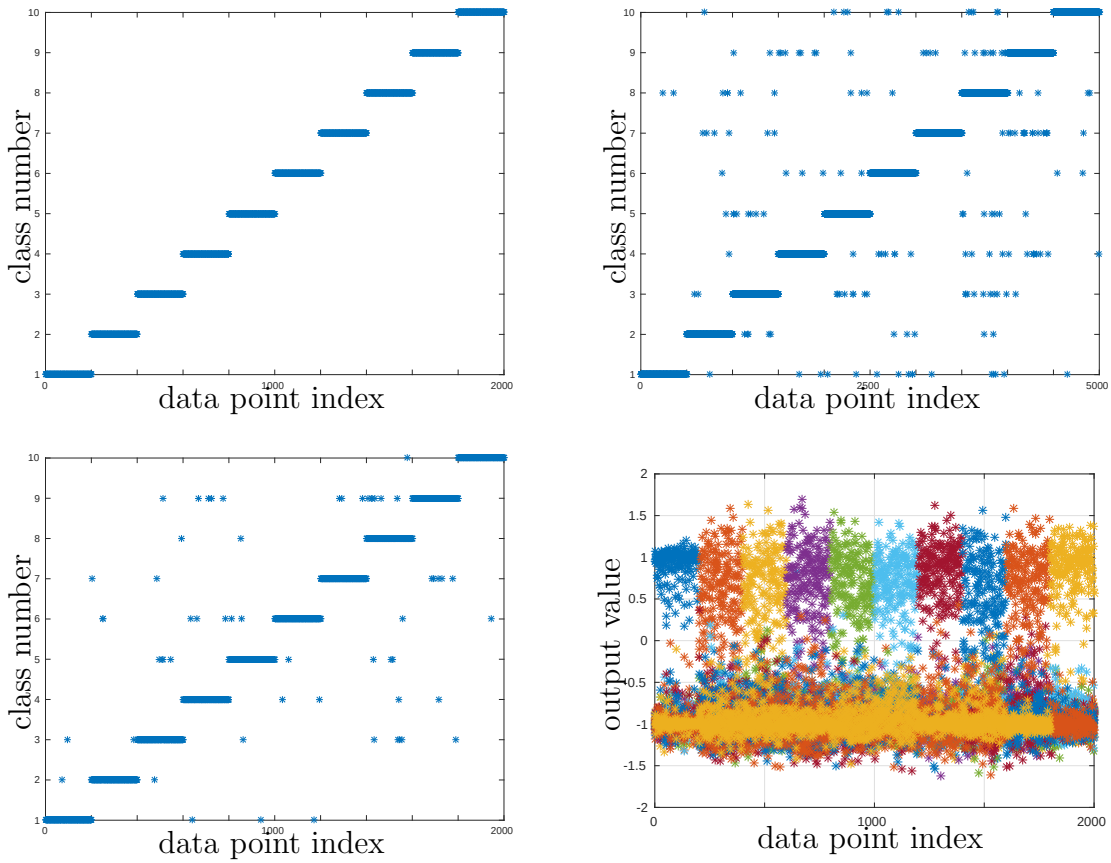

Figure 5: Deep RKM on USPS Handwritten Digits data set: (left-top) training data results (2000 data), (left-bottom) validation data results (2000 data), (right-top) test data results (5000 data), (right-bottom) output values for the 10 different classes on the validation set.

7 Conclusions

In this paper a theory of deep restricted kernel machines has been proposed. It is obtained by introducing a notion of conjugate feature duality where the conjugate features correspond to hidden features. Existing kernel machines such as least squares support vector machines for classification and regression, kernel PCA, matrix SVD and Parzen-type models are considered as building blocks within a Deep RKM, and are characterized through the conjugate feature duality. By means of the inner pairing one achieves a link with the energy expression of restricted Boltzmann machines, though with continuous variables in

a nonprobabilistic setting. It also provides an interpretation of visible and hidden units. Therefore this paper connects on the one hand to deep learning methods, while on the other hand also to least squares support vector machines and kernel methods. In this way the insights and foundations achieved in these different research areas could possibly mutually reinforce each other in the future. Many future work is possible in different directions, including e.g. efficient methods and implementations for big data, the extension to other loss functions and regularization schemes, treating multimodal data, different coupling schemes, models for clustering and semisupervised learning, and several others.

Acknowledgments

The research leading to these results has received funding from the European Research Council (FP7/2007-2013) / ERC AdG A-DATADRIVE-B (290923) under the European Union's Seventh Framework Programme. This paper reflects only the authors' views, the Union is not liable for any use that may be made of the contained information; Research Council KUL: GOA/10/09 MaNet, CoE PFV/10/002 (OPTEC), BIL12/11T; PhD/Postdoc grants; Flemish Government: FWO: PhD/Postdoc grants, projects: G0A4917N (Deep restricted kernel machines), G.0377.12 (Structured systems), G.088114N (Tensor based data similarity); IWT: PhD/Postdoc grants, projects: SBO POM (100031); iMinds Medical Information Technologies SBO 2014; Belgian Federal Science Policy Office: IUAP P7/19 (DYSCO, Dynamical systems, control and optimization, 2012-2017).

References

[1] Ackley, D.H., Hinton, G.E., Sejnowski, T.J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169.

[2] Alzate, C., Suykens, J.A.K. (2010). Multiway Spectral Clustering with Out-of-Sample Extensions through Weighted Kernel PCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(2), 335–347.

[3] Bengio, Y., Courville, A., Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 1798–1828.

[4] Bengio, Y. (2009). *Learning Deep Architectures for AI*, Now.

[5] Boser, B.E., Guyon, I.M., Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers, Proceedings COLT '92 Proceedings of the fifth annual workshop on Computational learning theory, 144–152.

[6] Boyd, S., Vandenberghe, L. (2004). *Convex Optimization*, Cambridge University Press.

[7] Chen, L.-C., Schwing, A.G., Yuille, A.L., Urtasun, R. (2015). Learning deep structured models, In International Conference on Machine Learning (ICML), Lille, France, July 2015.

[8] Cho, Y., Saul, L.K. (2009). Kernel Methods for Deep Learning, Advances in Neural Information Processing Systems 22 (NIPS 2009).

[9] Cortes, C., Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

[10] Damianou, A.C., Lawrence, N.D. (2013). Deep Gaussian Processes, Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS 2013)

[11] De Wilde, Ph. (1993). Class of Hamiltonian neural networks. *Phys. Rev. E*, *47*, 1392–1396.

[12] Fischer, A., Igel, C. (2014). Training Restricted Boltzmann Machines: An Introduction. *Pattern Recognition*, *47*, 25–39.

[13] Goldstein, H., Poole, C., Safko, J. (2002). *Classical Mechanics*, Addison-Wesley.

[14] Golub, G.H., Van Loan, C.F. (1989). *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD.

[15] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*, MIT Press.

[16] Hertz, J., Krogh, A., Palmer, R.G. (1991). *Introduction to the theory of neural computation*, Lecture notes Vol I, Santa Fe institute, Addison-Wesley.

[17] Hinton, G.E. (2005). What kind of graphical model is the brain? In Proc. 19th International Joint Conference on Artificial intelligence, 1765–1775.

[18] Hinton, G.E., Osindero, S., Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

[19] Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, *79*, 2554–2558.

[20] Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A. (2015). Deep Structured Output Learning for Unconstrained Text Recognition, ICLR 2015.

[21] Kurakin, A., Goodfellow, I., Bengio, S. (2016) Adversarial Machine Learning at Scale, arXiv:1611.01236 [cs.CV].

[22] Larochelle, H., Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. Proceedings of the 25th international conference on Machine learning (ICML 2008) p. 536.

[23] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition, Proceedings of the IEEE, *86*, 11, 2278–2324.

[24] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.-J. (2006). A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Schölkopf, B. and Smola, A. and Taskar, B. (Eds), *Predicting Structured Data*, MIT Press.

[25] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

[26] Lee, H., Grosse, R., Ranganath, R., Ng, A.Y. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, ICML 2009 Proceedings of the 26th Annual International Conference on Machine Learning, 609–616.

[27] Mairal, J., Koniusz, P., Harchaoui, Z., Schmid, C. (2014). Convolutional Kernel Networks. Adv. Neural Information Processing Systems (NIPS)

[28] Mall R., Langone R., Suykens, J.A.K. (2014). Multilevel Hierarchical Kernel Spectral Clustering for Real-Life Large Scale Complex Networks. *PLOS One*, *9*(6), e99966.

[29] Petersen, K.B., Pedersen, M.S. (2012). The Matrix Cookbook, Technical University of Denmark.

[30] Poggio, T., Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1481–1497.

[31] Rasmussen, C.E., Williams, C. (2006). *Gaussian Processes for Machine Learning*, MIT Press.

[32] Rockafellar, R.T. (1987). Conjugate Duality and Optimization, CBMS-NSF Regional Conference Series in Applied Mathematics.

[33] Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

[34] Salakhutdinov, R., Hinton, G.E. (2007). Using deep belief nets to learn covariance kernels for Gaussian processes, Advances in Neural Information Processing Systems 20 (NIPS 2007)

[35] Salakhutdinov, R. (2015). Learning Deep Generative Models. *Annu. Rev. Stat. Appl.*, *2*, 361–385.

[36] Salakhutdinov, R., Hinton, G.E. (2009). Deep Boltzmann Machines, International conference on artificial intelligence and statistics, 448–455.

[37] Saunders, C., Gammerman, A., Vovk, V. (1998). Ridge regression learning algorithm in dual variables, Proc. of the 15th Int. Conf. on Machine Learning (ICML-98), Madison-Wisconsin, 515–521.

[38] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

[39] Schölkopf, B., Smola, A., Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

[40] Schölkopf, B., Mika, S., Burges, C.C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.J (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*(5), 1000-1017.

[41] Schölkopf, B., Smola, A. (2002). *Learning with Kernels*, MIT Press, Cambridge, MA.

[42] Schwing, A.G., Urtasun, R. (2015). Fully Connected Deep Structured Networks, arXiv:1503.02351.

[43] Smale, S., Rosasco, L., Bouvrie, J., Caponnetto, A., Poggio, T. (2010). Mathematics of the neural response. *Foundations of Computational Mathematics*, *10*(1), 67–91.

[44] Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory. Chapter 6 in D.E. Rumelhart and J.L. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations. McGraw-Hill, New York.

[45] Srivastava, N., Salakhutdinov, R. (2014). Multimodal Learning with Deep Boltzmann Machines. *Journal of Machine Learning Research*, *15*, 2949–2980.

[46] Stewart, G.W. (1993). On the early history of the singular value decomposition. *SIAM Review*, *35*(4), 551–566

[47] Suykens, J.A.K., Vandewalle, J., De Moor, B. (1995). *Artificial Neural Networks for Modeling and Control of Non-Linear Systems*, Springer.

[48] Suykens, J.A.K., Vandewalle, J. (1999a). Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, *10*(4), 907–911.

[49] Suykens, J.A.K., Vandewalle, J. (1999b). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*(3), 293–300

[50] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J. (2002). *Least Squares Support Vector Machines*, World Scientific, Singapore.

[51] Suykens, J.A.K., Van Gestel, T., Vandewalle, J., De Moor, B. (2003). A support vector machine formulation to PCA analysis and its kernel version. *IEEE Transactions on Neural Networks*, *14*(2), 447–450.

[52] Suykens, J.A.K., Alzate, C., Pelckmans, K. (2010). Primal and dual model representations in kernel-based learning. *Statistics Surveys*, *4*, 148–183.

[53] Suykens, J.A.K. (2013). Generating quantum-measurement probabilities from an optimality principle. *Physical Review A*, *87*(5), 052134.

[54] Suykens, J.A.K. (2016). SVD revisited: a new variational principle, compatible feature maps and nonlinear extensions. *Applied and Computational Harmonic Analysis*, *40*(3), 600–609.

[55] Van Gestel, T., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., De Moor, B., Vandewalle, J. (2004). Benchmarking Least Squares Support Vector Machine Classifiers. *Machine Learning*, *54*(1), 5-32.

[56] Vapnik, V. (1998). *Statistical Learning Theory*, John Wiley & Sons, New York.

[57] Wahba, G. (1990). *Spline Models for Observational Data*, SIAM.

[58] Welling, M., Rosen-Zvi, M., Hinton, G.E. (2004). Exponential family harmoniums with an application to information retrieval, Advances in neural information processing systems.

[59] Wiering, M.A., Schomaker, L.R.B. (2014). Multi-Layer Support Vector Machines, Chapter 20 in: *Regularization, Optimization, Kernels, and Support Vector Machines*, J.A.K. Suykens, M. Signoretto, A. Argyriou (Eds.), Chapman & Hall/CRC, Machine Learning & Pattern Recognition Series, Boca Raton US, 457–476.

[60] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S. (2015). Conditional Random Fields as Recurrent Neural Networks, IEEE ICCV 2015.

Appendix

We explain here the role of the stabilization term in kernel PCA, as a modification to (15). In this case the objective function in the primal is

$$\begin{array}{ll} \min\limits_{w,e_i} & \frac{1}{2}w^Tw - \frac{\gamma}{2}\sum\limits_i e_i^2 + \frac{c_{\text{stab}}}{2}(\frac{1}{2}w^Tw - \frac{\gamma}{2}\sum\limits_i e_i^2)^2 \\ \text{subject to} & e_i = w^T\varphi(x_i),\ i = 1, ..., N. \end{array} \tag{79}$$

Denoting $J_0 = \frac{1}{2}w^Tw - \frac{\gamma}{2}\sum_i e_i^2$ the Lagrangian is $\mathcal{L} = J_0 + \frac{c_{\text{stab}}}{2}J_0^2 + \sum_i \alpha_i(e_i - w^T\varphi(x_i))$ from which it follows that

$$\left\{ \begin{array}{rcl} \dfrac{\partial \mathcal{L}}{\partial w} & = & 0 \Rightarrow (1 + c_{\text{stab}}J_0)w = \sum_i \alpha_i\varphi(x_i) \\ \dfrac{\partial \mathcal{L}}{\partial e_i} & = & 0 \Rightarrow (1 + c_{\text{stab}}J_0)\gamma e_i = \alpha_i \\ \dfrac{\partial \mathcal{L}}{\partial \alpha_i} & = & 0 \Rightarrow e_i = w^T\varphi(x_i). \end{array} \right.$$

Assuming that $1 + c_{\text{stab}}J_0 \neq 0$, elimination of w and e_i yields $\sum_j \alpha_j K_{ji} = (1/\gamma)\alpha_i$ with $K_{ji} = \varphi(x_j)^T\varphi(x_i)$, which is the solution that is also obtained for the original formulation (corresponding to $c_{\text{stab}} = 0$).