Journal of Machine Learning Research 6 (2005) 1043–1071 Submitted 5/03; Revised 10/04; Published 7/05

Learning the Kernel with Hyperkernels

Cheng Soon Ong* ChengSoon.Ong@tuebingen.mpg.de
Max Planck Institute for Biological Cybernetics and
Friedrich Miescher Laboratory,
Spemannstrasse 35, 72076 Tübingen, Germany.

Alexander J. Smola Alex.Smola@nicta.com.au

Robert C. Williamson Bob.Williamson@nicta.com.au
National ICT Australia,
Locked Bag 8001, Canberra ACT 2601, Australia
and Australian National University.

Editor: Ralf Herbrich

Abstract

This paper addresses the problem of choosing a kernel suitable for estimation with a Support Vector Machine, hence further automating machine learning. This goal is achieved by defining a Reproducing Kernel Hilbert Space on the space of kernels itself. Such a formulation leads to a statistical estimation problem similar to the problem of minimizing a regularized risk functional.

We state the equivalent representer theorem for the choice of kernels and present a semidefinite programming formulation of the resulting optimization problem. Several recipes for constructing hyperkernels are provided, as well as the details of common machine learning problems. Experimental results for classification, regression and novelty detection on UCI data show the feasibility of our approach.

Keywords: learning the kernel, capacity control, kernel methods, Support Vector Machines, representer theorem, semidefinite programming

1. Introduction

Kernel Methods have been highly successful in solving various problems in machine learning. The algorithms work by implicitly mapping the inputs into a feature space, and finding a suitable hypothesis in this new space. In the case of the Support Vector Machine (SVM), this solution is the hyperplane which maximizes the margin in the feature space. The feature mapping in question is defined by a kernel function, which allows us to compute dot products in feature space using only objects in the input space. For an introduction to SVMs and kernel methods, the reader is referred to numerous tutorials such as Burges (1998) and books such as Schölkopf and Smola (2002).

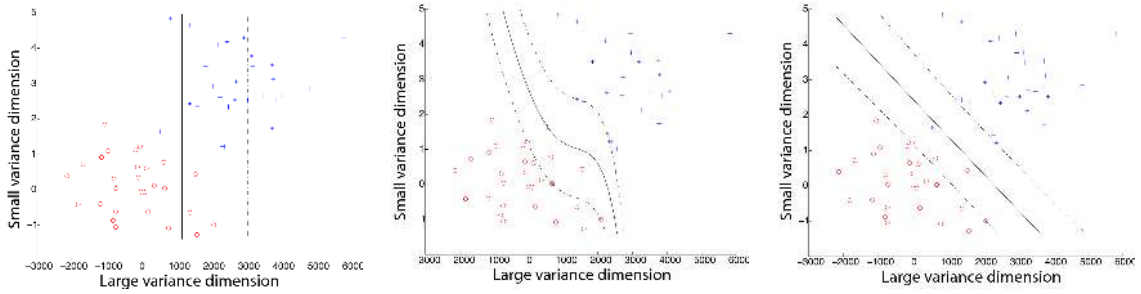
Choosing a suitable kernel function, and therefore a feature mapping, is imperative to the success of this inference process. This paper provides an inference framework for learning the kernel from training data using an approach akin to the regularized quality functional.

∗. This work was done when the author was at the Australian National University

1.1 Motivation

As motivation for the need for methods to learn the kernel, consider Figure 1, which shows the separating hyperplane, the margin and the training data for a synthetic dataset. Figure 1(a) shows the classification function for a support vector machine using a Gaussian radial basis function (RBF) kernel. The data has been generated using two Gaussian distributions with standard deviation 1 in one dimension and 1000 in the other. This difference in scale creates problems for the Gaussian RBF kernel, since it is unable to find a kernel width suitable for both directions. Hence, the classification function is dominated by the dimension with large variance. Increasing the value of the regularization parameter, C, and hence decreasing the smoothness of the function results in a hyperplane which is more complex, and equally unsatisfactory (Figure 1(b)). The traditional way to handle such data is to normalize each dimension independently.

Instead of normalising the input data, we make the kernel adaptive to allow independent scales for each dimension. This allows the kernel to handle unnormalised data. However, the resulting kernel would be difficult to hand-tune as there may be numerous free variables. In this case, we have a free parameter for each dimension of the input. We 'learn' this kernel by defining a quantity analogous to the risk functional, called the quality functional, which measures the 'badness' of the kernel function. The classification function for the above mentioned data is shown in Figure 1(c). Observe that it captures the scale of each dimension independently. In general, the solution does not consist of only a single kernel but a linear combination of them.



(a) Standard Gaussian RBF kernel (C=10) (b) Standard Gaussian RBF kernel (C=10^8) (c) RBF-Hyperkernel with adaptive widths

Figure 1: For data with highly non-isotropic variance, choosing one scale for all dimensions leads to unsatisfactory results. Plot of synthetic data, showing the separating hyperplane and the margins given for a uniformly chosen length scale (left and middle) and an automatic width selection (right).

1.2 Related Work

We analyze some recent approaches to learning the kernel by looking at the objective function that is being optimized and the class of kernels being considered. We will see later (Section 2) that this objective function is related to our definition of a quality functional.

Cross validation has been used to select the parameters of the kernels and SVMs (Duan et al., 2003, Meyer et al., 2003), with varying degrees of success. The objective function is the cross validation risk, and the class of kernels is a finite subset of the possible parameter settings. Duan et al. (2003) and Chapelle et al. (2002) test various simple approximations which bound the leave one out error, or some measure of the capacity of the SVM. The notion of Kernel Target Alignment (Cristianini et al., 2002) uses the objective function $tr(Kyy^\top)$ where y are the training labels, and K is from the class of kernels spanned by the eigenvectors of the kernel matrix of the combined training and test data. The semidefinite programming (SDP) approach (Lanckriet et al., 2004) uses a more general class of kernels, namely a linear combination of positive semidefinite matrices. They minimize the margin of the resulting SVM using a SDP for kernel matrices with constant trace. Similar to this, Bousquet and Herrmann (2002) further restricts the class of kernels to the convex hull of the kernel matrices normalized by their trace. This restriction, along with minimization of the complexity class of the kernel, allows them to perform gradient descent to find the optimum kernel. Using the idea of boosting, Crammer et al. (2002) optimize $\sum_t \beta_t K_t$, where β_t are the weights used in the boosting algorithm. The class of base kernels $\{K_t\}$ is obtained from the normalized solution of the generalized eigenvector problem. In principle, one can learn the kernel using Bayesian methods by defining a suitable prior, and learning the hyperparameters by optimizing the marginal likelihood (Williams and Barber, 1998, Williams and Rasmussen, 1996). As an example of this, when other information is available, an auxiliary matrix can be used with the EM algorithm for learning the kernel (Tsuda et al., 2003). Table 1 summarizes these approaches. The notation $K \succeq 0$ means that K is positive semidefinite, that is for all $a \in \mathbb{R}^n$, $a^\top K a \geqslant 0$.

Approach	Objective	Kernel class ($\mathcal{K}$)
Cross Validation	CV Risk	Finite set of kernels
Alignment	$y^\top K y$	$\{\sum_{i=1}^m \beta_i v_i v_i^\top$ such that v_i are eigenvectors of $K\}$
SDP	margin	$\{\sum_{i=1}^m \beta_i K_i$ such that $K_i \succeq 0, \mathrm{tr} K_i = c\}$
Complexity Bound	margin	$\{\sum_{i=1}^m \beta_i K_i$ such that $K_i \succeq 0, \mathrm{tr} K_i = c, \beta_i \geqslant 0\}$
Boosting	Exp/LogLoss	Base kernels from generalized eigenvector problem
Bayesian	neg. log-post.	dependent on prior
EM Algorithm	KL Divergence	linear combination of auxiliary matrix

Table 1: Summary of recent approaches to kernel learning.

1.3 Outline of the Paper

The contribution of this paper is a theoretical framework for learning the kernel. Using this framework, we analyze the regularized risk functional. Motivated by the ideas of Cristianini et al. (2003), we show (Section 2) that for most kernel-based learning methods there exists a functional, the *quality functional*, which plays a similar role to the empirical risk functional. We introduce a kernel on the space of kernels itself, a *hyperkernel* (Section 3), and its regularization on the associated Hyper Reproducing Kernel Hilbert Space (Hyper-RKHS). This leads to a systematic way of parameterizing kernel classes while managing overfitting (Ong

et al., 2002). We give several examples of hyperkernels and recipes to construct others (Section 4). Using this general framework, we consider the specific example of using the regularized risk functional in the rest of the paper. The positive definiteness of the kernel function is ensured using the positive definiteness of the kernel matrix (Section 5), and the resulting optimization problem is a semidefinite program. The semidefinite programming approach follows that of Lanckriet et al. (2004), with a different constraint due to a difference in regularization (Ong and Smola, 2003). Details of the specific optimization problems associated with the C-SVM, ν-SVM, Lagrangian SVM, ν-SVR and one class SVM are defined in Section 6. Experimental results for classification, regression and novelty detection (Section 7) are shown. Finally some issues and open problems are discussed (Section 8).

2. Kernel Quality Functionals

We denote by $\mathcal{X}$ the space of input data and $\mathcal{Y}$ the space of labels (if we have a supervised learning problem). Denote by $X_{\text{train}} := \{x_1, \ldots, x_m\}$ the training data and with $Y_{\text{train}} := \{y_1, \ldots, y_m\}$ a set of corresponding labels, jointly drawn independently and identically from some probability distribution $\Pr(x, y)$ on $\mathcal{X} \times \mathcal{Y}$. We shall, by convenient abuse of notation, generally denote Y_{train} by the vector y, when writing equations in matrix notation. We denote by K the kernel matrix given by $K_{ij} := k(x_i, x_j)$ where $x_i, x_j \in X_{\text{train}}$ and k is a positive semidefinite kernel function. We also use $\text{tr}K$ to mean the trace of the matrix and $|K|$ to mean the determinant.

We begin by introducing a new class of functionals Q on data which we will call *quality functionals*. Note that by quality we actually mean *badness* or lack of quality, as we would like to minimize this quantity. Their purpose is to indicate, given a kernel k and the training data, how suitable the kernel is for explaining the training data, or in other words, the *quality* of the kernel for the estimation problem at hand. Such quality functionals may be the Kernel Target Alignment, the negative log posterior, the minimum of the regularized risk functional, or any luckiness function for kernel methods. We will discuss those functionals after a formal definition of the quality functional itself.

2.1 Empirical and Expected Quality

Definition 1 (Empirical Quality Functional) *Given a kernel k, and data X, Y, we define $Q_{\text{emp}}(k, X, Y)$ to be an* empirical quality functional *if it depends on k only via $k(x_i, x_j)$ where $x_i, x_j \in X$ for $1 \leqslant i, j \leqslant m$.*

By this definition, Q_{emp} is a function which tells us how well matched k is to a specific dataset X, Y. Typically such a quantity is used to adapt k in such a manner that Q_{emp} is optimal (for example, optimal Kernel Target Alignment, greatest luckiness, smallest negative log-posterior), based on this one *single* dataset X, Y. Provided a sufficiently rich class of kernels $\mathcal{K}$ it is in general possible to find a kernel $k^* \in \mathcal{K}$ that attains the minimum of any such Q_{emp} regardless of the data. However, it is very unlikely that $Q_{\text{emp}}(k^*, X, Y)$ would be similarly small for other X, Y, for such a k^*. To measure the overall quality of k we therefore introduce the following definition:

Definition 2 (Expected Quality Functional) *Denote by* $Q_{\text{emp}}(k, X, Y)$ *an empirical quality functional, then*

$$Q(k) := \mathbf{E}_{X,Y}\left[Q_{\text{emp}}(k, X, Y)\right]$$

is defined to be the expected quality functional. Here the expectation is taken over X, Y*, where all* x_i, y_i *are drawn from* $\Pr(x, y)$*.*

Observe the similarity between the empirical quality functional, $Q_{\text{emp}}(k, X, Y)$, and the empirical risk of an estimator, $R_{\text{emp}}(f, X, Y) = \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, f(x_i))$ (where l is a suitable loss function); in both cases we compute the value of a functional which depends on some sample X, Y drawn from $\Pr(x, y)$ and a function and in both cases we have

$$Q(k) = \mathbf{E}_{X,Y}\left[Q_{\text{emp}}(k, X, Y)\right] \text{ and } R(f) = \mathbf{E}_{X,Y}\left[R_{\text{emp}}(f, X, Y)\right].$$

Here $R(f)$ denotes the expected risk. However, while in the case of the empirical risk we can interpret R_{emp} as the empirical estimate of the expected loss $R(f) = \mathbf{E}_{x,y}[l(x, y, f(x))]$, due to the general form of Q_{emp}, no such analogy is available for quality functionals. Finding a general-purpose bound of the expected error in terms of $Q(k)$ is difficult, since the definition of Q depends heavily on the algorithm under consideration. Nonetheless, it provides a general framework within which such bounds can be derived.

To obtain a generalization error bound, it is sufficient that Q_{emp} is concentrated around its expected value. Furthermore, one would require the deviation of the empirical risk to be upper bounded by Q_{emp} and possibly other terms. In other words, we assume a) we have given a concentration inequality on quality functionals, such as

$$\Pr\left\{|Q_{\text{emp}}(k, X, Y) - Q(k)| \geqslant \varepsilon_Q\right\} < \delta_Q,$$

and b) we have a bound on the deviation of the empirical risk in terms of the quality functional

$$\Pr\left\{|R_{\text{emp}}(f, X, Y) - R(f)| \geqslant \varepsilon_R\right\} < \delta(Q_{\text{emp}}).$$

Then we can chain both inequalities together to obtain the following bound

$$\Pr\left\{|R_{\text{emp}}(f, X, Y) - R(f)| \geqslant \varepsilon_R\right\} < \delta_Q + \delta(Q + \varepsilon_Q).$$

This means that the bound now becomes independent of the particular value of the quality functional obtained *on* the data, rather than the expected value of the quality functional. Bounds of this type have been derived for Kernel Target Alignment (Cristianini et al., 2003, Theorem 9) and the Algorithmic Luckiness framework (Herbrich and Williamson, 2002, Theorem 17).

2.2 Examples of Q_{emp}

Before we continue with the derivations of a regularized quality functional and introduce a corresponding Reproducing Kernel Hilbert Space, we give some examples of quality functionals and present their exact minimizers, whenever possible. This demonstrates that given a rich enough feature space, we can arbitrarily minimize the empirical quality functional Q_{emp}. The difference here from traditional kernel methods is the fact that we allow the

kernel to change. This extra degree of freedom allows us to overfit the training data. In many of the examples below, we show that given a feature mapping which can model the labels of the training data precisely, overfitting occurs. That is, if we use the training labels as the kernel matrix, we arbitrarily minimize the quality functional. The reader who is convinced that one can arbitrarily minimize Q_{emp}, by optimizing over a suitably large class of kernels, may skip the following examples.

Example 1 (Regularized Risk Functional) *These are commonly used in SVMs and related kernel methods (see Wahba (1990), Vapnik (1995), Schölkopf and Smola (2002)). They take on the general form*

$$R_{\text{reg}}(f, X_{\text{train}}, Y_{\text{train}}) := \frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, f(x_i)) + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \tag{1}$$

where $\|f\|_{\mathcal{H}}^2$ is the RKHS norm of f and l is a loss function such that for $f(x_i) = y_i$, $l(x_i, y_i, y_i) = 0$. By virtue of the representer theorem (see Section 3) we know that the minimizer of (1) can be written as a kernel expansion. This leads to the following definition of a quality functional, for a particular loss functional l:

$$Q_{\text{emp}}^{\text{regrisk}}(k, X_{\text{train}}, Y_{\text{train}}) := \min_{\alpha\in\mathbb{R}^m}\left[\frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, [K\alpha]_i) + \frac{\lambda}{2}\alpha^\top K\alpha\right]. \tag{2}$$

The minimizer of (2) is somewhat difficult to find, since we have to carry out a double minimization over K and α. However, we know that $Q_{\text{emp}}^{\text{regrisk}}$ is bounded from below by 0. Hence, it is sufficient if we can find a (possibly) suboptimal pair (α, k) for which $Q_{\text{emp}}^{\text{regrisk}} \leq \epsilon$ for any $\epsilon > 0$:

- *Note that for $K = \beta yy^\top$ and $\alpha = \frac{1}{\beta\|y\|^2}y$ we have $K\alpha = y$ and $\alpha^\top K\alpha = \beta^{-1}$. This leads to $l(x_i, y_i, f(x_i)) = 0$ and therefore $Q_{\text{emp}}^{\text{regrisk}}(k, X_{\text{train}}, Y_{\text{train}}) = \frac{\lambda}{2\beta}$. For sufficiently large β we can make $Q_{\text{emp}}^{\text{regrisk}}(k, X_{\text{train}}, Y_{\text{train}})$ arbitrarily close to 0.*
- *Even if we disallow setting K arbitrarily close to zero by setting $\text{tr}K = 1$, finding the minimum of (2) can be achieved as follows: let $K = \frac{1}{\|z\|^2}zz^\top$, where $z \in \mathbb{R}^m$, and $\alpha = z$. Then $K\alpha = z$ and we obtain*

$$\frac{1}{m}\sum_{i=1}^{m} l(x_i, y_i, [K\alpha]_i) + \frac{\lambda}{2}\alpha^\top K\alpha = \sum_{i=1}^{m} l(x_i, y_i, z_i) + \frac{\lambda}{2}\|z\|_2^2. \tag{3}$$

Choosing each $z_i = \text{argmin}_\zeta\, l(x_i, y_i, \zeta(x_i)) + \frac{\lambda}{2}\zeta^2$, where ζ are the possible hypothesis functions obtained from the training data, yields the minimum with respect to z. Since (3) tends to zero and the regularized risk is lower bounded by zero, we can still arbitrarily minimize $Q_{\text{emp}}^{\text{regrisk}}$. This is not surprising since the set of allowable K is huge.

Example 2 (Cross Validation) *Cross validation is a widely used method for estimating the generalization error of a particular learning algorithm. Specifically, the leave-one-out cross validation is an almost unbiased estimate of the generalization error (Luntz and*

Brailovsky, 1969). The quality functional for classification using kernel methods is given by:

$$Q_{\text{emp}}^{\text{loo}}(k, X_{\text{train}}, Y_{\text{train}}) := \min_{\alpha \in \mathbb{R}^m} \left[\frac{1}{m} \sum_{i=1}^{m} -y^i \operatorname{sign}([K\alpha^i]_i) \right],$$

which is optimized in Duan et al. (2003), Meyer et al. (2003).

Choosing $K = yy^\top$ and $\alpha^i = \frac{1}{\|y^i\|^2} y^i$, where α^i and y^i are the vectors α and y with the ith element set to zero, we have $K\alpha^i = y^i$. Hence we can match the training data perfectly. For a validation set of larger size, i.e. k-fold cross validation, the same result can be achieved by defining a corresponding α.

Example 3 (Kernel Target Alignment) *This quality functional was introduced by Cristianini et al. (2002) to assess the alignment of a kernel with training labels. It is defined by*

$$Q_{\text{emp}}^{\text{alignment}}(k, X_{\text{train}}, Y_{\text{train}}) := 1 - \frac{\operatorname{tr}(Kyy^\top)}{\|y\|_2^2 \|K\|_F}. \tag{4}$$

Here $\|y\|_2$ denotes the ℓ_2 norm of the vector of observations and $\|K\|_F$ is the Frobenius norm, i.e., $\|K\|_F^2 := \operatorname{tr}(KK^\top) = \sum_{i,j}(K_{ij})^2$. This quality functional was optimized in Lanckriet et al. (2004). By decomposing K into its eigensystem one can see that (4) is minimized, if $K = yy^\top$, in which case

$$Q_{\text{emp}}^{\text{alignment}}(k^*, X_{\text{train}}, Y_{\text{train}}) = 1 - \frac{\operatorname{tr}(y^\top yy^\top y)}{\|y\|_2^2 \|yy^\top\|_F} = 1 - \frac{\|y\|_2^4}{\|y\|_2^2 \|y\|_2^2} = 0.$$

We cannot expect that $Q_{\text{emp}}^{\text{alignment}}(k^*, X, Y) = 0$ for data other than that chosen to determine k^*, in other words, a restriction of the class of kernels is required. This was also observed in Cristianini et al. (2003).

The above examples illustrate how existing methods for assessing the quality of a kernel fit within the quality functional framework. We also saw that given a rich enough class of kernels $\mathcal{K}$, optimization of Q_{emp} over $\mathcal{K}$ would result in a kernel that would be useless for prediction purposes, in the sense that they can be made to look arbitrarily good in terms of Q_{emp} but with the result that the generalization performance will be poor. This is yet another example of the danger of optimizing too much and overfitting – there is (still) no free lunch.

3. Hyper Reproducing Kernel Hilbert Spaces

We now propose a conceptually simple method to optimize quality functionals over classes of kernels by introducing a Reproducing Kernel Hilbert Space *on the kernel k itself*, so to say, a Hyper-RKHS. We first review the definition of a RKHS (Aronszajn, 1950).

Definition 3 (Reproducing Kernel Hilbert Space) *Let $\mathcal{X}$ be a nonempty set (the index set) and denote by $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with the dot product $\langle \cdot, \cdot \rangle$ (and the norm $\|f\| := \sqrt{\langle f, f \rangle}$) if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties.*

1. *k has the reproducing property*

$$\langle f, k(x,\cdot)\rangle = f(x) \text{ for all } f \in \mathcal{H}, x \in \mathcal{X};$$

in particular, $\langle k(x,\cdot), k(x',\cdot)\rangle = k(x,x')$ for all $x, x' \in \mathcal{X}$.

2. *k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{\text{span}\{k(x,\cdot)|x \in \mathcal{X}\}}$ where $\overline{X}$ is the completion of the set X.*

In the rest of the paper, we use the notation k to represent the kernel function and $\mathcal{H}$ to represent the RKHS. In essence, $\mathcal{H}$ is a Hilbert space of functions, which has the special property of being generated by the kernel function k.

The advantage of optimization in an RKHS is that under certain conditions the optimal solutions can be found as the linear combination of a finite number of basis functions, regardless of the dimensionality of the space $\mathcal{H}$ the optimization is carried out in. The theorem below formalizes this notion (see Kimeldorf and Wahba (1971), Cox and O'Sullivan (1990)).

Theorem 4 (Representer Theorem) *Denote by $\Omega : [0,\infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $l : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the general regularized risk*

$$l\left((x_1, y_1, f(x_1)), \ldots, (x_m, y_m, f(x_m))\right) + \Omega\left(\|f\|_{\mathcal{H}}\right)$$

admits a representation of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x), \tag{5}$$

where $\alpha_i \in \mathbb{R}$ for all $1 \leqslant i \leqslant m$.

3.1 Regularized Quality Functional

To learn the kernel, we need to define a function space of kernels, a method to regularize them and a practical optimization procedure. We will address each of these issues in the following. We define a RKHS on kernels $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, simply by introducing the compounded index set, $\underline{\mathcal{X}} := \mathcal{X} \times \mathcal{X}$ and by treating k as a function $k : \underline{\mathcal{X}} \to \mathbb{R}$:

Definition 5 (Hyper Reproducing Kernel Hilbert Space) *Let $\mathcal{X}$ be a nonempty set. and denote by $\underline{\mathcal{X}} := \mathcal{X} \times \mathcal{X}$ the compounded index set. The Hilbert space $\underline{\mathcal{H}}$ of functions $k : \underline{\mathcal{X}} \to \mathbb{R}$, endowed with a dot product $\langle\cdot,\cdot\rangle$ (and the norm $\|k\| = \sqrt{\langle k, k\rangle}$) is called a Hyper Reproducing Kernel Hilbert Space if there exists a hyperkernel $\underline{k} : \underline{\mathcal{X}} \times \underline{\mathcal{X}} \to \mathbb{R}$ with the following properties:*

1. *$\underline{k}$ has the reproducing property $\langle k, \underline{k}(\underline{x},\cdot)\rangle = k(\underline{x})$ for all $k \in \underline{\mathcal{H}}$; in particular, $\langle \underline{k}(\underline{x},\cdot), \underline{k}(\underline{x}',\cdot)\rangle = \underline{k}(\underline{x},\underline{x}')$.*

2. *$\underline{k}$ spans $\underline{\mathcal{H}}$, i.e. $\underline{\mathcal{H}} = \overline{\text{span}\{\underline{k}(\underline{x},\cdot)|\underline{x} \in \underline{\mathcal{X}}\}}$.*

3. *$\underline{k}(x,y,s,t) = \underline{k}(y,x,s,t)$ for all $x, y, s, t \in \mathcal{X}$.*

This is a RKHS with the additional requirement of symmetry in its first two arguments (in fact, we can have a recursive definition of an RKHS of an RKHS ad infinitum, with suitable restrictions on the elements). We define the corresponding notations for elements, kernels, and RKHS by underlining it. What distinguishes $\underline{\mathcal{H}}$ from a normal RKHS is the particular form of its index set ($\underline{\mathcal{X}} = \mathcal{X}^2$) and the additional condition on $\underline{k}$ to be symmetric in its first two arguments, and therefore in its second two arguments as well.

This approach of defining a RKHS on the space of symmetric functions of two variables leads us to a natural regularization method. By analogy with the definition of the regularized risk functional (1), we proceed to define the regularized quality functional.

Definition 6 (Regularized Quality Functional) *Let X, Y be the combined training and test set of examples and labels respectively. For a positive semidefinite kernel matrix K on X, the* regularized quality functional *is defined as*

$$Q_{\mathrm{reg}}(k, X, Y) := Q_{\mathrm{emp}}(k, X, Y) + \frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}, \tag{6}$$

where $\lambda_Q \geqslant 0$ is a regularization constant and $\|k\|^2_{\underline{\mathcal{H}}}$ denotes the RKHS norm in $\underline{\mathcal{H}}$.

Note that although we have possibly non positive kernels in $\underline{\mathcal{H}}$, we define the regularized quality functional only on positive semidefinite kernel matrices. This is a slightly weaker condition than requiring a positive semidefinite kernel k, since we only require positivity on the data. Since Q_{emp} depends on k only via the data, this is sufficient for the above definition. Minimization of Q_{reg} is less prone to overfitting than minimizing Q_{emp}, since the regularization term $\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}$ effectively controls the complexity of the class of kernels under consideration. Bousquet and Herrmann (2002) provide a generalization error bound by estimating the Rademacher complexity of the kernel classes in the transduction setting. Regularizers other than $\|k\|^2_{\underline{\mathcal{H}}}$ are possible, such as ℓ_p penalties. In this paper, we restrict ourselves to the ℓ_2 norm (6). The advantage of (6) is that its minimizer satisfies the representer theorem.

Lemma 7 (Representer Theorem for Hyper-RKHS) *Let $\mathcal{X}$ be a set, Q_{emp} an arbitrary empirical quality functional, and X, Y the combined training and test set, then each minimizer $k \in \underline{\mathcal{H}}$ of the regularized quality functional $Q_{\mathrm{reg}}(k, X, Y)$ admits a representation of the form*

$$k(x, x') = \sum_{i,j}^{m} \beta_{ij}\underline{k}((x_i, x_j), (x, x')) \text{ for all } x, x' \in X, \tag{7}$$

where $\beta_{ij} \in \mathbb{R}$, for each $1 \leqslant i, j \leqslant m$.

Proof All we need to do is rewrite (6) so that it satisfies the conditions of Theorem 4. Let $\underline{x}_{ij} := (x_i, x_j)$. Then $Q_{\mathrm{emp}}(k, X, Y)$ has the properties of a loss function, as it only depends on k via its values at $\underline{x}_{ij}$. Note too that the kernel matrix K also only depends on k via its values at $\underline{x}_{ij}$. Furthermore, $\frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}}$ is an RKHS regularizer, so the representer theorem applies and (7) follows. ∎

Lemma 7 implies that the solution of the regularized quality functional is a linear combination of hyperkernels on the input data. This shows that even though the optimization takes

place over an entire Hilbert space of kernels, one can find the optimal solution by choosing among a finite number.

Note that the minimizer (7) is not necessarily positive semidefinite. In practice, this is not what we want, since we require a positive semidefinite kernel but we do not have any guarantees for examples in the test set. Therefore we need to impose additional constraints of the type $K \succeq 0$ or k is a Mercer Kernel. While the latter is almost impossible to enforce directly, the former could be verified directly, hence imposing a constraint only on the values of the kernel matrix $k(x_i, x_j)$ rather than on the kernel function k itself. This means that the conditions of the Representer Theorem apply and (7) applies (with suitable constraints on the coefficients β_{ij}).

Another option is to be somewhat more restrictive and require that all expansion coefficients $\beta_{i,j} \geqslant 0$ and all the functions be positive semidefinite kernels. This latter requirement can be formally stated as follows: For any fixed $\underline{x} \in \underline{\mathcal{X}}$ the hyperkernel $\underline{k}$ is a kernel in its second argument; that is for any fixed $\underline{x} \in \underline{\mathcal{X}}$, the function $k(x, x') := \underline{k}(\underline{x}, (x, x'))$, with $x, x' \in \mathcal{X}$, is a positive semidefinite kernel.

Proposition 8 *Given a hyperkernel, $\underline{k}$ with elements such that for any fixed $\underline{x} \in \underline{\mathcal{X}}$, the function $k(x_p, x_q) := \underline{k}(\underline{x}, (x_p, x_q))$, with $x_p, x_q \in \mathcal{X}$, is a positive semidefinite kernel, and $\beta_{ij} \geqslant 0$ for all $i, j = 1, \ldots, m$, then the kernel*

$$k(x_p, x_q) := \sum_{i,j=1}^{m} \beta_{ij} \underline{k}(x_i, x_j, x_p, x_q)$$

is positive semidefinite.

Proof The result is obtained by observing that positive combinations of positive semidefinite kernels are positive semidefinite. ■

While this may prevent us from obtaining the minimizer of the objective function, it yields a much more amenable optimization problem in practice, in particular if the resulting cone spans a large enough space (as happens with increasing m). In the subsequent derivations of optimization problems, we choose this restriction as it provides a more tractable problem in practice. In Section 4, we give examples and recipes for constructing hyperkernels. Before that, we relate our framework defined above to Bayesian inference.

3.2 A Bayesian Perspective

A generative Bayesian approach to inference encodes all knowledge we might have about the problem setting into a prior distribution. Hence, the choice of the prior distribution determines the behaviour of the inference, as once we have the data, we condition on the prior distribution we have chosen to obtain the posterior, and then marginalize to obtain the label that we are interested in. One popular choice of prior is the normal distribution, resulting in a Gaussian process (GP). All prior knowledge we have about the problem is then encoded in the covariance of the GP. There exists a GP analog to the Support Vector Machine (for example Opper and Winther (2000), Seeger (1999)), which

is essentially obtained (ignoring normalizing terms) by exponentiating the regularized risk functional used in SVMs.

In this section, we derive the prior and hyperprior implied by our framework of hyperkernels. This is obtained by exponentiating Q_{reg}, again ignoring normalization terms. Given the regularized quality functional (Equation 6), with the Q_{emp} set to the SVM with squared loss, we obtain the following equation.

$$Q_{\text{reg}}(k, X, Y) := \frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \frac{\lambda_Q}{2}\|k\|_{\underline{\mathcal{H}}}^2.$$

Exponentiating the negative of the above equation gives,

$$\begin{aligned}&\exp(-Q_{\text{reg}}(k, X, Y)) = \\ &\exp\left(-\frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2\right)\exp\left(-\frac{\lambda}{2}\|f\|_{\mathcal{H}}^2\right)\exp\left(-\frac{\lambda_Q}{2}\|k\|_{\underline{\mathcal{H}}}^2\right).\end{aligned} \tag{8}$$

We compare Equation (8) to Gaussian process estimation. The general scheme is known in Bayesian estimation as hyperpriors (Bishop, 1995, Chapter 10), which determine the distribution of the priors (here the GP with covariance k). Figure 2 describes the model of an ordinary GP, where f is drawn from a Gaussian distribution with covariance matrix K and y is conditionally independent given f. For hyperprior estimation, we draw the prior K from a distribution instead of setting it.

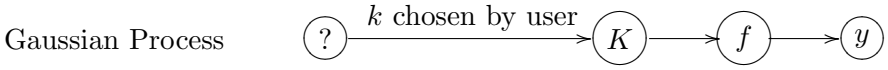


Figure 2: Generative model for Gaussian process estimation

To determine the distribution from which we draw the prior, we compute the hyperprior explicitly. For given data $Z = \{X, Y\}$ and applying Bayes' Rule, the posterior is given by

$$p(f|Z, k) = \frac{p(Z|f, k)p(f|k)p(k)}{p(k|Z)p(Z)}. \tag{9}$$

We have the directed graphical model shown in Figure 3 for a Hyperkernel-GP, where we assume that the covariance matrix of the Gaussian process K is drawn according to a distribution before performing further steps of dependency calculation. We shall now explicitly compute the terms in the numerator of Equation (9).

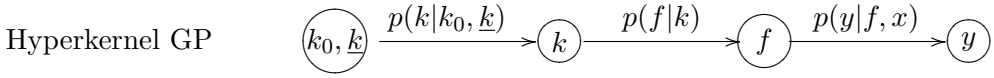


Figure 3: Generative model for Gaussian process estimation using hyperpriors on k defined by $\underline{k}$.

In the following derivations, we assume that we are dealing with finite dimensional objects, to simplify the calculations of the normalizing constants in the expressions for the distributions. Given that we have additive Gaussian noise, that is $\epsilon \sim \mathcal{N}(0, \frac{1}{\gamma_\epsilon}\mathbf{I})$, then,

$$p(y|f,x) \propto \exp\left(-\frac{\gamma_\epsilon}{2}(y - f(x))^2\right).$$

Therefore, for the whole dataset (assumed to be i.i.d.),

$$p(Y|f,X) = \prod_{i=1}^{m} p(y_i|f,x_i) = \left(\frac{2\pi}{\gamma_\epsilon}\right)^{-\frac{m}{2}} \exp\left(-\frac{\gamma_\epsilon}{2}\sum_{i=1}^{m}(y_i - f(x_i))^2\right).$$

We assume a Gaussian prior on the function f, with covariance function k. The positive semidefinite function, k, defines an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_k}$ in the RKHS denoted by $\mathcal{H}_k$. Then,

$$p(f|k) = \left(\frac{2\pi}{\gamma_f}\right)^{-\frac{F}{2}} \exp\left(-\frac{\gamma_f}{2}\langle f, f\rangle_{\mathcal{H}_k}\right)$$

where F is the dimension of f and γ_f is a constant.

We assume a Wishart distribution (Lauritzen, 1996, Appendix C), with p degrees of freedom and covariance k_0, for the prior distribution of the covariance function k, that is $k \sim \mathcal{W}_m(p, k_0)$. This is a hyperprior used in the Gaussian process literature.

$$p(k|k_0) = \frac{|k|^{\frac{p-(m+1)}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(kk_0)\right)}{\Gamma_m(p)|k|^{\frac{p}{2}}}$$

where $\Gamma_m(p)$ denotes the Gamma distribution, $\Gamma_m(p) = 2^{\frac{pm}{2}}\pi^{\frac{m(m-1)}{4}}\prod_{i=1}^{m}\Gamma\left(\frac{p-i+1}{2}\right)$. For more details of the Wishart distribution, the reader is referred to Lauritzen (1996).

Observe that $\mathrm{tr}(kk_0)$ is an inner product between two matrices. We can define a general inner product between two matrices, as the inner product defined in the RKHS denoted by $\underline{\mathcal{H}}$.

$$p(k|k_0,\underline{k}) = \frac{|k|^{\frac{p-(m+1)}{2}} \exp\left(-\frac{1}{2}\langle k, k_0\rangle_{\underline{\mathcal{H}}}\right)}{\Gamma_m(p)|k|^{\frac{p}{2}}}$$

We can interpret the above equation as measuring the similarity between the covariance matrix that we obtain from data and the expected covariance matrix (given by the user). This similarity is measured by a dot product defined by $\underline{k}$. Substituting the expressions for $p(Y|X,f), p(f|k)$ and $p(k|k_0,\underline{k})$ into the posterior (Equation 9), we get Equation (10) which is of the same form as the exponentiated negative quality (Equation 8).

$$\exp\left(-\frac{\gamma_\epsilon}{2}\sum_{i=1}^{m}(y_i - f(x_i))^2\right)\exp\left(-\frac{\gamma_f}{2}\langle f, f\rangle_{\mathcal{H}_k}\right)\exp\left(-\frac{1}{2}\langle k, k_0\rangle_{\underline{\mathcal{H}}}\right). \tag{10}$$

In a nutshell, we assume that the covariance function of the GP k, is distributed according to a Wishart distribution. In other words, we have two nested processes, a Gaussian and

a Wishart process, to model the data generation scheme. Hence we are studying a mixture of Gaussian processes. Note that the maximum likelihood (ML-II) estimator (MacKay, 1994, Williams and Barber, 1998, Williams and Rasmussen, 1996) in Bayesian estimation leads to the same optimization problems as those arising from minimizing the regularized quality functional.

4. Hyperkernels

Having introduced the theoretical basis of the Hyper-RKHS, it is natural to ask whether hyperkernels, $\underline{k}$, exist which satisfy the conditions of Definition 5. We address this question by giving a set of general recipes for building such kernels.

4.1 Power Series Construction

Suppose k is a kernel such that $k(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$, and suppose $g : \mathbb{R} \to \mathbb{R}$ is a function with positive Taylor expansion coefficients, that is $g(\xi) = \sum_{i=0}^{\infty} c_i \xi^i$ for basis functions ξ, $c_i \geqslant 0$ for all $i = 0, \ldots, \infty$, and convergence radius R. Then for pointwise positive $k(x, x') \leq \sqrt{R}$,

$$\underline{k}(\underline{x}, \underline{x}') := g(k(\underline{x})k(\underline{x}')) = \sum_{i=0}^{\infty} c_i (k(\underline{x})k(\underline{x}'))^i \tag{11}$$

is a hyperkernel. For $\underline{k}$ to be a hyperkernel, we need to check that first, $\underline{k}$ is a kernel, and second, for any fixed pair of elements of the input data, $\underline{x}$, the function $\underline{k}(\underline{x}, (x, x'))$ is a kernel, and third that is satisfies the symmetry condition. Here, the symmetry condition follows from the symmetry of k. To see this, observe that for any fixed $\underline{x}$, $\underline{k}(\underline{x}, (x, x'))$ is a sum of kernel functions, hence it is a kernel itself (since $k^p(x, x')$ is a kernel if k is, for $p \in \mathbb{N}$). To show that $\underline{k}$ is a kernel, note that $\underline{k}(\underline{x}, \underline{x}') = \langle \underline{\Phi}(\underline{x}), \underline{\Phi}(\underline{x}') \rangle$, where $\underline{\Phi}(\underline{x}) := (\sqrt{c_0}, \sqrt{c_1}k^1(\underline{x}), \sqrt{c_2}k^2(\underline{x}), \ldots)$. Note that we require pointwise positivity, so that the coefficients of the sum in Equation (11) are always positive. The Gaussian RBF kernel satisfies this condition, but polynomial kernels of odd degree are not always pointwise positive. In the following example, we use the Gaussian kernel to construct a hyperkernel.

Example 4 (Harmonic Hyperkernel) *Suppose k is a kernel with range $[0, 1]$, (RBF kernels satisfy this property), and set $c_i := (1 - \lambda_h)\lambda_h^i$, $i \in \mathbb{N}$, for some $0 < \lambda_h < 1$. Then we have*

$$\underline{k}(\underline{x}, \underline{x}') = (1 - \lambda_h) \sum_{i=0}^{\infty} \left(\lambda_h k(\underline{x})k(\underline{x}')\right)^i = \frac{1 - \lambda_h}{1 - \lambda_h k(\underline{x})k(\underline{x}')}. \tag{12}$$

For $k(x, x') = \exp(-\sigma^2 \|x - x'\|^2)$ this construction leads to

$$\underline{k}((x, x'), (x'', x''')) = \frac{1 - \lambda_h}{1 - \lambda_h \exp\left(-\sigma^2(\|x - x'\|^2 + \|x'' - x'''\|^2)\right)}. \tag{13}$$

As one can see, for $\lambda_h \to 1$, $\underline{k}$ converges to $\delta_{\underline{x},\underline{x}'}$, and thus $\|k\|^2_{\underline{\mathcal{H}}}$ converges to the Frobenius norm of k on $X \times X$.

$g(\xi)$	Power series expansion	Radius of Convergence
$\exp\xi$	$1+\frac{1}{1!}\xi+\frac{1}{2!}\xi^2+\frac{1}{3!}\xi^3+\ldots+\frac{1}{n!}\xi^n+\ldots$	∞
$\sinh\xi$	$\frac{1}{1!}\xi+\frac{1}{3!}\xi^3+\frac{1}{5!}\xi^5+\ldots+\frac{1}{(2n+1)!}\xi^{(2n+1)}+\ldots$	∞
$\cosh\xi$	$1+\frac{1}{2!}\xi^2+\frac{1}{4!}\xi^4+\ldots+\frac{1}{(2n)!}\xi^{(2n)}+\ldots$	∞
$\operatorname{arctanh}\xi$	$\frac{\xi}{1}+\frac{\xi^3}{3}+\frac{\xi^5}{5}+\ldots+\frac{\xi^{2n+1}}{2n+1}+\ldots$	1
$-\ln(1-\xi)$	$\frac{\xi}{1}+\frac{\xi^2}{2}+\frac{\xi^3}{3}+\ldots+\frac{\xi^n}{n}+\ldots$	1

Table 2: Hyperkernels by Power Series Construction.

It is straightforward to find other hyperkernels of this sort, simply by consulting tables on power series of functions. Table 2 contains a short list of suitable expansions.

However, if we want the kernel to adapt automatically to different widths for each dimension, we need to perform the summation that led to (12) for each dimension in its arguments separately. Such a hyperkernel corresponds to ideas developed in automatic relevance determination (ARD) (MacKay, 1994, Neal, 1996).

Example 5 (Hyperkernel for ARD) *Let $k_\Sigma(x,x') = \exp(-d_\Sigma(x,x'))$, where $d_\Sigma(x,x') = (x-x')^\top\Sigma(x-x')$, and Σ is a diagonal covariance matrix. Take sums over each diagonal entry $\sigma_j = \Sigma_{jj}$ separately to obtain*

$$\begin{aligned}\underline{k}((x,x'),(x'',x''')) &= (1-\lambda_h)\sum_{j=1}^{d}\sum_{i=0}^{\infty}\left(\lambda_h k_\Sigma(x,x')k_\Sigma(x'',x''')\right)^i \\ &= \prod_{j=1}^{d}\frac{1-\lambda_h}{1-\lambda_h\exp\left(-\sigma_j((x_j-x_j')^2+(x_j''-x_j''')^2)\right)}. \qquad (14)\end{aligned}$$

Eq. (14) holds since $k(\underline{x})$ factorizes into its coordinates. A similar definition also allows us to use a distance metric $d(x,x')$ which is a generalized radial distance as defined by Haussler (1999).

4.2 Hyperkernels Invariant to Translation

Another approach to constructing hyperkernels is via an extension of a result due to Smola et al. (1998) concerning the Fourier transform of translation invariant kernels.

Theorem 9 (Translation Invariant Hyperkernel) *Suppose $\underline{k}((x_1-x_1'),(x_2-x_2'))$ is a function which depends on its arguments only via x_1-x_1' and x_2-x_2'. Let $\mathcal{F}_1\underline{k}(\omega,(x_2-x_2'))$ denote the Fourier transform with respect to (x_1-x_1').*

The function $\underline{k}$ is a hyperkernel if $\underline{k}(\tau,\tau')$ is a kernel in τ,τ' and $\mathcal{F}_1\underline{k}(\omega,(x''-x'''))\geq 0$ for all $(x''-x''')$ and ω.

Proof From (Smola et al., 1998) we know that for $\underline{k}$ to be a kernel in one of its arguments, its Fourier transform has to be nonnegative. This yields the second condition. Next, we

need to show that $\underline{k}$ is a kernel in its own right. Mercer's condition requires that for arbitrary f the following is positive:

$$\begin{aligned} & \int f(x_1, x_1') f(x_2, x_2') \underline{k}((x_1 - x_1'), (x_2 - x_2')) dx_1 dx_1' dx_2 dx_2' \\ = & \int f(\tau_1 + x_1', x_1') f(\tau_2 + x_2', x_2') dx_{1,2} \underline{k}(\tau_1, \tau_2) d\tau_1 d\tau_2 \\ = & \int g(\tau_1) g(\tau_2) \underline{k}(\tau_1, \tau_2) d\tau_1 d\tau_2, \end{aligned}$$

where $\tau_1 = x_1 - x_1'$ and $\tau_2 = x_2 - x_2'$. Here g is obtained by integration over x_1 and x_2 respectively. The latter is exactly Mercer's condition on $\underline{k}$, when viewed as a function of two variables only. ■

This means that we can check whether a radial basis function (for example Gaussian RBF, exponential RBF, damped harmonic oscillator, generalized B_n spline), can be used to construct a hyperkernel by checking whether its Fourier transform is positive.

4.3 Explicit Expansion

If we have a finite set of kernels that we want to choose from, we can generate a hyperkernel which is a finite sum of possible kernel functions. This setting is similar to that of Lanckriet et al. (2004).

Suppose $k_i(x, x')$ is a kernel for each $i = 1, \ldots, n$ (for example the RBF kernel or the polynomial kernel), then

$$\underline{k}(\underline{x}, \underline{x}') := \sum_{i=1}^{n} c_i k_i(\underline{x}) k_i(\underline{x}'), k_i(\underline{x}) \geqslant 0, \forall \underline{x} \tag{15}$$

is a hyperkernel, as can be seen by an argument similar to that of section 4.1. $\underline{k}$ is a kernel since $\underline{k}(\underline{x}, \underline{x}') = \langle \underline{\Phi}(\underline{x}), \underline{\Phi}(\underline{x}') \rangle$, where $\underline{\Phi}(\underline{x}) := (\sqrt{c_1} k_1(\underline{x}), \sqrt{c_2} k_2(\underline{x}), \ldots, \sqrt{c_n} k_n(\underline{x}))$.

Example 6 (Polynomial and RBF combination) *Let $k_1(x, x') = (\langle x, x' \rangle + b)^{2p}$ for some choice of $b \in \mathbb{R}^+$ and $p \in \mathbb{N}$, and $k_2(x, x') = \exp(-\sigma^2 \|x - x'\|^2)$. Then,*

$$\begin{aligned} \underline{k}((x_1, x_1'), (x_2, x_2')) & = c_1 (\langle x_1, x_1' \rangle + b)^{2p} (\langle x_2, x_2' \rangle + b)^{2p} \\ & \quad + c_2 \exp(-\sigma^2 \|x_1 - x_1'\|^2) \exp(-\sigma^2 \|x_2 - x_2'\|^2) \end{aligned} \tag{16}$$

is a hyperkernel.

5. Optimization Problems for Regularized Risk based Quality Functionals

We will now consider the optimization of the quality functionals utilizing hyperkernels. We choose the regularized risk functional as the empirical quality functional; that is we set $Q_{\text{emp}}(k, X, Y) := R_{\text{reg}}(f, X, Y)$. It is possible to utilize other quality functionals, such as the Kernel Target Alignment (Example 12). We focus our attention on the regularized risk functional, which is commonly used in SVMs. Furthermore, we will only consider positive semidefinite kernels. For a particular loss function $l(x_i, y_i, f(x_i))$, we obtain the regularized quality functional.

$$\min_{k \in \underline{\mathcal{H}}} \min_{f \in \mathcal{H}_k} \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, f(x_i)) + \frac{\lambda}{2} \|f\|_{\mathcal{H}_k}^2 + \frac{\lambda_Q}{2} \|k\|_{\underline{\mathcal{H}}}^2. \tag{17}$$

By the representer theorem (Theorem 4 and Corollary 7) we can write the regularizers as quadratic terms. Using the soft margin loss, we obtain

$$\min_{\beta} \min_{\alpha} \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i f(x_i)) + \frac{\lambda}{2} \alpha^\top K \alpha + \frac{\lambda_Q}{2} \beta^\top \underline{K} \beta \;\text{ subject to }\; \beta \geqslant 0 \tag{18}$$

where $\alpha \in \mathbb{R}^m$ are the coefficients of the kernel expansion (5), and $\beta \in \mathbb{R}^{m^2}$ are the coefficients of the hyperkernel expansion (7).

For fixed k, the problem can be formulated as a constrained minimization problem in f, and subsequently expressed in terms of the Lagrange multipliers α. However, this minimum depends on k, and for efficient minimization we would like to compute the derivatives with respect to k. The following lemma tells us how (it is an extension of a result in Chapelle et al. (2002)):

Lemma 10 *Let $x \in \mathbb{R}^m$ and denote by $f(x, \theta), c_i : \mathbb{R}^m \to \mathbb{R}$ convex functions, where f is parameterized by θ. Let $R(\theta)$ be the minimum of the following optimization problem (and denote by $x(\theta)$ its minimizer):*

$$\underset{x \in \mathbb{R}^m}{\text{minimize}} \;\; f(x, \theta) \text{ subject to } c_i(x) \leq 0 \text{ for all } 1 \leq i \leq n.$$

Then $\partial_\theta^j R(\theta) = D_2^j f(x(\theta), \theta)$, where $j \in \mathbb{N}$ and D_2 denotes the derivative with respect to the second argument of f.

Proof At optimality we have a saddlepoint in the Lagrangian

$$\partial_x \mathcal{L}(x, \alpha) = \partial_x f(x, \theta) + \sum_{i=1}^{n} \alpha_i \partial_x c_i(x) = 0. \tag{19}$$

Furthermore, for all θ the Kuhn-Tucker conditions have to hold, and in particular also $\sum_{i=1}^n \alpha_i \partial_\theta c_i(x(\theta)) = 0$, since for all $\alpha_i > 0$ the condition $c_i(x) = 0$ and therefore also $\partial_\theta c_i(x(\theta)) = 0$ has to be satisfied. Taking higher order derivatives with respect to θ yields

$$0 = \partial_\theta^j \left[\sum_{i=1}^{n} \alpha_i \partial_x c_i(x(\theta)) \frac{\partial x}{\partial \theta} \right] = \partial_\theta^j \left[-\partial_x f(x, \theta) \frac{\partial x}{\partial \theta} \right]. \tag{20}$$

Here the last equality follows from (19). Next we use

$$\partial_\theta^{j+1} f(x, \theta) = \partial_\theta^j \left[D_2 f(x, \theta) + \partial_x f(x, \theta) \frac{\partial x}{\partial \theta} \right] = \partial_\theta^j D_2 f(x, \theta).$$

Repeated application then proves the claim. ■

Instead of directly minimizing Equation (18), we derive the dual formulation. Using the approach in Lanckriet et al. (2004), the corresponding optimization problems can be expressed as a SDP. In general, solving a SDP would be take longer than solving a quadratic

program (a traditional SVM is a quadratic program). This reflects the added cost incurred for optimizing over a class of kernels.

Semidefinite programming (Vandenberghe and Boyd, 1996) is the optimization of a linear objective function subject to constraints which are linear matrix inequalities and affine equalities.

Definition 11 (Semidefinite Program) *A semidefinite program (SDP) is a problem of the form:*

$$\begin{array}{ll} \min\limits_{x} & c^\top x \\ \textit{subject to} & F_0 + \sum\limits_{i=1}^{q} x_i F_i \succeq 0 \textit{ and } Ax = b \end{array}$$

where $x \in \mathbb{R}^p$ are the decision variables, $A \in \mathbb{R}^{p \times q}$, $b \in \mathbb{R}^p$, $c \in \mathbb{R}^q$, and $F_i \in \mathbb{R}^{r \times r}$ are given.

In general, linear constraints $Ax + a \geqslant 0$ can be expressed as a semidefinite constraint $diag(Ax + a) \succeq 0$, and a convex quadratic constraint $(Ax + b)^\top (Ax + b) - c^\top x - d \leqslant 0$ can be written as

$$\begin{bmatrix} I & Ax + b \\ (Ax + b)^\top & c^\top x + d \end{bmatrix} \succeq 0.$$

When $t \in \mathbb{R}$, we can write the quadratic constraint $a^\top A a \leqslant t$ as $\|A^{\frac{1}{2}} a\| \leqslant t$. In practice, linear and quadratic constraints are simpler and faster to implement in a convex solver.

We derive the corresponding SDP for Equation (17). The following proposition allows us to derive a SDP from a class of general convex programs. It follows the approach in Lanckriet et al. (2004), with some care taken with Schur complements of positive semidefinite matrices (Albert, 1969), and its proof is omitted for brevity.

Proposition 12 (Quadratic Minimax) *Let $m, n, M \in \mathbb{N}$, $H : \mathbb{R}^n \to \mathbb{R}^{m \times m}$, $c : \mathbb{R}^n \to \mathbb{R}^m$, be linear maps. Let $A \in \mathbb{R}^{M \times m}$ and $a \in \mathbb{R}^M$. Also, let $d : \mathbb{R}^n \to \mathbb{R}$ and $G(\xi)$ be a function and the further constraints on ξ. Then the optimization problem*

$$\begin{array}{lll} \underset{\xi \in \mathbb{R}^n}{\text{minimize}} \, \underset{x \in \mathbb{R}^m}{\text{maximize}} & -\frac{1}{2} x^\top H(\xi) x - c(\xi)^\top x + d(\xi) \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & G(\xi) \succeq 0 \end{array} \tag{21}$$

can be rewritten as

$$\begin{array}{ll} \underset{t, \xi, \gamma}{\text{minimize}} & \frac{1}{2} t + a^\top \gamma + d(\xi) \\ \textit{subject to} & \begin{bmatrix} diag(\gamma) & 0 & 0 & 0 \\ 0 & G(\xi) & 0 & 0 \\ 0 & 0 & H(\xi) & (A^\top \gamma - c(\xi)) \\ 0 & 0 & (A^\top \gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{array} \tag{22}$$

in the sense that the ξ which solves (22) also solves (21).

Specifically, when we have the regularized quality functional, $d(\xi)$ is quadratic, and hence we obtain an optimization problem which has a mix of linear, quadratic and semidefinite constraints.

Corollary 13 *Let H, c, A and a be as in Proposition 12, and $\Sigma \succeq 0$. Then the solution ξ^* to the optimization problem*

$$\begin{array}{lll} \underset{\xi}{\text{minimize}}\,\underset{x}{\text{maximize}} & -\frac{1}{2}x^\top H(\xi)x - c(\xi)^\top x + \frac{1}{2}\xi^\top \Sigma \xi \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & \xi \geqslant 0 \end{array} \tag{23}$$

can be found by solving the semidefinite programming problem

$$\begin{array}{ll} \underset{t,t',\xi,\gamma}{\text{minimize}} & \frac{1}{2}t + \frac{1}{2}t' + a^\top \gamma \\ \textit{subject to} & \gamma \geqslant 0 \\ & \xi \geqslant 0 \\ & \|\Sigma^{\frac{1}{2}}\xi\|^2 \leqslant t' \\ & \begin{bmatrix} H(\xi) & (A^\top\gamma - c(\xi)) \\ (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{array} \tag{24}$$

Proof By applying proposition 12, and introducing an auxiliary variable t' which upper bounds the quadratic term of ξ, the claim is proved. ■

Comparing the objective function in (21) with (18), we observe that $H(\xi)$ and $c(\xi)$ are linear in ξ. Let $\xi' = \varepsilon\xi$. As we vary ε the constraints are still satisfied, but the objective function scales with ε. Since ξ is the coeffient in the hyperkernel expansion, this implies that we have a set of possible kernels which are just scalar multiples of each other. To avoid this, we add an additional constraint on ξ which is $\mathbf{1}^\top\xi = c$, where c is a constant. This breaks the scaling freedom of the kernel matrix. As a side-effect, the numerical stability of the SDP problems improves considerably. We chose a linear constraint so that it does not add too much overhead to the optimization problem We make one additional simplification of the optimization problem, which is to replace the upper bound of the squared norm ($\|\Sigma^{\frac{1}{2}}\xi\|^2 \leqslant t'$) with and upper bound on the norm ($\|\Sigma^{\frac{1}{2}}\xi\| \leqslant t'$).

In our setting, the regularizer for controlling the complexity of the kernel is taken to be the squared norm of the kernel in the Hyper-RKHS. By looking at the constraints of Equation (24), this is expressed as a bound on the norm ($\|\Sigma^{\frac{1}{2}}\xi\| \leqslant t'$). Comparing this result to the SDP obtained in Lanckriet et al. (2004, Theorem 16), we see that the corresponding regularizer in their setting is $\mathrm{tr}(K) = c$, where c is a constant. Hence the main difference between the two SDPs is the choice of the regularizer for the kernel. However, the motivations of the two methods are different. This paper sets out an induction framework for learning the kernel, and for a particular choice of Q_{emp}, namely the regularized risk functional, we obtain an SDP which has similarities to the approach of Lanckriet et al. (2004). On the other hand, they start out with a transduction problem and derive the optimization problem directly. It is unclear at this point which is the better approach.

From the general framework above (Corollary 13, we derive several examples of machine learning problems, specifically binary classification, regression, and single class (also known as novelty detection) problems. The following examples illustrate our method for simultaneously optimizing over the class of kernels induced by the hyperkernel, as well as the hypothesis class of the machine learning problem. We consider machine learning problems based on kernel methods which are derived from (17). The derivation is essentially by application of Corollary 13 with the two additional conditions above.

6. Examples of Hyperkernel Optimization Problems

In this section, we define the following notation. For $p, q, r \in \mathbb{R}^n, n \in \mathbb{N}$ let $r = p \circ q$ be defined as element by element multiplication, $r_i = p_i \times q_i$ (the Hadamard product, or the $.*$ operation in Matlab). The pseudo-inverse (also known as the Moore-Penrose inverse) of a matrix K is denoted $K^\dagger$. Let $\vec{K}$ be the m^2 by 1 vector formed by concatenating the columns of an m by m matrix. We define the hyperkernel Gram matrix $\underline{K}$ by putting together m^2 of these vectors, that is we set $\underline{K} = [\vec{K}_{pq}]_{p,q=1}^m$. Other notations include: the kernel matrix $K = \text{reshape}(\underline{K}\beta)$ (reshaping a m^2 by 1 vector, $\underline{K}\beta$, to a m by m matrix), $Y = diag(y)$ (a matrix with y on the diagonal and zero everywhere else), $G(\beta) = YKY$ (the dependence on β is made explicit), $\mathbf{I}$ the identity matrix, $\mathbf{1}$ a vector of ones and $\mathbf{1}_{m\times m}$ a matrix of ones. Let w be the weight vector and b_{offset} the bias term in feature space, that is the hypothesis function in feature space is defined as $g(x) = w^\top \phi(x) + b_{offset}$ where $\phi(\cdot)$ is the feature mapping defined by the kernel function k.

The number of training examples is assumed to be m, that is $X_{\text{train}} = \{x_1, \ldots, x_m\}$ and $Y_{\text{train}} = y = \{y_1, \ldots, y_m\}$. Where appropriate, γ and χ are Lagrange multipliers, while η and ξ are vectors of Lagrange multipliers from the derivation of the Wolfe dual for the SDP, β are the hyperkernel coefficients, t_1 and t_2 are the auxiliary variables. When $\eta \in \mathbb{R}^m$, we define $\eta \geqslant 0$ to mean that each $\eta_i \geqslant 0$ for $i = 1, \ldots, m$.

We derive the corresponding SDP for the case when Q_{emp} is a C-SVM (Example 7). Derivations of the other examples follow the same reasoning, and are omitted.

Example 7 (Linear SVM (C-parameterization)) *A commonly used support vector classifier, the C-SVM (Bennett and Mangasarian, 1992, Cortes and Vapnik, 1995) uses an ℓ_1 soft margin, $l(x_i, y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$, which allows errors on the training set. The parameter C is given by the user. Setting the quality functional $Q_{\text{emp}}(k, X, Y) = \min_{f\in\mathcal{H}} \frac{C}{m} \sum_{i=1}^m l(x_i, y_i, f(x_i)) + \frac{1}{2}\|w\|^2_{\mathcal{H}}$,*

$$\begin{array}{ll} \min\limits_{k\in\underline{\mathcal{H}}} \min\limits_{f\in\mathcal{H}_k} & \dfrac{C}{m}\sum\limits_{i=1}^m \zeta_i + \dfrac{1}{2}\|f\|^2_{\mathcal{H}_k} + \dfrac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}} \\ \textit{subject to} & y_i f(x_i) \geqslant 1 - \zeta_i \\ & \zeta_i \geqslant 0 \end{array} \tag{25}$$

Recall the dual form of the C-SVM,

$$\begin{array}{ll} \max\limits_{\alpha\in\mathbb{R}^m} & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \textit{subject to} & \sum_{i=1}^m \alpha_i y_i = 0 \end{array}$$

$$0 \leqslant \alpha_i \leqslant \tfrac{C}{m} \text{ for all } i = 1, \ldots, m.$$

By considering the optimization problem dependent on f in (25), we can use the derivation of the dual problem of the standard C-SVM. Observe that we can rewrite $\|k\|^2_{\mathfrak{H}} = \beta^\top \underline{K}\beta$ due to the representer theorem for hyperkernels. Substituting the dual C-SVM problem into (25), we get the following matrix equation,

$$\begin{array}{ll} \min\limits_{\beta}\max\limits_{\alpha} & \mathbf{1}^\top\alpha - \frac{1}{2}\alpha^\top G(\beta)\alpha + \frac{\lambda_Q}{2}\beta^\top \underline{K}\beta \\ \text{subject to} & \alpha^\top y = 0 \\ & 0 \leqslant \alpha \leqslant \frac{C}{m} \\ & \beta \geqslant 0 \end{array} \tag{26}$$

This is of the quadratic form of Corollary 13 where $x = \alpha$, $\theta = \beta$, $H(\theta) = G(\beta)$, $c(\theta) = -\mathbf{1}$, $\Sigma = C\lambda_Q \underline{K}$, the constraints are $A = \left[\begin{array}{cccc} y & -y & \mathbf{I} & -\mathbf{I} \end{array}\right]^\top$ and $a = \left[\begin{array}{cccc} 0 & 0 & \mathbf{0} & \frac{C}{m}\mathbf{1} \end{array}\right]^\top$. Applying Corollary 13, we obtain the corresponding SDP.

The proof of Proposition 12 uses the Lagrange method. As an illustration of how this proof proceeds, we derive it for this special case of the C-SVM. The Lagrangian associated with (26) is

$$\mathfrak{L}(\alpha, \beta, \gamma, \eta, \xi) = \mathbf{1}^\top\alpha - \frac{1}{2}\alpha^\top G(\beta)\alpha + \frac{\lambda_Q}{2}\beta^\top \underline{K}\beta + \gamma y^\top\alpha + \eta^\top\alpha - \xi^\top(\alpha - \frac{C}{m}\mathbf{1}),$$

where $\beta \geqslant 0, \eta \geqslant 0, \xi \geqslant 0$. The minimum is achieved at

$$\alpha = G(\beta)^\dagger(\gamma y + \mathbf{1} + \eta - \xi),$$

and the corresponding dual optimization problem is

$$\underset{\beta,\gamma,\eta,\xi}{\text{minimize}} \frac{1}{2} z^\top G(\beta)^\dagger z + \frac{C}{m}\xi^\top\mathbf{1} + \frac{\lambda_Q}{2}\beta^\top \underline{K}\beta,$$

where $z = \gamma y + \mathbf{1} + \eta - \xi$. From this point, we replace the quadratic terms with auxiliary variables t_1 and t_2, and apply the Schur complement lemma (Albert, 1969). The resulting SDP after replacing $\|\underline{K}^{\frac{1}{2}}\beta\|^2 \leqslant t_2$ by $\|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2$, and introducing the scale breaking constraint $\mathbf{1}^\top\beta = 1$ is

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2}t_1 + \frac{C}{m}\xi^\top\mathbf{1} + \frac{\lambda_Q}{2}t_2 \\ \text{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \left[\begin{array}{cc} G(\beta) & z \\ z^\top & t_1 \end{array}\right] \succeq 0. \end{array} \tag{27}$$

Note that the value of the support vector coefficients, α, which optimizes the corresponding Lagrange function is $G(\beta)^\dagger z$, and the classification function, $f = sign(K(\alpha \circ y) - b_{offset})$, is given by $f = sign(KG(\beta)^\dagger(y \circ z) - \gamma)$.

Example 8 (Linear SVM (ν-parameterization)) *An alternative parameterization of the ℓ_1 soft margin was introduced by Schölkopf et al. (2000), where the user defined parameter $\nu \in [0,1]$ controls the fraction of margin errors and support vectors. Using ν-SVM as $Q_{\rm emp}$, that is, for a given ν, $Q_{\rm emp}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^m \zeta_i + \frac{1}{2}\|w\|^2_{\mathcal{H}} - \nu\rho$ subject to $y_i f(x_i) \geqslant \rho - \zeta_i$ and $\zeta_i \geqslant 0$ for all $i = 1,\ldots,m$, the corresponding SDP is given by*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2}t_1 - \chi\nu + \xi^\top \frac{1}{m} + \frac{\lambda_Q}{2}t_2 \\ \text{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \left[\begin{array}{cc} G(\beta) & z \\ z^\top & t_1 \end{array}\right] \succeq 0 \end{array} \tag{28}$$

where $z = \gamma y + \chi\mathbf{1} + \eta - \xi$.

The value of α which optimizes the corresponding Lagrange function is $G(\beta)^\dagger z$, and the classification function, $f = sign(K(\alpha\circ y) - b_{offset})$, is given by $f = sign(KG(\beta)^\dagger(y\circ z) - \gamma)$.

Example 9 (Quadratic SVM or Lagrangian SVM) *Instead of using an ℓ_1 loss class, Mangasarian and Musicant (2001) use an ℓ_2 loss class,*

$$l(x_i, y_i, f(x_i)) = \left\{\begin{array}{ll} 0 & \text{if } y_i f(x_i) \geqslant 1 \\ (1 - y_i f(x_i))^2 & \text{otherwise} \end{array}\right. ,$$

and regularized the weight vector as well as the bias term. The empirical quality functional derived from this is $Q_{\rm emp}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^m \zeta_i^2 + \frac{1}{2}(\|w\|^2_{\mathcal{H}} + b^2_{offset})$ subject to $y_i f(x_i) \geqslant 1 - \zeta_i$ and $\zeta_i \geqslant 0$ for all $i = 1,\ldots,m$. The resulting dual SVM problem has fewer constraints, as is evidenced by the smaller number of Lagrange multipliers needed in the corresponding SDP below.

$$\begin{array}{ll} \underset{\beta,\eta}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\lambda_Q}{2}t_2 \\ \text{subject to} & \eta \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \left[\begin{array}{cc} H(\beta) & (\eta + \mathbf{1}) \\ (\eta+\mathbf{1})^\top & t_1 \end{array}\right] \succeq 0 \end{array} \tag{29}$$

where $H(\beta) = Y(K + \mathbf{1}_{m\times m} + \lambda m\mathbf{I})Y$, and $z = \gamma\mathbf{1} + \eta - \xi$.

The value of α which optimizes the corresponding Lagrange function is $H(\beta)^\dagger(\eta+\mathbf{1})$, and the classification function, $f = sign(K(\alpha\circ y) - b_{offset})$, is given by $f = sign(KH(\beta)^\dagger((\eta + \mathbf{1})\circ y) + y^\top(H(\beta)^\dagger(\eta+\mathbf{1})))$.

Example 10 (Single class SVM or Novelty Detection) *For unsupervised learning, the single class SVM computes a function which captures regions in input space where the probability density is in some sense large (Schölkopf et al., 2001). A suitable quality functional $Q_{\rm emp}(k,X,Y) = \min_{f\in\mathcal{H}} \frac{1}{\nu m}\sum_{i=1}^m \zeta_i + \frac{1}{2}\|w\|^2_{\mathcal{H}} - \rho$ subject to $f(x_i) \geqslant \rho - \zeta_i$, and $\zeta_i \geqslant 0$ for*

all $i = 1, \ldots, m$, *and* $\rho \geqslant 0$. *The corresponding SDP for this problem is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2}t_1 + \xi^\top \frac{1}{\nu m} - \gamma + \frac{\lambda_Q}{2\nu}t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2 \\ & \left[\begin{array}{cc} K & z \\ z^\top & t_1 \end{array}\right] \succeq 0 \end{array} \tag{30}$$

where $z = \gamma\mathbf{1} + \eta - \xi$, *and* $\nu \in [0, 1]$ *is a user selected parameter controlling the proportion of the data to be classified as novel.*

The score to be used for novelty detection is given by $f = K\alpha - b_{offset}$, *which reduces to* $f = \eta - \xi$, *by substituting* $\alpha = K^\dagger(\gamma\mathbf{1} + \eta - \xi)$, $b_{offset} = \gamma\mathbf{1}$ *and* $K = reshape(\underline{K}\beta)$.

Example 11 (ν-Regression) *We derive the SDP for* ν *regression (Schölkopf et al., 2000), which automatically selects the* ε *insensitive tube for regression. As in the* ν*-SVM case in Example 8, the user defined parameter* ν *controls the fraction of errors and support vectors. Using the* ε*-insensitive loss,* $l(x_i, y_i, f(x_i)) = \max(0, |y_i - f(x_i)| - \varepsilon)$, *and the* ν*-parameterized quality functional,* $Q_{\text{emp}}(k, X, Y) = \min_{f \in \mathcal{H}} C\left(\nu\varepsilon + \frac{1}{m}\sum_{i=1}^m (\zeta_i + \zeta_i^*)\right)$ *subject to* $f(x_i) - y_i \leqslant \varepsilon - \zeta_i$, $y_i - f(x_i) \leqslant \varepsilon - \zeta_i^*$, $\zeta_i^{(*)} \geqslant 0$ *for all* $i = 1, \ldots, m$ *and* $\varepsilon \geqslant 0$, *the corresponding SDP is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\chi\nu}{\lambda} + \xi^\top \frac{1}{m\lambda} + \frac{\lambda_Q}{2\lambda}t_2 \\ \textit{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = \text{stddev}(Y_{train}) \\ & \left[\begin{array}{cc} F(\beta) & z \\ z^\top & t_1 \end{array}\right] \succeq 0 \end{array}, \tag{31}$$

where $z = \left[\begin{array}{c} -y \\ y \end{array}\right] - \gamma\left[\begin{array}{c} \mathbf{1} \\ -\mathbf{1} \end{array}\right] + \eta - \xi - \chi\left[\begin{array}{c} \mathbf{1} \\ \mathbf{1} \end{array}\right]$ *and* $F(\beta) = \left[\begin{array}{cc} K & -K \\ -K & K \end{array}\right]$.

The Lagrange function is minimized for $\alpha = F(\beta)^\dagger z$, *and substituting into* $f = K\alpha - b_{offset}$, *we obtain the regression function* $f = \left[\begin{array}{cc} -K & K \end{array}\right] F(\beta)^\dagger z - \gamma$.

Example 12 (Kernel Target Alignment) *For the Kernel Target Alignment approach (Cristianini et al., 2002),* $Q_{\text{emp}} = tr(Kyy^\top) = y^\top Ky$, *we directly minimize the regularized quality functional, obtaining the following optimization problem (Lanckriet et al., 2002),*

$$\begin{array}{ll} \underset{\beta}{\text{minimize}} & \frac{1}{2}t_1 + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2, \mathbf{1}^\top\beta = 1 \\ & \left[\begin{array}{cc} K & y \\ y^\top & t_1 \end{array}\right] \succeq 0. \end{array} \tag{32}$$

Note that for the case of Kernel Target Alignment, Q_{emp} *does not provide a direct formulation for the hypothesis function, but instead, it determines a kernel matrix* K. *This kernel matrix,* K, *can be utilized in a traditional SVM, to obtain a classification function.*

7. Experiments

In the following experiments, we use data from the UCI repository. Where the data attributes are numerical, we *did not perform any preprocessing* of the data. Boolean attributes are converted to $\{-1, 1\}$, and categorical attributes are arbitrarily assigned an order, and numbered $\{1, 2, \ldots\}$. The optimization problems in Section 6 were solved with an approximate hyperkernel matrix as described in Section 7.1. The SDPs were solved using SeDuMi (Sturm, 1999), and YALMIP (Löfberg, 2002) was used to convert the equations into standard form. We used the hyperkernel for automatic relevance determination defined by (14) for the hyperkernel optimization problems. The scaling freedom that (14) provides for each dimension means we do not have to normalize data to some arbitrary distribution.

For the classification and regression experiments, the datasets were split into 100 random permutations of 60% training data and 40% test data. We deliberately did not attempt to tune parameters and instead made the following choices uniformly for all datasets in classification, regression and novelty detection:

- The kernel width σ_i, for each dimension, was set to 50 times the 90% quantile of the value of $|x_i - x_j|$ over the training data. This ensures sufficient coverage without having too wide a kernel. This value was estimated from a 20% random sampling of the training data.
- λ was adjusted so that $\frac{1}{\lambda m} = 100$ (that is $C = 100$ in the Vapnik-style parameterization of SVMs). This has commonly been reported to yield good results.
- $\nu = 0.3$ for classification and regression. While this is clearly suboptimal for many datasets, we decided to choose it beforehand to avoid having to change *any* parameter. Clearly we could use previous reports on generalization performance to set ν to this value for better performance. For novelty detection, $\nu = 0.1$ (see Section 7.6 for details).
- λ_h for the Harmonic Hyperkernel was chosen to be 0.6, giving adequate coverage over various kernel widths in (12) (small λ_h emphasizes wide kernels almost exclusively, λ_h close to 1 will treat all widths equally).
- The hyperkernel regularization constant was set to $\lambda_Q = 1$.
- For the scale breaking constraint $\mathbf{1}^\top \beta = c$, c was set to 1 for classification as the hypothesis class only involves the sign of the trained function, and therefore is scale free. However, for regression, $c := \text{stddev}(Y_{\text{train}})$ (the standard deviation of the training labels) so that the hyperkernel coefficients are of the same scale as the output (the constant offset b_{offset} takes care of the mean).

In the following experiments, the hypothesis function is computed using the variables of the SDP. In certain cases, numerical problems in the SDP optimizer or in the pseudo-inverse may prevent this hypothesis from optimizing the regularized risk for the particular kernel matrix. In this case, one can use the kernel matrix K from the SDP and obtain the hypothesis function via a standard SVM.

7.1 Low Rank Approximation

Although the optimization of (17) has reduced the problem of optimizing over two possibly infinite dimensional Hilbert spaces to a finite problem, it is still formidable in practice as

there are m^2 coefficients for β. For an explicit expansion of type (15) one can optimize in the expansion coefficients $k_i(\underline{x})k_i(\underline{x}')$ directly, which leads to a quality functional with an ℓ_2 penalty on the expansion coefficients. Such an approach is appropriate if there are few terms in (15).

In the general case (or if the explicit expansion has many terms), one can use a low-rank approximation, as described by Fine and Scheinberg (2001) and Zhang (2001). This entails picking from $\left\{\underline{k}((x_i, x_j), \cdot)|1 \leq i, j \leq m^2\right\}$ a small fraction of terms, p (where $m^2 \gg p$), which approximate $\underline{k}$ on $X_{\text{train}} \times X_{\text{train}}$ sufficiently well. In particular, we choose an $m \times p$ truncated lower triangular matrix G such that $\|P\underline{K}P^\top - GG^\top\|_F \leqslant \delta$, where P is the permutation matrix which sorts the eigenvalues of $\underline{K}$ into decreasing order, and δ is the level of approximation needed. The norm, $\|\cdot\|_F$ is the Frobenius norm. In the following experiments, the hyperkernel matrix was approximated to $\delta = 10^{-6}$ using the incomplete Cholesky factorization method (Bach and Jordan, 2002).

7.2 Classification Experiments

Several binary classification datasets[1] from the UCI repository were used for the experiments. A set of synthetic data (labeled syndata in the results) sampled from two Gaussians was created to illustrate the scaling freedom between dimensions. The first dimension had a standard deviation of 1000 whereas the second dimension had a standard deviation of 1 (a sample result is shown in Figure 1). The results of the experiments are shown in Table 3.

From Table 3, we observe that our method achieves state of the art results for all the datasets, except the "heart" dataset. We also achieve results much better than previously reported for the "credit" dataset. Comparing the results for C-SVM and Tuned SVM, we observe that our method is always equally good, or better than a C-SVM tuned using 10-fold cross validation.

7.3 Effect of λ_Q and λ_h on Classification Error

In order to investigate the effect of varying the hyperkernel regularization constant, λ_Q, and the Harmonic Hyperkernel parameter, λ_h, we performed experiments using the C-SVM hyperkernel optimization (Example 7). We performed two sets of experiments with each of our chosen datasets. The results shown in Table 4.

From Table 4, we observe that the variation in classification accuracy over the whole range of the hyperkernel regularization constant, λ_Q is less than the standard deviation of the classification accuracies of the various datasets (compare with Table 3). This demonstrates that our method is quite insensitive to the regularization parameter over the range of values tested for the various datasets.

The method shows a higher sensitivity to the harmonic hyperkernel parameter. Since this parameter effectively selects the scale of the problem, by selecting the "width" of the kernel, it is to be expected that each dataset would have a different ideal value of λ_h. It is to be noted that the generalization accuracy at $\lambda_h = 0.6$ is within one standard deviation (see Table 3 and 4) of the best accuracy achieved over the whole range tested.

1. We classified window vs. non-window for glass data, the other datasets are all binary.

Data	C-SVM	ν-SVM	Lag-SVM	Best other	CV Tuned SVM (C)
syndata	2.8±2.4	**1.9±1.9**	2.4±2.2	NA	5.9±5.4 (10^8)
pima	**23.5±2.0**	27.7±2.1	23.6±1.9	23.5	24.1±2.1 (10^4)
ionosph	6.6±1.8	6.7±1.8	6.4±1.9	**5.8**	6.1±1.8 (10^3)
wdbc	3.3±1.2	3.8±1.2	**3.0±1.1**	3.2	5.2±1.4 (10^6)
heart	19.7±3.3	19.3±2.4	20.1±2.8	**16.0**	23.2±3.7 (10^4)
thyroid	7.2±3.2	10.1±4.0	6.2±3.1	**4.4**	5.2±2.2 (10^5)
sonar	14.8±3.7	15.3±3.7	**14.7±3.6**	15.4	15.3±4.1 (10^3)
credit	14.6±1.8	**13.7±1.5**	14.7±1.8	22.8	15.3±2.0 (10^8)
glass	6.0±2.4	8.9±2.6	**6.0±2.2**	NA	7.2±2.7 (10^3)

Table 3: Hyperkernel classification: Test error and standard deviation in percent. The second, third and fourth columns show the results of the hyperkernel optimizations of C-SVM (Example 7), ν-SVM (Example 8) and Lagrangian SVM (Example 9) respectively. The results in the fifth column shows the best results from (Freund and Schapire, 1996, Rätsch et al., 2001, Meyer et al., 2003). The rightmost column shows a C-SVM tuned in the traditional way. A Gaussian RBF kernel was tuned using 10-fold cross validation on the training data, with the best value of C shown in brackets. A grid search was performed on (C, σ). The values of C tested were $\{10^{-2}, 10^{-1}, \ldots, 10^{9}\}$. The values of the kernel width, σ, tested were between 10% and 90% quantile of the distance between a pair of sample of points in the data. These quantiles were estimated by a random sample of 20% of the training data.

	λ_h		λ_Q	
Data	Error	Deviation	Error	Deviation
syndata	3.0±1.1	2.2	2.8±0.0	2.2
pima	25.7±2.6	1.9	24.5±0.1	1.5
ionosph	6.6±1.0	1.7	7.2±0.1	1.9
wdbc	2.9±0.4	0.9	2.7±0.2	0.8
heart	19.7±2.0	3.0	19.4±0.9	2.8
thyroid	6.5±2.8	3.0	6.7±0.3	3.7
sonar	15.7±1.6	3.4	15.1±0.2	3.3
credit	16.0±1.8	1.6	14.7±0.4	1.6
glass	5.9±1.0	2.3	5.2±0.3	2.3

Table 4: Effect of varying λ_h and λ_Q on classification error. In the left experiment, we fixed $\lambda_Q = 1$, and λ_h was varied with the values $\lambda_h = \{0.1, 0.2, \ldots, 0.9, 0.92, 0.94, 0.96, 0.98\}$.In the right, we set $\lambda_h = 0.6$ and varied $\lambda_Q = \{10^{-4}, 10^{-3}, \ldots, 10^{5}\}$. The error columns (columns 2 and 4) report the average error on the test set and the standard deviation of the error over the different parameter settings. The deviation columns (columns 3 and 5) report the average standard deviation over 10 random 60%/40% splits.

7.4 Computational Time

One of the concerns of an SDP optimization problem is the computational complexity. Instead of performing worst case analysis of computational complexity, we perform an empirical test to investigate the scaling behaviour of the proposed method. The total computation time for the first 10 splits of the data was measured, and the average time taken for each split was computed and plotted on a log scale plot in Figure 4. The slope of the graph demonstrates that we have an approximately cubic scaling in computational time.

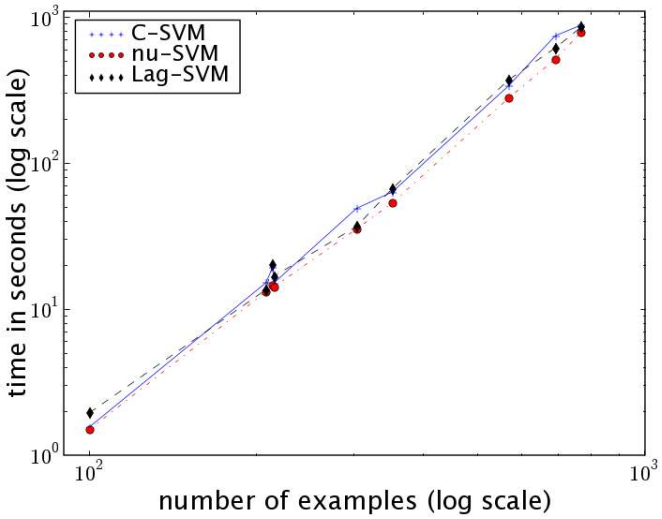


Figure 4: A log scale plot of computational time (in seconds), measured using MATLAB's cputime, against the number of examples in the respective datasets. The slope of the least squares fit through the points are 3.13, 3.05 and 3.03 for C-SVM (Example 7), ν-SVM (Example 8) and Lag-SVM (Example 9) respectively, demonstrating that the algorithms have approximately cubic scaling.

7.5 Regression Experiments

In order to demonstrate that we can solve problems other than binary classification using the same framework, we performed some experiments using regression and novelty detection datasets. The results of the regression experiments are shown in Table 5. We used *the same parameter settings* as in the previous section.

Comparing the second and fourth columns, we observe that the hyperkernel optimization problem performs better than a ε-SVR tuned using cross validation for all the datasets except the servo dataset. Meyer et al. (2003) used a 90%/10% split of the data for their experiments, while we used a 60%/40% split, which may account for the better performance in the cpu and servo datasets. The reason for the much better rate on the "auto imports" dataset remains a mystery.

Data	ν-SVR	Best other	CV Tuned ε-SVR
auto-mpg	7.83±0.96	7.11	9.47±1.55
boston	12.96±3.38	9.60	15.78±4.30
auto imports($\times 10^6$)	5.91±2.41	0.25	7.51±5.33
cpu($\times 10^3$)	4.41±3.64	3.16	12.02±20.73
servo	0.74±0.26	0.25	0.62±0.25

Table 5: Hyperkernel regression: Mean Squared Error. The second column shows the results from the hyperkernel optimization of the ν-regression (Example (11)). The results in the third column shows the best results from (Meyer et al., 2003). The rightmost column shows a ε-SVR with a gaussian kernel tuned using 10-fold cross validation on the training data. Similar to the classification setting, grid search was performed on (C, σ). The values of C tested were $\{10^{-2}, 10^{-1}, \ldots, 10^{9}\}$. The values of the kernel width, σ, tested were between the 10% and 90% quantiles of the distance between a pair of sample of points in the data. These quantiles were estimated by a random 20% sample of the training data.

7.6 Novelty Detection

We applied the single class support vector machine to detect outliers in the USPS data. The test set of the default split in the USPS database was used in the following experiments. The parameter ν was set to 0.1 for these experiments, hence selecting up to 10% of the data as outliers.

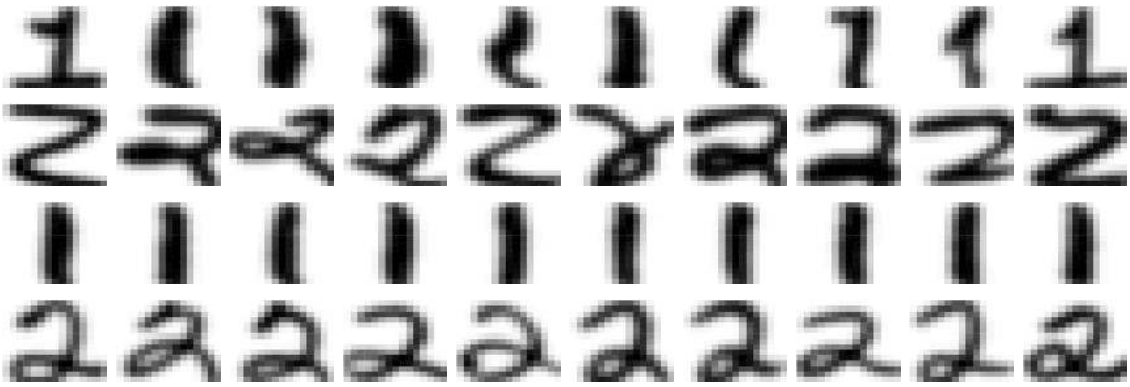


Figure 5: Top rows: Images of digits '1' and '2', considered novel by algorithm; Bottom: typical images of digits '1' and '2'.

Since there is no quantitative method for measuring the performance of novelty detection, we cannot directly compare our results with the traditional single class SVM. We can only subjectively conclude, by visually inspecting a sample of the digits, that our approach works for novelty detection of USPS digits. Figure 5 shows a sample of the digits. We can

see that the algorithm identifies 'novel' digits, such as in the top two rows of Figure 5. The bottom two rows shows a sample of digits which have been deemed to be 'common'.

8. Summary and Outlook

The regularized quality functional allows the systematic solution of problems associated with the choice of a kernel. Quality criteria that can be used include Kernel Target Alignment, regularized risk and the log posterior. The regularization implicit in our approach allows the control of overfitting that occurs if one optimizes over a too large a choice of kernels.

We have shown that when the empirical quality functional is the regularized risk functional, the resulting optimization problem is convex, and in fact is a SDP. This SDP, which learns the best kernel given the data, has a Bayesian interpretation in terms of a hierarchical Gaussian process. We define more general kernels which may have many free parameters, and optimize over them without overfitting. The experimental results on classification demonstrate that it is possible to achieve state of the art performance using our approach with no manual tuning. Furthermore, the same framework and parameter settings work for various datasets as well as regression and novelty detection.

This approach makes support vector based estimation approaches more automated. Parameter adjustment is less critical compared to when the kernel is fixed, or hand tuned. Future work will focus on deriving improved statistical guarantees for estimates derived via hyperkernels which match the good empirical performance.

Acknowledgements The authors would like to thank Stéphane Canu, Laurent El Ghaoui, Michael Jordan, John Lloyd, Daniela Pucci de Farias, Matthias Seeger, Grace Wahba and the referees for their helpful comments and suggestions. The authors also thank Alexandros Karatzoglou for his help with SVLAB. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverses. *SIAM Journal on Applied Mathematics*, 17(2):434 – 440, 1969.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337 – 404, 1950.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1 – 48, 2002.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23 – 34, 1992.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

O. Bousquet and D. Herrmann. On the complexity of learning the kernel matrix. In *Advances in Neural Information Processing Systems 15*, pages 399–406, 2002.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121 – 167, 1998.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131 – 159, 2002.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676 – 1695, 1990.

K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems 15*, pages 537–544, 2002.

N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 367 – 373, Cambridge, MA, 2002. MIT Press.

N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On optimizing kernel alignment. Technical report, UC Davis Department of Statistics, 2003.

K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41 – 59, 2003.

S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243 – 264, Dec 2001. http://www.jmlr.org.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learing*, pages 148 – 146. Morgan Kaufmann Publishers, 1996.

D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99 - 10, Computer Science Department, UC Santa Cruz, 1999.

R. Herbrich and R.C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3:175 – 212, 2002.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82 – 95, 1971.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the International Conference on Machine Learning*, pages 323–330. Morgan Kaufmann, 2002.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27 – 72, 2004.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

J. Löfberg. YALMIP, yet another LMI parser, 2002. http://www.control.isy.liu.se/ ˜johanl/yalmip.html.

A. Luntz and V. Brailovsky. On estimation of characters obtained in statistical procedure of recognition (in Russian). *Technicheskaya Kibernetica*, 3, 1969.

D. J. C. MacKay. Bayesian non-linear modelling for the energy prediction competition. *ASHRAE Transcations*, 4:448 – 472, 1994.

O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161 – 177, 2001.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55(1–2):169–186, 2003.

R. Neal. *Bayesian Learning in Neural Networks*. Springer, 1996.

C. S. Ong and A. J. Smola. Machine learning using hyperkernels. In *Proceedings of the International Conference on Machine Learning*, pages 568–575, 2003.

C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In *Neural Information Processing Systems*, volume 15, pages 495–502. MIT Press, 2002.

M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311 – 326, Cambridge, MA, 2000. MIT Press.

G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for adaboost. *Machine Learning*, 42 (3):287 – 320, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207 – 1245, 2000.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

M. Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Edinburgh, Division of Informatics, 1999.

A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(5):637 – 649, 1998.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11/12(1 - 4):625 – 653, 1999.

K. Tsuda, S. Akaho, and K. Asai. The EM algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research*, 4:67–81, 2003.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review.*, 38(1):49 – 95, 1996.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514 – 520, Cambridge, MA, 1996. MIT Press.

Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 20(12): 1342 – 1351, 1998.

T. Zhang. Some sparse approximation bounds for regression problems. In *Proc. 18th International Conf. on Machine Learning*, pages 624 – 631. Morgan Kaufmann, San Francisco, CA, 2001.