Journal of Machine Learning Research 3 (2003) xxxx Submitted xx/xx; Published xx/xx

Learning the Kernel with Hyperkernels

Cheng Soon Ong Cheng.Ong@anu.edu.au
Alexander J. Smola Alex.Smola@anu.edu.au
Machine Learning Group, Research School of Information Sciences and Engineering,
Australian National University
Canberra, ACT 0200, Australia

Robert C. Williamson Bob.Williamson@anu.edu.au
National ICT Australia and Research School of Information Sciences and Engineering,
Australian National University
Canberra, ACT 0200, Australia

Editor: U.N. Known (joint publication with www.kernel-machines.org)

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 very much akin 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 (e.g. (Burges, 1998)) and books (e.g. (Schölkopf and Smola, 2002)).

Choosing a suitable kernel function, and therefore a feature mapping, is imperative to the success of this inference process. To date, there are few systematic techniques to assist in this choice. Even the restricted problem of choosing the "width" of a parameterized family of kernels (e.g. Gaussian kernel) has not had a simple and elegant solution.

There are still no general principles to guide the choice of a) which family of kernels to choose, b) efficient parameterizations over this space, and c) suitable penalty terms to combat overfitting. Whilst not yet providing a complete solution to these problems, this paper does present a framework that allows the optimization within a parameterized family relatively simply, and crucially, intrinsically captures the tradeoff between the size of the family of kernels and the sample size available. Furthermore, the solution presented *is* for optimizing kernels themselves, rather than the kernel matrix as done by (Crammer et al., 2002, Lanckriet et al., 2002).

This problem is very much akin to the situation in neural networks 10 years ago, where the choice of the function (not the function class) was also hindered by the three issues mentioned above. This leads to the idea that possibly the kernel trick *on kernels*, may be useful to alleviate the issues.

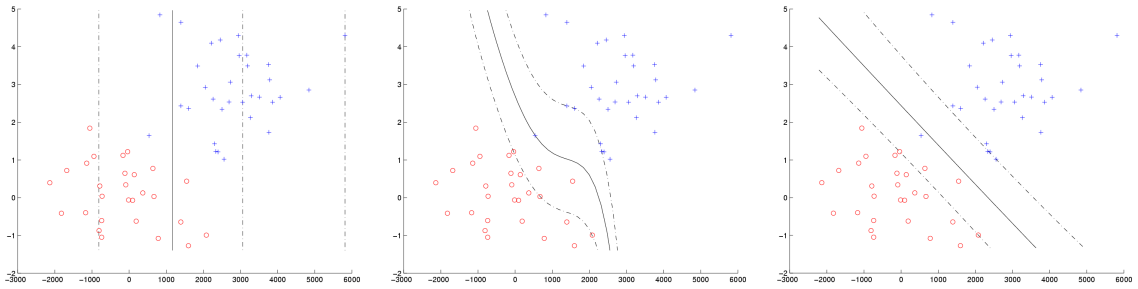
1.1 Motivation

As motivation for the need for methods to learn the kernel function, 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.

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 no-



(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 (top) and an automatic width selection (bottom).

tion of alignment (Cristianini et al., 2002) uses the objective function $y^\top Ky$ 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. Note that the definition in Cristianini et al. (2002) looks somewhat different ($\mathrm{tr}(Kyy^\top)$), yet it is algebraically identical to the definition above. The semidefinite programming (SDP) approach (Lanckriet et al., 2002) 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 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. Table 1 summarizes these approaches. The notation $K \succeq 0$ means that K is positive semidefinite, that is for all $a \in \mathbb{R}^n$, $a^\top Ka \geqslant 0$.

1.3 Outline of the Paper

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. We give several examples of hyperkernels and recipes to construct others (Section 4) and show (Section 5)

Approach	Objective function	Kernel class ($\mathcal{K}$)
Cross Validation	CV Risk	Finite set of kernels
Alignment	$y^\top K y$	$\sum_{i=1}^m \beta_i v_i v_i^\top \vert v_i$ are eigenvectors of K
SDP	margin	$\sum_{i=1}^m \beta_i K_i \vert K_i \succeq 0, \mathrm{tr} K_i = c$
Complexity Bound	margin	$\sum_{i=1}^m \beta_i K_i \vert K_i \succeq 0, \mathrm{tr} K_i = c, \beta_i \geqslant 0$
Boosting	ExpLoss or LogLoss	$\sum_t \beta_t K_t \vert K_t$ maximum eigenvector
Bayesian	neg. log-posterior	dependent on prior

Table 1: Summary of recent approaches to kernel learning

how they can be used in practice via semidefinite programming. Experimental results for classification, regression and novelty detection (Section 6) are shown. Finally, some issues and open problems are discussed (Section 7).

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 $x_i, x_j \in X_{\text{train}}$. We also use $\mathrm{tr} K$ 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*. 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 (e.g., optimal alignment, greatest luckiness, smallest negative log-posterior), based on this one *single* dataset X, Y. Provided a sufficiently rich class of kernels k it is in general possible to find a kernel k^* that attains the minimum of any such Q_{emp} regardless of the data.[1] However, it is very unlikely that $Q_{\text{emp}}(k^*, X, Y)$ would be similarly small for other

1. Note that by quality we actually mean *badness*, as we would like to minimize this quantity.

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, $R_{\text{emp}}(f, X, Y) = \frac{1}{m}\sum_{i=1}^{m} \ell(x_i, y_i, f(x_i))$ (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 the empirical estimate of the expected loss $R(f) = \mathbf{E}_{x,y}[\ell(x, y, f(x))]$, 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.

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 (and possibly other terms)

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

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 (**?**, 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 (in many cases below we use the labels), we can arbitrarily minimize the empirical quality functional Q_{emp}.

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

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

where $\|f\|_{\mathcal{H}}^2$ is the RKHS norm of f. 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 cost functional ℓ:

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

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 $Q_{\mathrm{emp}}^{\mathrm{regrisk}}$ is bounded from below by 0. Hence, it is sufficient if we can find a (possibly) suboptimal pair (α, k) for which $Q_{\mathrm{emp}}^{\mathrm{regrisk}} \leq \epsilon$ 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 $c(x_i, y_i, f(x_i)) = 0$ and therefore $Q_{\mathrm{emp}}^{\mathrm{regrisk}}(k, X_{\mathrm{train}}, Y_{\mathrm{train}}) = \frac{\lambda}{2\beta}$. For sufficiently large β we can make $Q_{\mathrm{emp}}^{\mathrm{regrisk}}(k, X_{\mathrm{train}}, Y_{\mathrm{train}})$ arbitrarily close to 0.*
- *Even if we disallow setting K arbitrarily close to zero by setting $\mathrm{tr}K = 1$, 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*

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

Choosing each $z_i = \mathrm{argmin}_\zeta\, \ell(x_i, y_i, \zeta) + \frac{\lambda}{2}\zeta^2$, where ζ comes from the class of possible hypotheses, 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 $Q_{\mathrm{emp}}^{\mathrm{regrisk}}$.

Example 2 (Negative Log-Posterior) *This functional is similar to R_{reg}, as it includes a regularization term (in this case the negative log prior), a loss term (the negative log-likelihood), and additionally, the log-determinant of K (Schölkopf and Smola, 2002, Chapter 16). The latter measures the size of the space spanned by K. This leads to the following quality functional:*

$$Q_{\mathrm{emp}}^{\mathrm{logpost}}(k, X_{\mathrm{train}}, Y_{\mathrm{train}}) := \min_{f\in\mathbb{R}^m}\left[-\log p(y_i|x_i, f_i) + \frac{1}{2}f^\top K^{-1}f + \frac{1}{2}\log\det K\right] \tag{4}$$

Note that $Q_{\mathrm{emp}}^{\mathrm{logpost}}$ is not bounded from below in general. Any K which does not have full rank will send (4) to $-\infty$, hence Q_{emp} is minimized trivially. If we fix the determinant of K to be some constant to ensure that K is full rank, we can set

$$K = \beta\|y\|^{-2}yy^\top + \beta^{-\frac{1}{m-1}}(\mathbf{1} - \|y\|^{-2}yy^\top)$$

which leads to $|K| = 1$. *Under the assumption that the minimum of* $-\log p(y_i, x_i, f_i)$ *with respect to* f_i *is attained at* $f_i = y_i$, *we can see that* $\beta \longrightarrow \infty$ *leads to the overall minimum of* $Q_{\text{emp}}^{\text{logpost}}(k, X_{\text{train}}, Y_{\text{train}})$.

Example 3 (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:*

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

Choosing $K = yy^\top$ *and* $\alpha^i = \frac{1}{\|y^i\|^2} y^i$, *where* α^i *and* y^i *are the vectors with the* i*th element set to zero, we have* $K\alpha = y$. *Hence we can achieve perfect prediction. 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 4 (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*

$$Q_{\text{emp}}^{\text{alignment}}(k, X_{\text{train}}, Y_{\text{train}}) := 1 - \frac{y^\top K y}{\|y\|_2^2 \|K\|_2}. \tag{5}$$

Here $\|y\|_2$ *denotes the* ℓ_2 *norm of the vector of observations and* $\|K\|_2$ *is the Frobenius norm, i.e.,* $\|K\|_2^2 := \operatorname{tr}(KK^\top) = \sum_{i,j} (K_{ij})^2$. *By decomposing* K *into its eigensystem one can see that (5) is minimized, if* $K = yy^\top$, *in which case*

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

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.*

Example 5 (Luckiness for Classification with Kernels) *Recently the concept of algorithmic luckiness (Herbrich and Williamson, 2002) was introduced to assess the quality of an estimate in a sample and algorithm dependent fashion. We define the quality functional for a kernel method to be:*

$$Q_{\text{emp}}^{\text{luckiness}}(k, X_{\text{train}}, Y_{\text{train}}) := \min_{j \in \mathbb{N}} \left\{ j \geqslant \left(\frac{\varepsilon_j(X_{\text{train}}) \|\mathcal{A}(X_{\text{train}}, Y_{\text{train}})\|_1}{\Gamma_{(X_{\text{train}}, Y_{\text{train}})}(w_{\mathcal{A}(X_{\text{train}}, Y_{\text{train}})})} \right)^2 \right\}$$

where ε_i *is the smallest* ε *such that* $\{\phi(x_1), \ldots, \phi(x_n)\}$ *can be covered by at most* i *balls of radius* ε, $\mathcal{A}(X_{\text{train}}, y_{\text{train}})$ *is the* α *vector (dual coefficients) of the maximum margin solution,* $w_{\mathcal{A}(X_{\text{train}}, y_{\text{train}})}$, *is the corresponding weight vector,* ϕ *is the feature mapping corresponding to* k, *and* $\Gamma_{X_{\text{train}}, Y_{\text{train}}}$ *is the normalized margin* $\min_{(x,y) \in (X_{\text{train}}, Y_{\text{train}})} \frac{y_i \langle \phi(x_i), w \rangle}{\|\phi(x_i)\| \|w\|}$

For $K = yy^\top$, *we can cover the feature space by balls of radius* 1, *that is* $\varepsilon_j(X_{\text{train}}) = 1$ *for all* $j \geqslant 2$. *Since the algorithmic luckiness framework depends on the choice of a particular*

algorithm, we have to choose a rule for α. We consider any choice for which $y_i\alpha_i \geq 0$ and $\|\alpha\|_1 = 1$, as is satisfied for SVM, linear programming estimators, and many boosting algorithms. For this choice, the empirical error vanishes with margin 1 and by construction $\|\mathcal{A}(X_{\text{train}}, y_{\text{train}})\|_1 = 1$. Hence, $Q_{\text{emp}}^{\text{luckiness}}(k, X_{\text{train}}, Y_{\text{train}}) = 1$, which is the global minimum.

Example 6 (Radius-Margin Bound) *For SVMs without thresholding and with no training errors, Vapnik (1998) proposed the following upper bound on the generalization error of the classifier in terms of the radius and margin of the SVM (Bartlett and Shawe-Taylor, 1999).*

$$T = \frac{1}{m}\frac{R^2}{\gamma^2}$$

where R and γ are the radius and the margin of the training data. We can define a quality functional:

$$Q_{\text{emp}}^{\text{radius}}(k, X_{\text{train}}, Y_{\text{train}}) = \frac{1}{m}R^2\alpha^\top K\alpha$$

Choosing $K = \beta yy^\top$ and $\alpha = \frac{1}{\beta\|y\|^2}y$, we obtain a bound on the radius $R^2 \leqslant \beta(\max_i y_i^2)$, and an expression for the margin, $\alpha^\top K\alpha = \beta^{-1}$. Therefore $Q_{\text{emp}}^{\text{radius}}(k, X_{\text{train}}, Y_{\text{train}}) \leqslant \frac{\beta^2}{m}$, which can be made arbitrarily close to zero by letting $\beta \longrightarrow 0$.

The above examples illustrate how many 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, and have the special property of being defined 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 $\ell : (\mathcal{X} \times \mathbb{R}^2)^m \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the general regularized risk*

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

admits a representation of the form

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x_i, x). \tag{6}$$

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

3.1 Regularized Quality Functional

Definition 1 allows one to define an RKHS on kernels $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$, simply by introducing the compounded index set, $\underline{\mathcal{X}} := \mathcal{X} \times \mathcal{X}$ and by treating k as functions $k : \underline{\mathcal{X}} \to \mathbb{R}$:

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*

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

   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. $\underline{\mathcal{H}} = \overline{\operatorname{span}\{\underline{k}(\underline{x}, \cdot) | \underline{x} \in \underline{\mathcal{X}}\}}$.*

3. *For any fixed $\underline{x} \in \underline{\mathcal{X}}$ the hyperkernel $\underline{k}$ is a kernel in its second argument; i.e. 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 kernel.*

This is a RKHS with an additional requirement on its elements. In fact, we can have a recursive definition of an RKHS of an RKHS ad infinitum. 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 a kernel in its second argument for any fixed first argument. This condition somewhat

limits the choice of possible kernels but it allows for simple optimization algorithms which consider kernels $k \in \underline{\mathcal{H}}$, which are in the convex cone of $\underline{k}$.

By analogy with the definition of the regularized risk functional (1), we proceed to define the regularized quality functional as

$$Q_{\text{reg}}(k, X, Y) := Q_{\text{emp}}(k, X, Y) + \frac{\lambda_Q}{2}\|k\|^2_{\underline{\mathcal{H}}} \tag{7}$$

where $\lambda_Q > 0$ is a regularization constant and $\|k\|^2_{\underline{\mathcal{H}}}$ denotes the RKHS norm in $\underline{\mathcal{H}}$. 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. The complexity class can be derived from the results of Bousquet and Herrmann (2002). Regularizers other than $\frac{\lambda_Q}{2}\|k\|^2$ are possible, such as L_p penalties. In this paper, we restrict ourselves to the L_2 norm (7). The advantage of (7) is that its minimizer satisfies the representer theorem.

Corollary 6 (Representer Theorem for Hyper-RKHS) *Let $\mathcal{X}$ be a set, Q_{emp} an arbitrary empirical quality functional, and X, Y as defined above, then each minimizer $k \in \underline{\mathcal{H}}$ of the regularized quality functional $Q_{\text{reg}}(k, X, Y)$ admits a representation of the form*

$$k(x, x') = \sum_{i,j}^{m} \alpha^m_{i,j} \underline{k}((x_i, x_j), (x, x')). \text{ for all } x, x' \in X \tag{8}$$

Proof All we need to do is rewrite (7) so that it satisfies the conditions of Theorem 4. Let $\underline{x}_{ij} := (x_i, x_j)$. Then $Q(k, X, Y)$ has the properties of a loss function, as it 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 (8) follows. ■

Corrolary 6 imples 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 (8) is not necessarily positive semidefinite. This is not surprising, since we did not explicitly require this property in the optimization problem. In practice, however, this is clearly not what we want and therefore we need to impose additional constraints of the type $K \succeq 0$ or $k \in \{\text{Mercer Kernel}\}$.

While the latter is almost impossible to enforce directly, the former could be verified directly, hence imposing a constraint only on the values $k(x_i, x_j)$ rather than on k itself. This means that the conditions of the Representer Theorem apply and (8) applies (with suitable constraints on the coefficients $\alpha_{i,j}$).

Another option is to be somewhat more restrictive and require that all expansion coefficients $\alpha_{i,j} \geqslant 0$. 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.

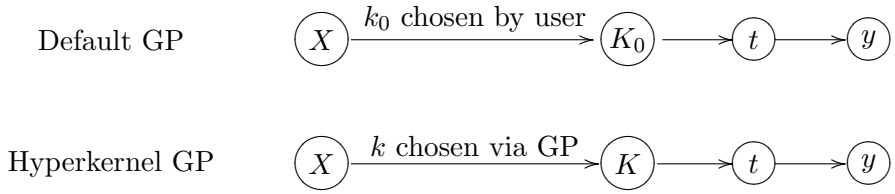


Figure 2: Top: Generative model for Gaussian Process estimation; Bottom: Generative model for Gaussian Process estimation using hyperpriors on k defined by $\underline{k}$.

3.2 A Bayesian Perspective

It is well known that there exists a Gaussian Process (GP) analog to the Support Vector Machine (e.g. (Opper and Winther, 2000)), which is essentially obtained (ignoring normalizing terms) by exponentiating the regularized risk functional used in SVMs.

An analogous insight can be obtained by exponentiating $-Q_{\text{reg}}$, again ignoring normalization terms. In a nutshell, exponentiating $-\frac{1}{2}\|k\|^2$ means that we assume a GP with covariance function $\underline{k}$ to be the underlying distribution for the kernels k, or more precisely, the restriction of such a GP to the cone of covariance functions. In other words, we have two nested GPs to model the data generation scheme. Hence we are studying a mixture of Gaussian processes.

The general scheme is well known in Bayesian estimation and is commonly referred to as hyperpriors, which determine the distribution of the priors (here the GP with covariance k). Figure 2 describes the model: whereas in an ordinary GP, t is drawn from a Gaussian distribution with covariance matrix K_0 and y is conditionally independent given t, in a Hyperkernel-GP we assume that K itself is drawn according to a GP before performing further steps of dependency calculation.

Note that the MAP2 (maximum a posteriori-2) method (MacKay, 1994) 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, and convergence radius R. Then for $k(x, x') \leq \sqrt{R}$,

$$\underline{k}(\underline{x}, \underline{x}') := g(k(\underline{x})k(\underline{x}')) = \sum_{i=0}^{\infty} c_i (k(\underline{x})k(\underline{x}'))^i \tag{9}$$

is a hyperkernel. For $\underline{k}$ to be a hyperkernel, we need to check that firstly, $\underline{k}$ is a kernel, and secondly, for any fixed pair of elements of the input data, $\underline{x}$, the function $\underline{k}(\underline{x},(x,x'))$ is a kernel. 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 $\underline{\Phi}(\underline{x}) := (\sqrt{c_0}, \sqrt{c_1}k^1(\underline{x}), \sqrt{c_2}k^2(\underline{x}), \ldots)$.

Example 7 (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*

$$\underline{k}(\underline{x},\underline{x}') = (1-\lambda_h)\sum_{i=0}^{\infty}\left(\lambda_h k(\underline{x})k(\underline{x}')\right)^i = \frac{1-\lambda_h}{1-\lambda_h k(\underline{x})k(\underline{x}')}. \tag{10}$$

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

$$\underline{k}((x,x'),(x'',x''')) = \frac{1-\lambda_h}{1-\lambda_h \exp\left(-\sigma^2(\|x-x'\|^2 + \|x''-x'''\|^2)\right)}. \tag{11}$$

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 $X \times X$.

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.

$g(\xi)$	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.

However, if we want the kernel to adapt automatically to different widths for each dimension, we need to perform the summation that led to (10) for each dimension in its arguments separately. Such a hyperkernel corresponds to ideas developed in automatic relevance determination (ARD) (MacKay, 1994, Neal, 1996).

Example 8 (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 a diagonal covariance matrix. Take sums over each diagonal entry $\sigma_j = \Sigma_{jj}$ separately to obtain*

$$\begin{aligned}\underline{k}((x,x'),(x'',x''')) &= (1-\lambda_h)\sum_{j=1}^{d}\sum_{i=0}^{\infty}\left(\lambda_h k_\Sigma(x,x')k_\Sigma(x'',x''')\right)^i \\ &= \prod_{j=1}^{d}\frac{1-\lambda_h}{1-\lambda_h\exp\left(-\sigma_j((x_j-x_j')^2 + (x_j''-x_j''')^2)\right)}.\end{aligned} \tag{12}$$

Eq. (12) 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 7 (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 (e.g. Gaussian RBF, exponential RBF, damped harmonic oscillator, generalized B_n spline), can be used to construct a hyperkernel by checking whether their Fourier transform is positive.

4.3 Explicit Expansion

If we have a finite set of kernels that we want to choose from, this results in a hyperkernel which is a finite sum of possible kernel functions. This setting is similar that of Lanckriet et al. (2002).

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

$$\underline{k}(\underline{x}, \underline{x}') := \sum_{i=1}^{n} c_i k_i(\underline{x}) k_i(\underline{x}'), k_i(\underline{x}) \geqslant 0, \forall \underline{x} \tag{13}$$

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 9 (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,*

$$\begin{aligned} \underline{k}((x_1, x_1'), (x_2, x_2')) &= c_1 (\langle x_1, x_1' \rangle + b)^{2p} (\langle x_2, x_2' \rangle + b)^{2p} \\ &\quad + c_2 \exp(-\sigma^2 \|x_1 - x_1'\|^2) \exp(-\sigma^2 \|x_2 - x_2'\|^2) \end{aligned} \tag{14}$$

is a hyperkernel.

5. Optimization Problem

We will now consider the optimization of the quality functionals utilizing hyperkernels. We choose the regularized risk functional as the empirical quality functional, i.e. $Q_{\text{emp}}(k, X, Y) = R_{\text{reg}}(f, X, Y)$. Hence, for a particular loss function $\ell(x_i, y_i, f(x_i))$, we obtain the regularized quality functional.

$$\min_{k \in \underline{\mathcal{H}}} \min_{f \in \mathcal{H}_k} \frac{1}{m} \sum_{i=1}^{m} \ell(x_i, y_i, f(x_i)) + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_k} + \frac{\lambda_Q}{2} \|k\|^2_{\underline{\mathcal{H}}}. \tag{15}$$

By the representer theorem (Theorem 4 and 6) we can write the regularizers as quadratic terms. Using the soft margin loss, we obtain

$$\min_{\beta} \max_{\alpha} \frac{1}{m} \sum i = 1^m \max(0, 1 - y_i f(x_i)) + \frac{\lambda}{2} \alpha^\top K \alpha + \frac{\lambda_Q}{2} \beta^\top \underline{k} \beta \tag{16}$$

where $\alpha \in \mathbb{R}^m$ are the coefficients of the kernel expansion (6), and $\beta \in \mathbb{R}^{m^2}$ are the coefficients of the hyperkernel expansion (8). The corresponding optimization problems can be expressed as a SDP. In general, a SDP would be take longer to solve than a quadratic program (as in a traditional SVM). This reflects the added cost incurred for optimizing over a class of kernels.

5.1 Semidefinite Programming Formulations

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

$$\begin{aligned} \min_x \quad & c^\top x \\ \text{subject to} \quad & F_0 + \sum_{i=1}^{q} x_i F_i \succeq 0 \text{ and } Ax = b \end{aligned}$$

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.

We derive the corresponding SDP for Equation (15). The following proposition allows us to derive a SDP from a class of general quadratic programs. It is a straightforward extension of the derivation in (Lanckriet et al., 2002) and its proof can be found in Appendix A.

Proposition 9 (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

$$\begin{array}{lll} \underset{\xi\in\mathbb{R}^n}{\text{minimize}} \underset{x\in\mathbb{R}^m}{\text{maximize}} & -\frac{1}{2}x^\top H(\xi)x - c(\xi)^\top x + d(\xi) \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & G(\xi) \succeq 0 \end{array} \tag{17}$$

can be rewritten as

$$\begin{array}{ll} \underset{t,\xi,\gamma}{\text{minimize}} & \frac{1}{2}t + a^\top\gamma + d(\xi) \\ \textit{subject to} & \begin{bmatrix} diag(\gamma) & 0 & 0 & 0 \\ 0 & G(\xi) & 0 & 0 \\ 0 & 0 & H(\xi) & (A^\top\gamma - c(\xi)) \\ 0 & 0 & (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{array} \tag{18}$$

in the sense that the ξ which solves (18) also solves (17).

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 10 *Let H, c, A and a be as in Proposition 9, and $\Sigma \succeq 0$, then the solution ξ^* to the optimization problem*

$$\begin{array}{lll} \underset{\xi}{\text{minimize}} \underset{x}{\text{maximize}} & -\frac{1}{2}x^\top H(\xi)x - c(\xi)^\top x + \frac{1}{2}\xi^\top\Sigma\xi \\ \textit{subject to} & H(\xi) \succeq 0 \\ & Ax + a \geqslant 0 \\ & \xi \geqslant 0 \end{array} \tag{19}$$

can be found by solving the a semidefinite programming problem

$$\begin{array}{ll} \underset{t,t',\xi,\gamma}{\text{minimize}} & \frac{1}{2}t + \frac{1}{2}t' + a^\top\gamma \\ \textit{subject to} & \gamma \geqslant 0 \\ & \xi \geqslant 0 \\ & \|\Sigma^{\frac{1}{2}}\xi\| \leqslant t' \\ & \begin{bmatrix} H(\xi) & (A^\top\gamma - c(\xi)) \\ (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \end{array} \tag{20}$$

Proof By applying proposition 9, and introducing an auxiliary variable t' which upper bounds the quadratic term of ξ, the claim is proved. ∎

Comparing the objective function in (19) with (16), 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.

5.2 Examples of Hyperkernel Optimization Problems

From the general framework above, 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 (15). The derivation is basically by application of Corollary 10. Appendix B details the derivation of Example 10. Derivations of the other examples follow the same reasoning, and are omitted.

In this subsection, we define the following notation. For $p, q, r \in \mathbb{R}^n, n \in \mathbb{N}$ let $r = p \odot q$ be defined as element by element multiplication, $r_i = p_i \times q_i$ (the $.*$ operation in Matlab). The pseudo-inverse (or Moore-Penrose inverse) of a matrix K is denoted $K^\dagger$. Define the hyperkernel Gram matrix $\underline{K}$ by $\underline{K}_{ijpq} = \underline{k}((x_i, x_j), (x_p, x_q))$, 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 and $\mathbf{1}$ a vector of ones.

The number of training examples is assumed to be m, i.e. $X_{\text{train}} = \{x_1, \ldots, x_m\}$ 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.

Example 10 (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, which allows errors on the training set. The parameter C is given by the user, and the resulting SDP is*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2}t_1 + \frac{C}{m}\xi^\top \mathbf{1} + \frac{C\lambda_Q}{2}t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2 \\ & \begin{bmatrix} G(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0 \end{array} \tag{21}$$

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

The classification function is given by $f = sign(KG(\beta)^\dagger(y \odot z) - \gamma)$.

Example 11 (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_{emp}, the corresponding SDP is given by*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2}t_1 - \chi\nu + \xi^\top \frac{1}{m} + \frac{\lambda_Q}{2}t_2 \\ \textit{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}}\beta\| \leqslant t_2 \\ & \begin{bmatrix} G(\beta) & z \\ z^\top & t_1 \end{bmatrix} \succeq 0 \end{array} \tag{22}$$

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

The classification function is given by $f = sign(KG(\beta)^\dagger (y \odot z) - \gamma)$.

Example 12 (Quadratic SVM (Lagrangian SVM)) *Instead of using an* L_1 *loss class, Mangasarian and Musicant (2001) uses an* L_2 *loss class, and regularized the weight vector as well as the bias term. The resulting dual SVM problem has fewer constraints, as is evidenced by the smaller number of Lagrange multipliers needed in the SDP below.*

$$\begin{array}{ll} \underset{\beta,\eta}{\text{minimize}} & \frac{1}{2} t_1 + \frac{\lambda_Q}{2} t_2 \\ \textit{subject to} & \eta \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}} \beta\| \leqslant t_2 \\ & \left[\begin{array}{cc} H(\beta) & (\eta + \mathbf{1}) \\ (\eta + \mathbf{1})^\top & t_1 \end{array} \right] \succeq 0 \end{array} \tag{23}$$

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

The classification function is given by $f = sign(KH(\beta)^\dagger ((\eta + \mathbf{1}) \odot y))$.

Example 13 (Single class SVM) *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). The corresponding SDP for this problem, also known as novelty detection, is shown below.*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi}{\text{minimize}} & \frac{1}{2} t_1 + \xi^\top \frac{1}{\nu m} - \gamma + \frac{\lambda_Q}{2\nu} t_2 \\ \textit{subject to} & \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}} \beta\| \leqslant t_2 \\ & \left[\begin{array}{cc} K & z \\ z^\top & t_1 \end{array} \right] \succeq 0 \end{array} \tag{24}$$

where $z = \gamma \mathbf{1} + \eta - \xi$, *and* $\nu \in [0, 1]$ *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 $f = K\alpha - \gamma \mathbf{1}$, *which reduces to* $f = \eta - \xi$, *by substituting* $\alpha = K^\dagger (\gamma \mathbf{1} + \eta - \xi)$ *and* $K = reshape(\underline{k}\beta)$.

Example 14 (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 11, the user defined parameter* ν *controls the fraction of errors and support vectors.*

$$\begin{array}{ll} \underset{\beta,\gamma,\eta,\xi,\chi}{\text{minimize}} & \frac{1}{2} t_1 + \frac{\chi \nu}{\lambda} + \xi^\top \frac{1}{m\lambda} + \frac{\lambda_Q}{2\lambda} t_2 \\ \textit{subject to} & \chi \geqslant 0, \eta \geqslant 0, \xi \geqslant 0, \beta \geqslant 0 \\ & \|\underline{K}^{\frac{1}{2}} \beta\| \leqslant t_2 \\ & \left[\begin{array}{cc} F(\beta) & z \\ z^\top & t_1 \end{array} \right] \succeq 0 \end{array} \tag{25}$$

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 regression function is given by $f = \left[\begin{array}{cc} -K & K \end{array} \right] F(\beta)^\dagger z - \gamma$.

Example 15 (Kernel Target Alignment) *For the Alignment approach (Cristianini et al., 2002), $Q_{\text{emp}} = y^\top K y$, we directly minimize the regularized quality functional, obtaining the following optimization problem,*

$$\begin{array}{ll} \underset{k \in \underline{\mathcal{H}}}{\text{minimize}} & -\frac{1}{2} y^\top K y + \frac{\lambda_Q}{2} \beta^\top \underline{K} \beta \\ \textit{subject to} & \beta \geqslant 0 \end{array}$$

6. Experiments

In the following experiments, we use data from the UCI repository. Where the data attributes are numerical, we *do 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 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 (HARD) defined by (12) for the hyperkernel optimization problems. The scaling freedom that HARD 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.
- $\nu = 0.3$. 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 6.5 for details).
- λ_h for the Harmonic Hyperkernel was chosen to be 0.6, giving adequate coverage over various kernel widths in (10) (small λ_h focus almost exclusively on wide kernels, 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, $c := \text{std}(Y_{\text{train}})$ so that the hyperkernel coefficients are of the same scale as the output (the constant offset b 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.

6.1 Low Rank Approximation

Although the optimization of (15) 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 (13) 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 (13).

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 (2000) and Zhang (2001). This entails picking from $\{\underline{k}((x_i, x_j), \cdot)|1 \leq i, j \leq m\}$ 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 $\|P\underline{K}P^\top - GG^\top\|_F \leqslant \eta$, where P is the permutation matrix which sorts the eigenvalues of $\underline{K}$ into decreasing order, and η is the level of approximation needed. The norm, $\|\cdot\|_F$ is the Frobenius norm. In the following experiments, the hyperkernel matrix was approximated to $\eta = 10^{-6}$ using the incomplete Cholesky factorization method (Bach and Jordan, 2002).

6.2 Classification Experiments

Several binary classification datasets[2] from the UCI repository were used for the experiments. A set of synthetic data sampled from two Gaussians was created to illustrate the scaling freedom between dimensions. The first dimension has standard deviation of 1000 whereas the second dimension has standard deviation of 1 (a sample result is shown in Figure 1). The results of the experiments are shown in Table 3. The second, third and fourth columns show the results of the hyperkernel optimizations of C-SVM (Example 10), ν-SVM (Example 11) and Lagrangian SVM (Example 12) 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 the training data.

In an attempt to lower the computational time required for the experiments, we also performed low rank approximation of the kernel matrix, which effectively is a selection of a subset of the training data. Table 4 shows the results obtained when we approximate the *kernel matrix* using a tolerance of 10^{-6}. The number of data points selected was forced to be between 80 and 300, to control the size of the kernel matrix. The rightmost two columns are repeated from Table 3. The results from the approximate problem were all within one

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

Data	C-SVM	ν-SVM	Lag-SVM	Best other	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

standard deviation of the method using all the data points. This shows the potential of using a subset of the training data to obtain an equally good classification result. The second column shows the average value of the constant η used in the approximation.

Data	Approx. η	C-SVM	ν-SVM	Lag-SVM	other	Tuned SVM (C)
syndata	0	2.9±2.4	1.9±1.9	2.4±2.1	NA	5.9±5.4 (10^8)
pima	4×10^{-7}	23.8±2.0	27.2±2.3	24.1±1.9	23.5	24.1±2.1 (10^4)
ionosph	5×10^{-7}	6.6±2.0	6.8±1.8	6.4±1.9	5.8	6.1±1.8 (10^3)
wdbc	3×10^{-4}	3.3±1.2	3.8±1.2	3.0±1.1	3.2	5.2±1.4 (10^6)
heart	2×10^{-7}	19.5±3.3	19.4±2.5	20.1±2.8	16.0	23.2±3.7 (10^4)
thyroid	1×10^{-9}	6.0±3.1	7.2±3.6	5.5±2.6	4.4	5.2±2.2 (10^5)
sonar	3×10^{-15}	14.8±3.7	15.6±3.8	14.8±3.5	15.4	15.3±4.1 (10^3)
credit	1×10^{-4}	14.8±1.8	13.8±1.6	14.8±1.8	22.8	15.3±2.0 (10^8)
glass	3×10^{-7}	5.9±2.4	8.5±2.6	5.8±2.2	NA	7.2±2.7 (10^3)

Table 4: Approximate kernel classification: Test error and standard deviation in percent

Ong et al. (2002) reported results which were based on an optimization problem where we iteratively alternated between optimizing the kernel coefficients and hyperkernel coefficients. This could potentially result in local minima. In that setting, the regularized quality functional performed poorly on the Ionosphere dataset. This is not the case here, where we optimize using a SDP. Note that the results here for the tuned C-SVM (rightmost column) for the synthetic data and the credit data have improved over our earlier results (see (Ong and Smola, 2003)). This was because we only searched up till $C = 10^6$ in our earlier work, and we searched further for our current results. This demonstrates another advantage of the hyperkernel optimization over parameter selection using cross validation, we can only search a finite number of values (usually only a small number of these) for each parameter.

6.3 Effect of λ_Q and λ_h on Classification Error

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 10). We performed two sets of experiments. In the first, we set

$\lambda_Q = 1$ and varied $\lambda_h = \{0.1, 0.2, \ldots, 0.9, 0.92, 0.94, 0.96, 0.98\}$. In the second experiment, we set $\lambda_h = 0.6$ and varied $\lambda_Q = \{10^{-4}, 10^{-3}, \ldots, 10^5\}$. The results shown in Figures 4 and 3 are the average error over 10 random 60%/40% splits.

From Figure 3, we observe[3] 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 and 4). This demonstrates that our method is 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, λ_h (Figure 4). 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.

6.4 Regression Experiments

To demonstrate that we can solve problems other than binary classification using the same framework, we performed regression and novelty detection. The results of regression are shown in Table 5. We have *the same parameter settings* as in the previous section. The second column shows the results from the hyperkernel optimization of the ν-regression (Example (14)). 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 sample of the training data.

Data	ν-SVR	Best other	Tuned ε-SVM
auto-mpg	7.76±1.05	7.11	9.61±1.18
boston	12.02±2.25	9.60	15.04±3.31
auto imports($\times 10^6$)	4.42±1.04	0.25	5.76±1.28
cpu($\times 10^3$)	4.25±2.89	3.16	9.79±7.29
servo	0.79±0.16	0.25	0.57±0.14

Table 5: Hyperkernel regression: Mean Squared Error

Meyer et al. (2003) use a 90%/10% split of the data for their experiments. Analyzing their results further indicate that, for the auto imports, cpu and servo datasets, their results were obtained using projection pursuit regression, which is spline based. This indicates that we may have been searching in the incorrect class of kernels (for these specific datasets) when considering Gaussian kernels.

3. The training error for the sonar dataset is zero.

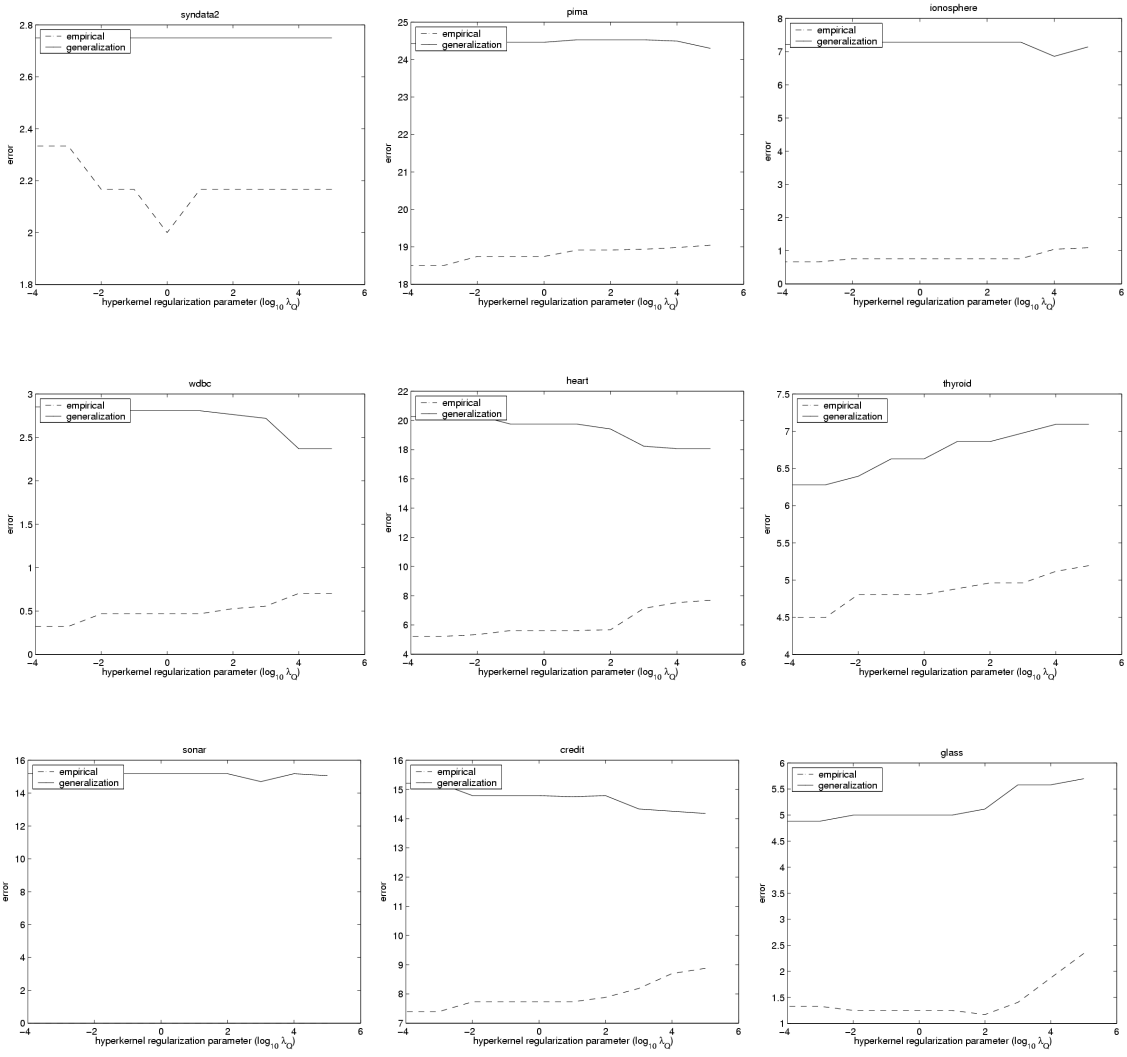


Figure 3: Effect of varying λ_Q on classification error

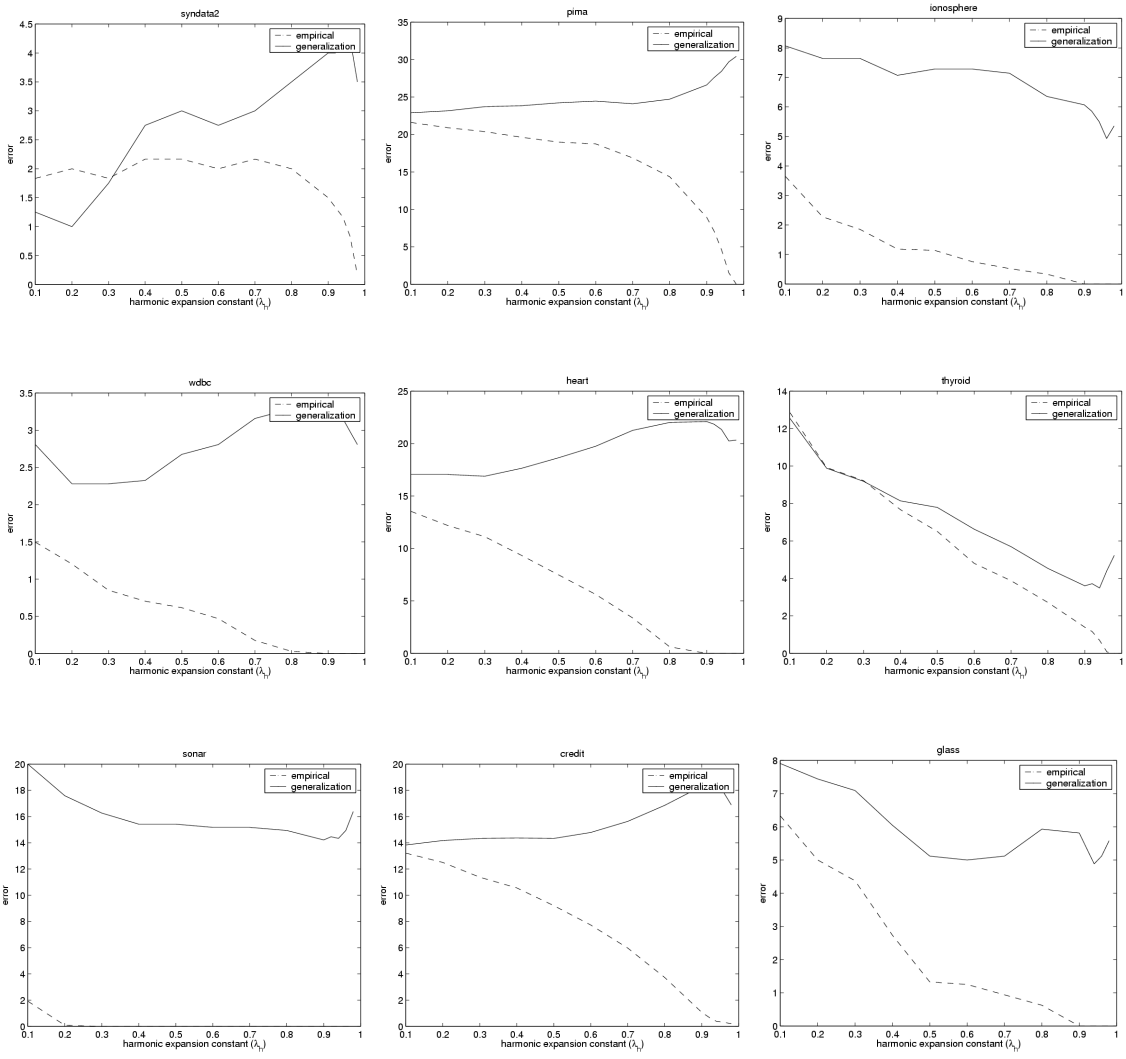


Figure 4: Effect of varying λ_h on classification error

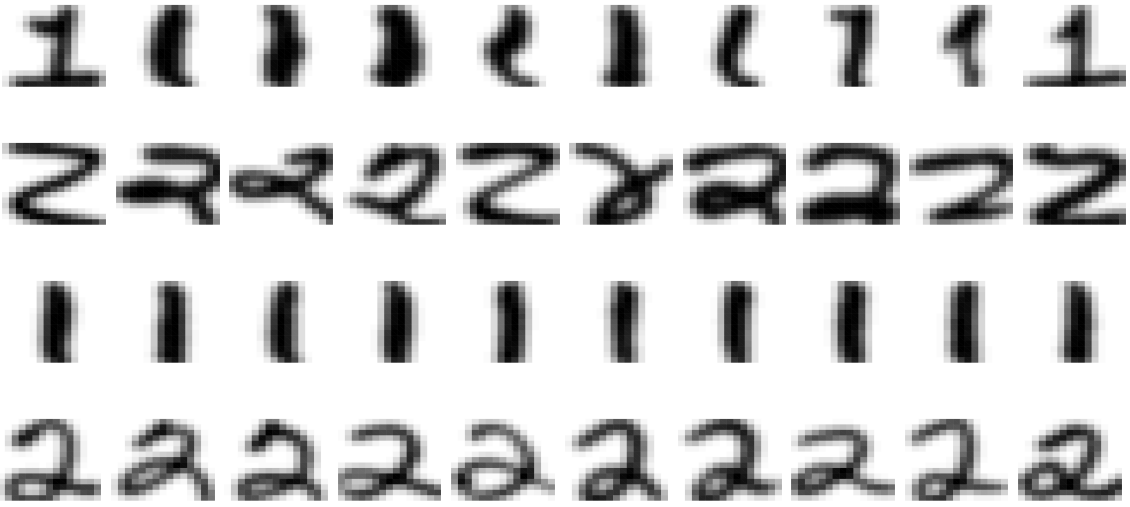


Figure 5: Top rows: Images of digits '1' and '2', considered novel by algorithm; Bottom: typical images of digits '1' and '2'.

6.5 Novelty Detection

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

7. 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 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. This SDP, which learns the best kernel given the data, has a Bayesian interpretation of a hierarchical GP. Since we can optimize over the whole class of kernel functions, we can define more general kernels which may have many free parameters, without overfitting. The experimental results on classification demonstrate that it is possible to achieve the state of the art. Furthermore, the same framework and parameter settings work for various datasets as well as regression and novelty detection.

It is important to stress that our approach has made support vector estimation more automated. Parameter adjustment is less critical compared to the case when the kernel is fixed. Future work will focus on deriving improved statistical guarantees for estimates derived via hyperkernels which match the good empirical performance. Extensions of the method to learning functions more general than kernels, such as kernels without the positivity constraint, metrics, and operators are also under investigation.

Acknowledgements This work was supported by a grant of the Australian Research Council. The authors would like to thank Laurent El Ghaoui, Michael Jordan, John Lloyd, Daniela Pucci de Farias and Grace Wahba for their helpful comments and suggestions. The authors also thank Alexandros Karatzoglou for his help with SVLAB.

Appendix A. Proof of Proposition 9

We will make use of a theorem due to (Albert, 1969) which is a generalization of the Schur complement lemma for positive semidefinite matrices.

Theorem 11 (Generalized Schur Complement) *Let* $X = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}$, *where* A *and* C *are symmetric. Then*

$$X \succeq 0 \text{ if and only if } A \succeq 0, AA^\dagger B = B \text{ and } C - B^\top A^\dagger B \succeq 0 \tag{26}$$

where $A^\dagger$ *is the Moore-Penrose inverse of* A.

We prove the proposition that the solution of the quadratic minimax problem (17) is obtained by minimizing the SDP (18).

Proof [of Proposition 9] Rewrite the terms of the objective function in (17) dependent on x in terms of their Wolfe dual. The corresponding Lagrange function is

$$L(x, \xi, \gamma) = -\frac{1}{2} x^\top H(\xi) x - c(\xi)^\top x + \gamma^\top (Ax + a), \tag{27}$$

where $\gamma \in \mathbb{R}^M$ is a vector of Lagrange multipliers with $\gamma \geqslant 0$. By differentiating $L(x, \xi, \gamma)$ with respect to x and setting the result to zero, one obtains that (27) is maximized with respect to x for $x = H(\xi)^\dagger (A^\top \gamma - c(\xi))$ and subsequently we obtain the dual

$$D(\xi, \gamma) = \frac{1}{2} (A^\top \gamma - c(\xi))^\top H(\xi)^\dagger (A^\top \gamma - c(\xi)) + \gamma^\top a. \tag{28}$$

Note that $H(\xi)^\dagger H(\xi) H(\xi)^\dagger = H(\xi)^\dagger$. For equality constraints in (17), such as $Bx + b = 0$, we get correspondingly free dual variables.

The dual optimization problem is given by inserting (28) into (17)

$$\begin{array}{ll} \underset{\xi,\gamma}{\text{minimize}} & \frac{1}{2} (A^\top \gamma - c(\xi))^\top H(\xi)^\dagger (A^\top \gamma - c(\xi)) + \gamma^\top a + d(\xi) \\ \text{subject to} & H(\xi) \succeq 0, G(\xi) \succeq 0, \gamma \geqslant 0. \end{array} \tag{29}$$

Introducing an auxiliary variable, t, which serves as an upper bound on the quadratic objective term gives an objective function linear in t and γ. Then (29) can be written as

$$\begin{array}{ll} \underset{\xi,\gamma}{\text{minimize}} & \frac{1}{2}t + \gamma^\top a + d(\xi) \\ \text{subject to} & t \succeq (A^\top\gamma - c(\xi))^\top H(\xi)^\dagger (A^\top\gamma - c(\xi)), \\ & H(\xi) \succeq 0, G(\xi) \succeq 0, \gamma \geqslant 0. \end{array} \tag{30}$$

From the properties of the Moore-Penrose inverse, we get $H(\xi)H(\xi)^\dagger(A^\top\gamma - c(\xi)) = (A^\top\gamma - c(\xi))$. Since $H(\xi) \succeq 0$, by Theorem 11, the quadratic constraint in (30) is equivalent to

$$\begin{bmatrix} H(\xi) & (A^\top\gamma - c(\xi)) \\ (A^\top\gamma - c(\xi))^\top & t \end{bmatrix} \succeq 0 \tag{31}$$

Stacking all the constraints in (30) as one linear matrix inequality proves the claim. ■

Appendix B. Derivation of SDP for C-SVM

We derive the corresponding SDP for the case when Q_{emp} is a C-SVM (Example 10).

We begin our derivation from the regularized quality functional (15). Dividing throughout by λ and setting the cost function to the L_1 soft margin loss, that is $\ell(x_i, y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$ we get the following equation.

$$\begin{array}{ll} \min_{k\in\underline{\mathcal{H}}} \min_{f\in\mathcal{H}_k} & \frac{1}{m\lambda}\sum_{i=1}^m \xi_i + \frac{1}{2}\|f\|^2_{\mathcal{H}_k} + \frac{\lambda_Q}{2\lambda}\|k\|^2_{\underline{\mathcal{H}}} \\ \text{subject to} & y_i f(x_i) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \end{array} \tag{32}$$

Recall the form of the C-SVM,

$$\begin{array}{ll} \min\limits_{w,\xi} & \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^m \xi_i \\ \text{subject to} & y_i(\langle x_i, w\rangle + b) \geqslant 1 - \xi_i \\ & \xi_i \geqslant 0 \text{ for all } i = 1, \ldots, m \end{array}$$

and its dual,

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

By considering the optimization problem dependent on f in (32), we can use the derivation of the dual problem of the standard C-SVM. Observe that $C = \lambda^{-1}$, and we can rewrite

$\|k\|^2_{\underline{\mathcal{H}}} = \beta^\top \underline{K}\beta$ due to the representer theorem. Substituting the dual C-SVM problem into (32), we get the following matrix equation,

$$\begin{array}{ll} \min\limits_{\beta}\max\limits_{\alpha} & \mathbf{1}^\top\alpha - \frac{1}{2}\alpha^\top G(\beta)\alpha + \frac{C\lambda_Q}{2}\beta^\top \underline{K}\beta \\ \text{subject to} & \alpha^\top y = 0 \\ & 0 \leqslant \alpha_i \leqslant \frac{C}{m} \text{ for all } i = 1, \ldots, m \\ & \beta_i \geqslant 0 \end{array} \tag{33}$$

This is of the quadratic form of Corollary 10 where $x = \alpha$, $\xi = \beta$, $H(\xi) = G(\beta)$, $c(\xi) = \mathbf{1}$, $\Sigma = C\lambda_Q\underline{K}$, $A = \left[\begin{array}{cccc} y & -y & \mathbf{I} & -\mathbf{I} \end{array}\right]^\top$ and $a = \frac{\mathbf{C}}{m}$. Applying Corollary 10, we obtain the SDP in Example 10.

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.

P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 43–54, Cambridge, MA, 1999. MIT Press.

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

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

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*, 2002. In press.

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.

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 representation. Technical report, IBM Watson Research Center, New York, 2000.

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*. Morgan Kaufmann, 2002.

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. http://www.jmlr.org.

D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 2003. Forthcoming.

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*, 2003. submitted.

C. S. Ong, A. J. Smola, and R. C. Williamson. Hyperkernels. In *Neural Information Processing Systems*, volume 15. 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, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 2001.

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

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

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

J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999. Special issue on Interior Point Methods (CD supplement with software).

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.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

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

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.