Learning the Kernel Matrix with Semi-Definite Programming

Gert Lanckriet GERT@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Science, U.C. Berkeley

Nello Cristianini NELLO@SUPPORT-VECTOR.NET
Peter Bartlett PETER.BARTLETT@ANU.EDU.AU
BIOwulf Technologies / Division of Computer Science and Department of Statistics, U.C. Berkeley

Laurent El Ghaoui ELGHAOUI@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Science, U.C. Berkeley

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Division of Computer Science and Department of Statistics, U.C. Berkeley

Abstract

Kernel-based learning algorithms work by embedding the data into a Euclidean space, and then searching for linear relations among the embedded data points. The embedding is performed implicitly, by specifying the inner products between each pair of points in the embedding space. This information is contained in the so-called kernel matrix, a symmetric and positive definite matrix that encodes the relative positions of all points. Specifying this matrix amounts to specifying the geometry of the embedding space and inducing a notion of similarity in the input space—classical model selection problems in machine learning. In this paper we show how the kernel matrix can be learned from data via Semi-Definite Programming techniques. When applied to a kernel matrix associated with both training and test data this gives a powerful transductive algorithm—using the labelled part of the data one can learn an "optimal" embedding also for the unlabelled part. The induced similarity between test points is learned by using training points and their labels. Importantly, these learning problems are convex, so we obtain a method for learning both the model class and the function without local minima. Finally, the novel approach presented in the paper is supported by positive empirical results.

1. Introduction

Finding a representation of the data that makes it easier to detect certain given structure is an important task that usually precedes the use of learning algorithms.

In kernel-based learning methods, such a representation is implicit in the choice of the kernel—a function that specifies the inner product between the images of two given data points in a certain space, dictating their relative positions in it. The choice of the space itself is also a component of the choice of a representation.

It is important to observe that we do not necessarily need to choose a kernel *function*—the embedding of a finite set of points is entirely specified by writing a finite-dimensional kernel *matrix*, also known as a "Gram matrix." It is possible to prove that any symmetric positive definite matrix is a valid Gram matrix, in the sense that it specifies the values of some inner product. This suggests viewing the model selection problem in terms of Gram matrices rather than kernel functions.

In this paper we address the problem of transduction—completing the labelling of a partially labelled dataset. In other words, we are only required to make predictions on a finite set of points known a priori. Instead of learning a function, we just need to learn a set of labels. Equivalently, our data live in a finite set, completely specified at the start.

We will address this problem by learning a kernel matrix corresponding to the entire dataset, a matrix that optimizes a certain cost function depending on the available labels. In other words, we use the available labels to learn a good embedding, and we apply it to both the labelled and the unlabelled data. The resulting kernel matrix can then be used in combination with a number of existing learning algorithms that use kernels, for example support vector machines.

All this can be done in full generality by using techniques from semi-definite programming, a branch of convex optimization that deals with optimizing convex functions over the convex cone of positive semi-definite matrices, or convex subsets thereof. The cost functions we use are motivated by error bounds, and are convex.

The paper is organized as follows. In Section 2, the essentials of kernel-based learning are presented. Section 3 introduces semi-definite programming and shows how this technique can be used to optimize kernel matrices, to obtain a novel algorithm. These pieces are put together in Section 4, where we explain how to learn the kernel matrix. Section 5 presents novel error bounds that motivate our cost functions, and positive empirical results are reported in Section 6.

2. Kernel Methods

Kernel-based learning algorithms (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002) work by embedding the data into a Hilbert space, and searching for linear relations in such a space. The embedding is performed implicitly, by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. This approach has several advantages, the most important deriving from the fact that often the inner product in the embedding space can be computed much more easily than the coordinates of the points themselves.

Given an input set $\mathcal{X}$, and an embedding space $\mathcal{F}$, we consider a map $\Phi : \mathcal{X} \to \mathcal{F}$. Given two points $x_i \in \mathcal{X}$ and $x_j \in \mathcal{X}$, the function that returns the inner product between their images in the space $\mathcal{F}$ is known as the *kernel function.*

Definition A *kernel* is a function k, such that $k(x, z) = \langle \Phi(x), \Phi(z) \rangle$ for all $x, z \in \mathcal{X}$, where Φ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{F}$.

We also consider the matrix $K_{ij} = k(x_i, x_j)$:

$$K = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n ,$$

which is known as the "kernel matrix" or "Gram matrix." It is a symmetric positive definite matrix, and since it specifies the inner products between all pairs of points $\{x_i\}_{i=1}^n$, it completely determines the relative positions between those points in the embedding space.

Since in this paper we will consider a *finite* input set $\mathcal{X}$, we can characterize kernel functions and matrices in the following simple way.

Proposition Every positive definite and symmetric matrix is a kernel matrix, that is, an inner product matrix in some embedding space. Conversely, every kernel matrix is symmetric and positive definite.

Notice that, if we have a kernel matrix, we do not need to know the kernel function, nor the implicitly defined map Φ, nor the coordinates of the points $\Phi(x_i)$. We do not even need $\mathcal{X}$ to be a vector space; in fact in this paper it will be a generic finite set. We are guaranteed that the data are implicitly mapped to some Hilbert space by simply checking that the kernel matrix satisfies the conditions above.

The solutions sought by kernel-based algorithms such as the support vector machine (SVM) are linear functions in the feature space:

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}),$$

for some weight vector $\mathbf{w}$. The kernel can be exploited whenever the weight vector can be expressed as a linear combination of the training points, $\mathbf{w} = \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i)$, implying that we can express f as follows:

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

An important issue in applications is that of choosing a kernel k for a given learning task; intuitively, we wish to choose a kernel that induces the "right" metric in the space.

For the special case of two-class classification, several measures of separation between two sets of data have been developed. For example, we can define the *alignment* between a kernel and a set of labels, or the *margin* of a separation (the distance between the convex hulls of the two classes), and its noise-tolerant version the "*soft margin.*"

We will first define the alignment between two kernels, then this will be extended to the alignment between kernel and labels, by constructing a "target kernel" $k(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j$ with $y_i \in \mathcal{Y} = \{-1, +1\}$.

Given an (unlabelled) sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we use the following (Frobenius) inner product between Gram matrices, $\langle K_1, K_2 \rangle_F = \sum_{i,j=1}^n k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j)$.

Alignment The (empirical) alignment of a kernel k_1 with a kernel k_2 with respect to the sample S is the quantity

$$\hat{A}(S, k_1, k_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}},$$

where K_i is the kernel matrix for the sample S using kernel k_i.

This can also be viewed as the cosine of the angle between two bi-dimensional vectors K_1 and K_2, representing the Gram matrices. If we consider $K_2 = yy^T$, where y is the vector of $\{\pm 1\}$ labels for the sample, then

$$\hat{A}(S, K, yy^T) = \frac{\langle K, yy^T \rangle_F}{\sqrt{\langle K, K \rangle_F \langle yy^T, yy^T \rangle_F}} = \frac{\langle K, yy^T \rangle_F}{m\sqrt{\langle K, K \rangle_F}},$$

since $\langle yy^T, yy^T \rangle_F = m^2$.

Secondly, we will define the margin of separation, which will then be expressed using duality.

Margin Given a linearly separable labelled sample $S_l = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the hyperplane $(\mathbf{w}, b)$ that solves the optimization problem

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \langle \mathbf{w}, \mathbf{w} \rangle \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1, i = 1, \ldots, n \end{aligned} \tag{1}$$

realizes the maximal margin classifier with geometric margin $\gamma = 1/\|\mathbf{w}_*\|_2$ (Cristianini & Shawe-Taylor, 2000), where $\mathbf{w}_*$ is that $\mathbf{w}$ that optimizes (1). This margin is also called hard margin (because of the linear separability).

Geometrically, γ corresponds to the distance between the convex hulls (the smallest convex sets that contain the data in each class) of the two classes (Bennett & Bredensteiner, 2000). By transforming (1) into its corresponding dual problem (Cristianini & Shawe-Taylor, 2000), we can express the squared inverse margin $w(K) = 1/\gamma^2$ corresponding to a kernel matrix K as follows:

$$\begin{aligned} w(K) &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle \\ &= \max_{\alpha} \; 2\alpha^T e - \alpha^T G(K) \alpha \; : \; \alpha \geq 0, \;\; \alpha^T y = 0, \end{aligned} \tag{2}$$

where e is the n-vector of ones, $\alpha \in \mathbb{R}^n$, $G(K)$ is defined by $G_{ij}(K) = [K]_{ij} y_i y_j = k(\mathbf{x}_i, \mathbf{x}_j) y_i y_j$ and $\alpha \geq 0 \Leftrightarrow \alpha_i \geq 0, \; i = 1, \ldots, n$.

For a non-linearly separable labelled sample S_l, we can define the noise-tolerant soft margin (Schölkopf & Smola, 2002). Geometrically, this corresponds to the distance between reduced convex hulls of the two classes (Bennett & Bredensteiner, 2000).

Both the alignment (Cristianini et al., 2001) and the soft margin (Schölkopf & Smola, 2002) are concentrated quantities. This means that we can reliably estimate their expected value from a finite sample, or estimate their value on the test set by observing them on the training set.

These quantities can now be used to obtain bounds on the generalization of given classifiers (under the assumption that the data have been drawn i.i.d.). As we will see in Section 5, γ can be used to bound the performance of support vector machines: for a thresholded version of $f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$, with $\|\mathbf{w}\|^2 = 1/\gamma^2$, if $y_i f(\mathbf{x}_i) \geq 1$ for $i = 1, \ldots, n$, then the proportion of errors on the test data is, with probability $1 - \delta$, bounded by

$$\frac{1}{\sqrt{n}} \left(4 + \sqrt{2 \log(1/\delta)} + \sqrt{\frac{BC}{n\gamma^2}} \right), \tag{3}$$

where $\mathrm{trace}(K) \leq B$, and the kernel matrix K is of the form $K = \sum_{i=1}^m \mu_i K_i$ for a fixed set $\{K_1, \ldots, K_m\}$, and C depends on these m kernel matrices. As we will see in Section 4, this is exactly what we need for learning the kernel matrix.

3. Semi-Definite Programming

Semi-definite programming (Vandenberghe & Boyd, 1996) deals with the optimization of convex functions over the convex cone[1] $\mathcal{P} = \left\{ X \in \mathbb{R}^{p \times p} | X = X^T, X \succeq 0 \right\}$ of symmetric positive semi-definite matrices, or subsets of this cone. Given the Proposition in Section 2, $\mathcal{P}$ can be viewed as a search space for possible kernel matrices. This consideration leads to the key problem addressed in this paper—we wish to specify a convex cost function that will enable us to learn the optimal kernel matrix from $\mathcal{P}$ using semi-definite programming. Because of the convexity, this approach allows us to avoid problems with local minima.

Definition The general form of a Semi-Definite Program (SDP) is

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subject to} \quad & F(x) = F_0 + x_1 F_1 + \ldots + x_n F_n \succeq 0 \\ & Ax = b, \end{aligned} \tag{4}$$

where $x \in \mathbb{R}^p$ and $F_i = F_i^T \in \mathbb{R}^{p \times p}$. $F(x) \succeq 0$ (called a linear matrix inequality, LMI) restricts $F(x)$ to be contained in the positive semi-definite cone $\mathcal{P}$. LMI's

[1] $S \subseteq \mathbb{R}^d$ is a convex cone if $x, y \in S, \lambda, \mu \geq 0 \Rightarrow \lambda x + \mu y \in S$.

are annotated with $\succeq$ instead of $\geq$ to distinguish them from normal linear inequality statements. Notice that the objective is linear in the unknowns x, and that both the LMI and the equality constraint are linear in x (e.g., with x being the entries of F for an appropriate choice of the F_i's).

We will now show how optimizing the alignment as well as optimizing the margin can be cast as a semi-definite programming problem. For simplicity, we assume in this section that all labels are known. Our goal is to find the optimal embedding (i.e., the optimal kernel matrix K) such that a measure of separation between those two sets of data is maximized.

Alignment We want to find the kernel matrix K which is maximally aligned with the set of labels y:

$$\begin{aligned} \max_K \quad & \hat{A}(S, K, yy^T) \\ \text{subject to} \quad & K \succeq 0. \end{aligned}$$

This is equivalent to the following optimization problem:

$$\begin{aligned} \max_K \quad & \langle K, yy^T \rangle \\ \text{subject to} \quad & \langle K, K \rangle = 1 \\ & K \succeq 0. \end{aligned}$$

Given the linearity of the objective in K, we can replace the first constraint by $\langle K, K \rangle \leq 1$ because of the maximization. Using the Schur complement lemma[2] for this constraint, one can write the optimization problem as:

$$\begin{aligned} \max_{A,K} \quad & \langle K, yy^T \rangle \\ \text{subject to} \quad & \operatorname{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^T \\ K & I \end{pmatrix} \succeq 0 \\ & K \succeq 0. \end{aligned}$$

Or, by stacking the constraints in one LMI:

$$\begin{aligned} \max_{A,K} \quad & \langle K, yy^T \rangle \\ \text{subject to} \quad & \begin{pmatrix} A & K^T & O & O \\ K & I & O & O \\ O & O & 1 - \operatorname{trace}(A) & O \\ O & O & O & K \end{pmatrix} \succeq 0. \end{aligned} \tag{5}$$

[2]Consider the partitioned symmetric matrix

$$X = X^T = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}.$$

$S = C - B^T A^{-1} B$ is the Schur complement of A in X (provided $\det(A) \neq 0$). The Schur complement lemma then says: if $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.

This results in the standard form (4) (consider the entries of the matrices A and K as the unknowns x_i): when using an alignment criterion, the kernel matrix K can indeed by optimized in an SDP setting.

Hard margin Inspired by (3), let us try to find the kernel matrix K for which the corresponding embedding shows maximal margin between the convex hull of both sets of data, keeping the trace of K constant:

$$\min_{K \succeq 0} w(K) \qquad \text{s.t. } \operatorname{trace}(K) = c, \tag{6}$$

with $w(K)$ the squared inverse margin defined in (2) and c a constant. Assume that $K \succ 0$, hence $G \succ 0$ (the following can be extended to the general case). We note that $w(K)$ is convex in K (it is the pointwise maximum of affine functions of K). Given the convex constraints in (6), the optimization problem is thus certainly convex in K. In order to express it as an SDP, we write this as:

$$\begin{aligned} \min_{K \succeq 0, t} t \ : \quad & t \geq \max_\alpha \ 2\alpha^T e - \alpha^T G(K) \alpha, \\ & \alpha \geq 0, \ \ \alpha^T y = 0, \ \ \operatorname{trace}(K) = c. \end{aligned} \tag{7}$$

We will now express $t \geq \max_\alpha \ 2\alpha^T e - \alpha^T G(K)\alpha$ as an LMI; we express the constraint using the dual minimization problem. This will allow us to drop the minimization and use the Schur complement lemma to obtain an LMI.

Define the Lagrangian of the maximization problem (2) by

$$\mathcal{L}(\alpha, \nu, \lambda) = 2\alpha^T e - \alpha^T G(K)\alpha + 2\nu^T \alpha + 2\lambda y^T \alpha.$$

By duality, we have

$$w(K) = \max_\alpha \min_{\nu \geq 0, \lambda} \ \mathcal{L}(\alpha, \nu, \lambda) = \min_{\nu \geq 0, \lambda} \ \max_\alpha \ \mathcal{L}(\alpha, \nu, \lambda).$$

Since $G \succ 0$, at the optimum, we have

$$\alpha = G(K)^{-1}(e + \nu + \lambda y),$$

and can form the dual problem

$$w(K) = \min_{\nu, \lambda} (e+\nu+\lambda y)^T G(K)^{-1}(e+\nu+\lambda y) \ : \ \nu \geq 0.$$

We obtain that for any $t > 0$, the constraint $w(K) \leq t$ is true if and only if there exist $\nu \geq 0$ and λ such that

$$(e + \nu + \lambda y)^T G(K)^{-1}(e + \nu + \lambda y) \leq t,$$

or, equivalently (using the Schur complement lemma), such that

$$\begin{pmatrix} G(K) & e + \nu + \lambda y \\ (e + \nu + \lambda y)^T & t \end{pmatrix} \succeq 0$$

holds. Stacking all constraints in one single LMI, (7) can be expressed as a standard SDP (4):

$$\min_{K,t,\lambda,\nu} \quad t \tag{8}$$
$$\text{s.t. } \operatorname{trace}(K) = c,$$
$$\begin{pmatrix} K & O & O & O \\ O & G(K) & e+\nu+\lambda y & O \\ O & (e+\nu+\lambda y)^T & t & O \\ O & O & O & \operatorname{diag}(\nu) \end{pmatrix} \succeq 0.$$

Thus when using a margin criterion, the kernel matrix K can be optimized in an SDP setting. This can easily be extended for the soft margin case: similar theoretical results have been obtained but are omitted here because of space limitations.

4. Learning the kernel matrix

By assuming all labels to be known, we have thus far dealt only with optimizing the kernel matrix, rather than learning it—the SDP approach in the previous section optimizes the embedding of labelled points, according to some measure of separation. In the current section, we drop the assumption that all labels are known and address the transductive learning problem. By optimizing the embedding for the labelled part of the data, we hope to learn a good embedding also for the unlabelled part.

Formally, we consider a kernel matrix which has the following structure:

$$K = \begin{pmatrix} K_{tr} & K_{trt} \\ K_{trt}^T & K_t \end{pmatrix}, \tag{9}$$

where $K_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle$, $i, j = 1, \ldots, n_{tr}, n_{tr} + 1, \ldots, n_{tr} + n_t$ with n_{tr} and n_t the number of labelled (training) and unlabelled (test) data points respectively. By optimizing a cost function over the "training-data block" K_{tr}, we want to learn the optimal mixed block K_{trt} and the optimal "test-data block" K_t.

This implies that training and test-data blocks must somehow be entangled: tuning training-data entries in K (to optimize their embedding) should imply that test-data entries are automatically tuned in some optimal way as well ("optimal" meaning that we can expect good generalization). This can be achieved by constraining the search space of possible kernel matrices: we control the capacity of the search space of possible kernel matrices in order to prevent overfitting and achieve good generalization on test data. A possible constraint is given by

$$K = \sum_{i=1}^{m} \mu_i K_i, \tag{10}$$

where the set $\mathcal{K} = \{K_1, \ldots, K_m\}$ is given and the μ_i are to be optimized. $\mathcal{K}$ could be a set of initial "bad guesses" of the kernel matrix (e.g., linear, Gaussian or polynomial kernels with different kernel parameter values). The K_i could also be chosen as the rank-one matrices $K_i = v_i v_i^T$, with v_i the eigenvectors of K_0, an initial kernel matrix.

Replacing K by K_{tr} in (5) and (8), the problem of optimizing over the training set, under the additional constraint (10), leads to the following semi-definite programming formulation for learning the kernel matrix:

Alignment

$$\max_{A,\mu_i} \left\langle \sum_i \mu_i K_{i,tr}, yy^T \right\rangle \tag{11}$$
$$\text{s.t.} \begin{pmatrix} A & \sum_i \mu_i K_{i,tr}^T & O & O \\ \sum_i \mu_i K_{i,tr} & I & O & O \\ O & O & 1-\operatorname{trace}(A) & O \\ O & O & O & \sum_i \mu_i K_i \end{pmatrix} \succeq 0,$$

where $K_{i,tr}$ represents the training-data block of K_i. The alignment is optimized over the labelled data (use of $K_{i,tr}$), while the positive semi-definiteness should hold for the entire kernel matrix K (use of K_i).

Hard margin

$$\min_{\mu_i,t,\lambda,\nu} \quad t \tag{12}$$
$$\text{s.t. } \operatorname{trace}\left(\sum_i \mu_i K_i\right) = c,$$
$$\begin{pmatrix} \sum_i \mu_i K_i & O & O & O \\ O & G(\sum_i \mu_i K_{i,tr}) & e+\nu+\lambda y & O \\ O & (e+\nu+\lambda y)^T & t & O \\ O & O & O & \operatorname{diag}(\nu) \end{pmatrix} \succeq 0.$$

At this point, the SDP approach becomes consistent with the bound in (3), because K is now indeed considered as a linear combination $K = \sum_{i=1}^m \mu_i K_i$ for a fixed set $\{K_1, \ldots, K_m\}$. The margin is optimized over the labelled data (use of $K_{i,tr}$), while the positive semi-definiteness and the trace constraint are imposed for the entire kernel matrix K (use of K_i).

This gives a new transduction method for hard margin SVMs (real transduction with rank one matrices from eigenvectors in (10); otherwise not fully transduction, but this doesn't seem a disadvantage). Vapnik's original straightforward transduction approach scales exponentially in the number of test samples. This offers a more practice transduction method that is polynomial time - special purpose methods exploiting the particular structure of the problem should result in very efficient algorithms. Before giving empirical results, we provide a fuller investigation of the error bound (3).

5. Error Bounds for Transduction

In the problem of transduction, we have access to the unlabelled test data, as well as the labelled training data, and the aim is to optimize accuracy in predicting the test data. We assume that the data are fixed, and that the order is chosen randomly, yielding a random partition into training and test sets. For convenience, we suppose here that the training and test sets have the same size.

Fix a sequence of $2n$ pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{2n}, y_{2n})$ from $\mathcal{X} \times \mathcal{Y}$. Let $\pi : \{1, \ldots, 2n\} \to \{1, \ldots, 2n\}$ be a random permutation, chosen uniformly, and let $(X_i, Y_i) = (\mathbf{x}_{\pi(i)}, y_{\pi(i)})$. The first half of this randomly ordered sequence is the training data, and the second half is the test data. For a function $f : \mathcal{X} \to \Re$, we write the proportion of errors on the test data of a thresholded version of f as

$$\operatorname{er}(f) = \frac{1}{n} |\{n + 1 < i < 2n : Y_i f(X_i) \leq 0\}|.$$

The following theorem shows that the error of a kernel classifier on the test data can be bounded in terms of the average of a certain cost function of the training data margins, as well as properties of the kernel matrix. For $\gamma > 0$, define the margin cost function $\phi_\gamma : \Re \to \Re^+$ as

$$\phi_\gamma(a) = \begin{cases} 1 & \text{if } a \leq 0, \\ 1 - a/\gamma & 0 < a \leq \gamma, \\ 0 & a > \gamma. \end{cases}$$

We consider kernel classifiers obtained by thresholding kernel expansions of the form

$$f(x) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle = \sum_{i=1}^{2n} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{13}$$

where $\mathbf{w} = \sum_{i=1}^{2n} \alpha_i \Phi(\mathbf{x}_i)$ is chosen with bounded norm,

$$\|\mathbf{w}\|^2 = \sum_{i,j=1}^{2n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \alpha' K \alpha \leq 1. \tag{14}$$

Here, K is the $2n \times 2n$ kernel matrix with $K_{ij} = k(X_i, X_j)$. With this constraint, the value of the margin $y f(\mathbf{x})$ is the distance in feature space between $\Phi(\mathbf{x})$ and the decision boundary, such that γ as defined here is fully consistent with γ as defined in (1). Let F_K denote the class of functions of the form (13) satisfying (14).

We are also interested in the class of kernel expansions obtained from certain linear combinations of a fixed set $\{K_1, \ldots, K_m\}$ of kernel matrices. Define F_B as the class of kernel expansions of the form (13) with $k = \sum_{j=1}^{m} \beta_j k_j$, $K_{ij} = k(X_i, X_j)$, $\alpha' K \alpha \leq 1$, $K \succeq 0$, and $\operatorname{trace}(K) \leq B$.

Theorem 1 *Let $\phi : \Re \to \Re^+$ satisfy $\phi \geq \phi_\gamma$. With probability at least $1 - \delta$ over the data (X_i, Y_i) chosen as above, every function $f \in F_K$ has $\operatorname{er}(f)$ no more than*

$$\frac{1}{n} \sum_{i=1}^{n} \phi(Y_i f(X_i)) + \frac{1}{\sqrt{n}} \left(4 + 2\sqrt{\log(1/\delta)} + \sqrt{\frac{\operatorname{trace}(K)}{n\gamma^2}} \right).$$

Moreover, for any set $\{K_1, \ldots, K_m\}$ of kernel matrices, define $C = \mathbf{E} \sup(\sigma' K \sigma)$, where the expectation is over σ chosen uniformly from $\{\pm 1\}^{2n}$ and the supremum is over matrices $K = \sum_j \beta_j K_j$ satisfying $K \succeq 0$ and $\operatorname{trace}(K) = 1$. Then $C \leq n$ and, with probability at least $1 - \delta$, every function $f \in F_B$ has $\operatorname{er}(f)$ no more than

$$\frac{1}{n} \sum_{i=1}^{n} \phi(Y_i f(X_i)) + \frac{1}{\sqrt{n}} \left(4 + 2\sqrt{\log(1/\delta)} + \sqrt{\frac{BC}{n\gamma^2}} \right).$$

Notice that, in both cases, the test error is bounded by a sum of the average over the training data of a margin cost function plus a complexity penalty term that depends on the ratio between the trace of the kernel matrix and the squared margin parameter, γ^2. The kernel matrix here is the full matrix, combining both test and training data. The proof of the theorem is in the appendix.

6. Empirical results

In Cristianini et al. (2001) empirical results are given for optimization of the alignment using a kernel matrix $K = \sum_{i=1}^{N} \mu_i v_i v_i^T$. The results show that optimizing the alignment indeed improves the generalization power of Parzen windows classifiers. It turns out that in this particular case, the SDP in (11) boils down to exactly the quadratic program that is obtained in Cristianini et al., (2001) and thus those results fit within the scope of the current paper.

Here we present results for hard margin support vector machines. The margin is maximized according to (12), using a kernel matrix $K = \sum_{i=1}^{3} \mu_i K_i$, where the K_i's are initial "bad guesses" of the kernel matrix. We use a polynomial kernel function

$k_1(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1^T \mathbf{x}_2)^d$ for K_1, a Gaussian kernel function $k_2(\mathbf{x}_1, \mathbf{x}_2) = \exp(-0.5(\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)/\sigma)$ for K_2 and a linear kernel function $k_3(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$ for K_3. Afterwards, all K_i are normalized. In this setting, no simplifications are possible in (12) and we obtain a true semi-definite program.

Empirical results on standard benchmark datasets are summarized in Table 1. The Wisconsin breast cancer dataset contained 16 incomplete examples which were not used. The breast cancer and sonar data were obtained from the UCI repository while the heart data were obtained from STATLOG. Each dataset was randomly partitioned into 60% training and 40% test sets. The reported results are the averages over 30 random partitions. The kernel parameters for K_1 and K_2 are given in Table 1 by d and σ respectively. For each of the kernel matrices, an SVM is trained using the training block K_{tr} and tested using the mixed block K_{trt} as defined in (9). Finally, the margin for the initial kernel matrices K_i is compared with the margin for the optimal K^*, as is the test set performance. First of all, notice that not every K_i gives rise to a linearly separable embedding of the training data, in which case no hard margin classifier can be found (indicated with a dash). The matrix K^*, however, always allows the training of a hard margin SVM and its margin is indeed larger than the margin for each of the different components K_i; this is consistent with the SDP optimization in (12). Furthermore, the number of test set errors is smaller for K^* than for each of the different components K_i. This supports the use of the error bound (3).

Table 1. Margin and number of test-set errors (TSE) for SVMs trained and tested with the initial kernel matrices K_1, K_2, K_3 and with the optimal kernel matrix K^*, learned using semi-definite programming (12) with $c = \sum_i \text{trace}(K_i)$. A dash means that no hard margin classifier could be found.

	K_1	K_2	K_3	K^*
Breast cancer	$d = 2$	$\sigma = 0.5$		
margin	0.010	0.136	-	0.300
TSE	19.7	28.8		11.4
Sonar	$d = 2$	$\sigma = 0.1$		
margin	0.035	0.198	0.006	0.352
TSE	15.5	19.4	21.9	13.8
Heart	$d = 2$	$\sigma = 0.5$		
margin	-	0.159	-	0.285
TSE		49.2		36.6

The result for the heart dataset is worth noting—although K_1 and K_3 fail to give rise to a linearly separable embedding, combining them linearly with K_2 improves margin and test set performance significantly.

7. Conclusions

In this paper we have presented a new method for learning a kernel matrix from data, that makes use of Semi-Definite Programming (SDP) ideas. This approach is motivated by the fact that every symmetric, positive definite matrix can be considered as a kernel matrix (corresponding to a certain embedding of a finite set of data) and vice versa. Secondly, SDP deals in essence with the optimization of convex cost functions over the convex cone of positive semi-definite matrices (or convex subsets of this cone). Thus convex optimization and machine learning concerns merge to provide a powerful method for learning the kernel matrix with SDP.

The learning process is conducted in a transductive setting—using the labelled data one can learn a good embedding, which can then be applied to the unlabelled part of the data. After deriving a new generalization bound that gives rise to a convex cost function, we impose convex constraints that control the capacity of the search space of possible kernels and yield a meaningful learning procedure that can be implemented by SDP. Positive empirical results on standard benchmark datasets prove the power of this novel approach to kernel-based learning.

Appendix: Proof of Theorem 1

For a function $g : \mathcal{X} \times \mathcal{Y} \to \Re$, define

$$\begin{aligned} \hat{\mathbf{E}}_1 g(X,Y) &= \frac{1}{n}\sum_{i=1}^{n} g(X_i, Y_i), \\ \hat{\mathbf{E}}_2 g(X,Y) &= \frac{1}{n}\sum_{i=1}^{n} g(X_{n+i}, Y_{n+i}). \end{aligned}$$

The proof of the first part involves the following five steps:

Step 1. For any class F of real functions defined on $\mathcal{X}$,

$$\begin{aligned} &\sup_{f\in F} \mathrm{er}(f) - \hat{\mathbf{E}}_1 \phi_\gamma(Yf(X)) \\ &\quad \le \sup_{f\in F} \hat{\mathbf{E}}_2 \phi_\gamma(Yf(X)) - \hat{\mathbf{E}}_1 \phi_\gamma(Yf(X)). \end{aligned}$$

To see this, notice that $\mathrm{er}(f)$ is the average over the test set of the indicator function of $Yf(X) \le 0$, and that $\phi_\gamma(Yf(X))$ bounds this function.

Step 2. For any class G of $[0,1]$-valued functions,

$$\Pr\left(\sup_{g\in G}\hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g \geq \mathbf{E}\left(\sup_{g\in G}\hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g\right) + \epsilon\right) \leq \exp\left(\frac{-\epsilon^2 n}{4}\right),$$

where the expectation is over the random permutation. This follows from McDiarmid's inequality (McDiarmid, 1989). To see this, we need to define the random permutation π using a set of $2n$ independent random variables. To this end, choose $\pi_1,\ldots,\pi_{2n}$ uniformly at random from the interval $[0,1]$. These are almost surely distinct. For $j = 1,\ldots,2n$, define $\pi(j) = |\{i : \pi_i \leq \pi_j\}|$, that is, $\pi(j)$ is the position of π_j when the random variables are ordered by size. It is easy to see that, for any g, $\hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g$ changes by no more than $2/n$ when one of the π_i changes. McDiarmid's inequality implies the result.

Step 3. For any class G of $[0,1]$-valued functions,

$$\mathbf{E}\left(\sup_{g\in G}\hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g\right) \leq \hat{R}_{2n}(G) + \frac{4}{\sqrt{n}},$$

where $\hat{R}_{2n}(G) = \mathbf{E}\sup_{g\in G}\frac{1}{n}\sum_{i=1}^{2n}\sigma_i g(X_i)$, and the expectation is over the independent, uniform, $\{\pm 1\}$-valued random variables $\sigma_1,\ldots,\sigma_{2n}$. This result is essentially Lemma 3 of Bartlett and Mendelson (2001); that lemma contained a similar bound for i.i.d. X_i, but the same argument holds for fixed X_i, randomly permuted.

Step 4. If the class F of real-valued functions defined on $\mathcal{X}$ is closed under negations, $\hat{R}_{2n}(\phi_\gamma\circ F) \leq \frac{1}{\gamma}\hat{R}_{2n}(F)$, where each $f \in F$ defines a $g \in \phi_\gamma \circ F$ by $g(x,y) = \phi_\gamma(yf(x))$. This bound is an adaptation of the contraction lemma in Ledoux & Talagrand (1991).

Step 5. For the class F_K of kernel expansions (13) satisfying (14),

$$\hat{R}_{2n}(F_K) \leq \frac{1}{n}\sqrt{\mathrm{trace}(K)}.$$

This is Lemma 26 of Bartlett and Mendelson (2001).

Combining gives the first part of the theorem. To prove the second part, notice (as in the proof of Lemma 26 of Bartlett and Mendelson, 2001) that

$$\begin{aligned}\hat{R}_{2n}(F_B) &= \frac{1}{n}\mathbf{E}\sup_{\beta}\sup_{\|w\|\leq 1}\langle w, \sum_{i=1}^{2n}\sigma_i\Phi(X_i)\rangle \\ &= \frac{1}{n}\mathbf{E}\sup_{\beta}\left\|\sum_{i=1}^{2n}\sigma_i\Phi(X_i)\right\| \\ &\leq \frac{1}{n}\sqrt{\mathbf{E}\sup_{\beta}\sigma' K\sigma},\end{aligned}$$

where $\sigma = (\sigma_1,\ldots,\sigma_{2n})$ is the vector of Rademacher random variables, and the supremum is over $\beta = (\beta_1,\ldots,\beta_m)$ for which the matrix $K = \sum_{j=1}^m \beta_j K_j$ satisfies the conditions $K \succeq 0$ and $\mathrm{trace}(K) \leq B$. Now, $\mathrm{trace}(K) = \sum_{j=1}^m \beta_j \mathrm{trace}(K_j)$, and each trace in the sum is positive, so the supremum must be achieved for $\mathrm{trace}(K) = B$. So we can write that $\hat{R}_{2n}(F_B)$ is no more than

$$\frac{\sqrt{B\mathbf{E}\sup\sigma' K\sigma}}{n},$$

where the supremum is over $K = \sum_j \beta_j K_j$ satisfying $K \succeq 0$ and $\mathrm{trace}(K) = 1$. Defining $C = \mathbf{E}\sup\sigma' K\sigma$ gives the result.

Notice that $\sigma' K\sigma$ is no more than $\lambda\|\sigma\|^2 = n\lambda$, where λ is the maximum eigenvalue of K. Using $\lambda \leq \mathrm{trace}(K) = 1$ shows that $C \leq n$.

Acknowledgements

We would like to acknowledge support from ONR MURI N00014-00-1-0637 and NSF grant IIS-9988642.

References

Bartlett, P. L. & Mendelson, S. (2001). *Rademacher and gaussian complexities: Risk bounds and structural results.* Technical Report, Australian National University.

Bennett, K. P. & Bredensteiner, E. J. (2000). Duality and geometry in SVM classifiers. *Proceedings of the Seventeenth International Conference on Machine Learning.* San Francisco: Morgan Kaufmann.

Cristianini, N. & Shawe-Taylor J. (2000). *An Introduction to Support Vector Machines.* Cambridge: Cambridge University Press.

Cristianini, N., Shawe-Taylor J., Kandola J. & Elisseeff A. (2001). On kernel target alignment. *Advances in Neural Information Processing Systems, 14.* Cambridge, MA: MIT Press.

Ledoux, M. & Talagrand, M. (1991). *Probability in Banach Spaces: Isoperimetry and Processes.* NY: Springer-Verlag.

McDiarmid, C. (1989). On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge: Cambridge University Press.

Schölkopf, B. & Smola, A. (2002). *Learning with Kernels.* Cambridge, MA: MIT Press.

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