Ann. Inst. Statist. Math.
Vol. 55, No. 2, 391–408 (2003)

DEALING WITH LARGE DIAGONALS IN KERNEL MATRICES

JASON WESTON[1], BERNHARD SCHÖLKOPF[1], ELEAZAR ESKIN[2], CHRISTINA LESLIE[2,3] AND WILLIAM STAFFORD NOBLE[4]

[1] *Max-Planck-Institut für biologische Kybernetik, Spemannstr. 38, D-72076 Tübingen, Germany,* e-mail: jason.weston@tuebingen.mpg.de; bernhard.schoelkopf@tuebingen.mpg.de

[2] *Department of Computer Science, Columbia University, 450 Computer Science Building, MC 0401, 1214 Amsterdam Avenue, New York, NY 10027, U.S.A.,* e-mail: eeskin@cs.columbia.edu; cleslie@cs.columbia.edu

[3] *Center for Computational Biology (C2B2), Russ Berrie Pavilion, Columbia University, 1150 St. Nicholas Avenue, New York, NY 10032, U.S.A.*

[4] *Department of Genome Sciences, University of Washington, Health Sciences Center, Box 357730, 1705 NE Pacific Street, Seattle, WA 98195, U.S.A.,* e-mail: noble@gs.washington.edu

(Received June 3, 2002; revised September 9, 2002)

Abstract. In kernel methods, all the information about the training data is contained in the Gram matrix. If this matrix has large diagonal values, which arises for many types of kernels, then kernel methods do not perform well. We propose and test several methods for dealing with this problem by reducing the dynamic range of the matrix while preserving the positive definiteness of the Hessian of the quadratic programming problem that one has to solve when training a Support Vector Machine, which is a common kernel approach for pattern recognition.

Key words and phrases: Kernel methods, Support Vector Machines, pattern recognition, bioinformatics, microarray data analysis, transduction, regularization.

1. Introduction

The present paper is structured as follows. We start with a concise summary of some aspects of Support Vector Machines (SVMs), with a focus on a particular formulation which we will use subsequently. The discussion assumes some previous knowledge of SVMs. In Section 2, we introduce the problem of large diagonals, followed by our proposed method to handle it (Section 3). Section 4 presents experiments, and Section 5 summarizes our conclusions.

SVMs and related kernel methods can be considered an approximate implementation of the structural risk minimization principle suggested by Vapnik (1979). To this end, they minimize an objective function containing a trade-off between two goals, that of minimizing the training error, and that of minimizing a regularization term. In SVMs, the latter is a function of the margin of separation between the two classes in a binary pattern recognition problem. This margin is measured in a so-called feature space $\mathcal{H}$ which is a Hilbert space into which the training patterns are mapped by means of a map

$$\Phi : \mathcal{X} \to \mathcal{H}. \tag{1.1}$$

Here, the input domain $\mathcal{X}$ can be an arbitrary nonempty set. The art of designing an SVM for a task at hand consists of selecting a feature space with the property that dot

products between mapped input points, $\langle \Phi(x), \Phi(x') \rangle$, can be computed in terms of a so-called *kernel*

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{1.2}$$

which can be evaluated efficiently. Such a kernel necessarily belongs to the class of *positive definite kernels* (e.g. Berg *et al.* (1984)), i.e., it satisfies

$$\sum_{i,j=1}^{m} a_i a_j k(x_i, x_j) \geq 0 \tag{1.3}$$

for all $a_i \in \mathbb{R}$, $x_i \in \mathcal{X}$, $i = 1, \ldots, m$. The kernel can be thought of as a nonlinear similarity measure that corresponds to the dot product in the associated feature space. Using k, we can carry out all algorithms in $\mathcal{H}$ that can be cast in terms of dot products, examples being SVMs and PCA (for an overview, see Schölkopf and Smola (2002)). To train a classifier $f(x) = \operatorname{sgn}(\langle \boldsymbol{w}, \Phi(x) \rangle + b)$, where

$$\boldsymbol{w} = \sum_{j=1}^{m} a_j \Phi(x_j), \tag{1.4}$$

the SVM pattern recognition algorithm minimizes the quadratic form

$$\|\boldsymbol{w}\|^2 = \sum_{i,j=1}^{m} a_i a_j K_{ij} \tag{1.5}$$

subject to the constraints

$$y_i[\langle \Phi(x_i), \boldsymbol{w} \rangle + b] \geq 1, \quad \text{i.e.,} \quad y_i \left[\sum_{j=1}^{m} a_j K_{ij} + b \right] \geq 1. \tag{1.6}$$

This formulation comes from the usual primal formulation rewritten using the representer theorem, which says that the solution can be rewritten as $\sum_i \alpha_i \Phi(x_i)$. Also, we are considering the zero training error case. Nonzero training errors are incorporated as suggested by Cortes and Vapnik (1995), for all $i \in \{1, \ldots, m\}$. Here,

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\} \tag{1.7}$$

are the training examples, and

$$K_{ij} := k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{1.8}$$

is the Gram matrix.

Note that the regularizer (1.5) equals the squared length of the weight vector $\boldsymbol{w}$ in $\mathcal{H}$. Sometimes, a modification of this approach is considered, where the regularizer

$$\sum_{i=1}^{m} a_i^2 \tag{1.9}$$

is used instead of (1.5). Whilst this is no longer the squared length of a weight vector in the feature space $\mathcal{H}$, it is instructive to re-interpret it as the squared length in a different feature space, namely in $\mathbb{R}^m$.

To this end, we consider the feature map

$$\Phi_m(x) := (k(x, x_1), \ldots, k(x, x_m))^\top, \tag{1.10}$$

sometimes called the *empirical kernel map* (Tsuda (1999); Schölkopf and Smola (2002)) (cf. also Hastie and Tibshirani (1990)). In this case, the (primal) SVM optimization problem consists in minimizing

$$\|\boldsymbol{a}\|^2 \tag{1.11}$$

subject to

$$y_i[\langle \Phi_m(x_i), \boldsymbol{a} \rangle + b] \geq 1 \tag{1.12}$$

for all $i \in \{1, \ldots, m\}$, where $\boldsymbol{a} = (a_1, \ldots, a_m)^\top \in \mathbb{R}^m$. In view of (1.10), however, the constraints (1.12) are equivalent to $y_i[\sum_{j=1}^m a_j K_{ij} + b] \geq 1$, i.e. to (1.6), while the regularizer $\|\boldsymbol{a}\|^2$ equals (1.9).

Therefore, using the regularizer (1.9) and the original kernel corresponds to a standard SVM with the desired kernel used as input features. This SVM operates in an m-dimensional feature space with the standard SVM regularizer, i.e., the squared weight of the weight vector in the feature space. We can thus train a classifier using the regularizer (1.9) simply by training an SVM with the kernel

$$k_m(x, x') := \langle \Phi_m(x), \Phi_m(x') \rangle, \tag{1.13}$$

and thus, by definition of Φ_m, using the Gram matrix

$$K_m = KK^\top, \tag{1.14}$$

where K denotes the Gram matrix of the original kernel. The last equation shows that when employing the empirical kernel map, it is not necessary to use a positive definite kernel. The reason is that no matter what K is, the Gram matrix $KK^\top$ is always positive definite, which is sufficient for an SVM. (Here, as in (1.3), we allow for a nonzero null space in our usage of the concept of positive definiteness.)

In summary then, when using the modified regularizer we can simply train a standard (linear) SVM with the empirical kernel map as input (feature) vectors. This will allow us to consider non-positive definite kernels, which will be useful for the problem of kernel matrices with large diagonals.

2. The problem of large diagonals

An important feature of kernel methods is that the input domain $\mathcal{X}$ does not have to be a vector space. The inputs might just as well be discrete objects such as strings. Moreover, the map Φ might compute rather complex features of the inputs. Examples thereof are polynomial kernels (Boser *et al.* (1992)), where Φ computes all products (of a given order) of entries of the inputs (in this case, the inputs are vectors), and string

kernels (Watkins (2000); Haussler (1999); Lodhi *et al.* (2002)), which, for instance, can compute the number of common substrings (not necessarily contiguous) of a certain length $n \in \mathbb{N}$ of two strings x, x' in $O(n|x||x'|)$ time. Here, we assume that x and x' are two finite strings over a finite alphabet Σ. For the string kernel of order n, a basis for the feature space consists of the set of all strings of length n, Σ^n. In this case, Φ maps a string x into a vector whose entries indicate whether the respective string of length n occurs as a substring in x. By construction, these will be rather sparse vectors—a large number of *possible* substrings do not occur in a given string. Therefore, the dot product of two *different* vectors will take a value which is much smaller than the dot product of a vector with itself. This can also be understood as follows: any string shares *all* substrings with itself, but relatively few substrings with another string. Therefore, it will typically be the case that we are faced with *large diagonals*. By this we mean that, given some training inputs $x_1, \ldots, x_m$, we have

$$k(x_i, x_i) \gg |k(x_i, x_j)| \quad \text{for} \quad x_i \neq x_j, \ i, j \in \{1, \ldots, m\}. \tag{2.1}$$

The diagonal terms $k(x_i, x_i)$ are necessarily nonnegative for positive definite kernels, hence no modulus on the left hand side. In this case, the associated Gram matrix will have large diagonal elements.

Note that in the machine learning literature, this problem is sometimes referred to as *diagonal dominance*. However, the latter term is used in linear algebra for matrices where the absolute value of each diagonal element is greater than the sum of the absolute values of the other elements in its row (or column). Real diagonally dominant matrices with positive diagonal elements are positive definite.

Let us next consider an innocuous application which is rather popular with SVMs: handwritten digit recognition. We suppose that the data are handwritten characters represented by images in $[0, 1]^N$ (here, $N \in \mathbb{N}$ is the number of pixels), and that only a small fraction of the images is ink (i.e. few entries take the value 1). In that case, we typically have $\langle x, x \rangle > \langle x, x' \rangle$ for $x \neq x'$, and thus the polynomial kernel (which is what most commonly is used for SVM handwritten digit recognition)

$$k(x, x') = \langle x, x' \rangle^d \tag{2.2}$$

satisfies $k(x, x) \gg |k(x, x')|$ already for moderately large d—it has large diagonals.

Note that as in the case of the string kernel, one can also understand this phenomenon in terms of the sparsity of the vectors in the feature space. It is known that the polynomial kernel of order d effectively maps the data into a feature space whose dimensions are spanned by all products of d pixels. Clearly, if some of the pixels take the value zero to begin with, then an even larger fraction of all possible products of d pixels (assuming $d > 1$) will be zero. Therefore, the sparsity of the vectors will increase with d.

In practice, it has been observed that SVMs do not work well in this situation. Empirically, they work much better if the images are scaled such that the individual pixel values are in $[-1, 1]$, i.e., that the background value is -1. In this case, the data vectors are less sparse and thus further from being orthogonal.

Indeed, large diagonals correspond to approximate orthogonality of any two different patterns mapped into the feature space. To see this, assume that $x \neq x'$ and note that

due to $k(x,x) \gg |k(x,x')|$,

$$\cos(\angle(\Phi(x),\Phi(x'))) = \frac{\langle\Phi(x),\Phi(x')\rangle}{\sqrt{\langle\Phi(x),\Phi(x)\rangle\,\langle\Phi(x'),\Phi(x')\rangle}} = \frac{k(x,x')}{\sqrt{k(x,x)k(x',x')}} \approx 0.$$

In some cases, an SVM trained using a kernel with large diagonals will *memorize* the data. Let us consider a simple toy example, using X as data matrix and Y as label vector, respectively:

$$X = \begin{pmatrix} 1&0&0&9&0&0&0&0&0&0\\ 1&0&0&0&0&8&0&0&0&0\\ 1&0&0&0&0&0&0&9&0&0\\ 0&0&9&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&8&0&0&0\\ 0&0&0&0&0&0&0&0&0&9 \end{pmatrix}, \quad Y = \begin{pmatrix} +1\\ +1\\ +1\\ -1\\ -1\\ -1 \end{pmatrix}.$$

The Gram matrix for these data (using the linear kernel $k(x,x') = \langle x,x'\rangle$) is

$$K = \begin{pmatrix} 82&1&1&0&0&0\\ 1&65&1&0&0&0\\ 1&1&82&0&0&0\\ 0&0&0&81&0&0\\ 0&0&0&0&64&0\\ 0&0&0&0&0&81 \end{pmatrix}.$$

A standard SVM finds the solution $f(x) = \operatorname{sgn}(\langle w,x\rangle + b)$ with

$$w = (0.04, 0, -0.11, 0.11, 0, 0.12, -0.12, 0.11, 0, -0.11)^\top, \qquad b = -0.02.$$

It can be seen from the coefficients of the weight vector w that this solution has but memorized the data: all the entries which are larger than 0.1 in absolute value correspond to dimensions which are nonzero only for *one* of the training points. We thus end up with a look-up table. A *good* solution for a linear classifier, on the other hand, would be to just choose the first feature, e.g., $f(x) = \operatorname{sgn}(\langle w,x\rangle + b)$, with $w = (2,0,0,0,0,0,0,0,0,0)^\top$, $b = -1$.

In the limit where K approaches a diagonal matrix D, the data memorization property can be seen analytically. Using bold face notation for coefficient vectors (with $\boldsymbol{e}$ denoting the vector whose coefficients are all 1), we have in this case

$$\boldsymbol{\alpha} = D^{-1}(\boldsymbol{y} - b\boldsymbol{e}) \tag{2.3}$$

$$b = \frac{\boldsymbol{e}^\top D^{-1}\boldsymbol{y}}{\boldsymbol{e}^\top D^{-1}\boldsymbol{e}} \tag{2.4}$$

and all training points are classified correctly. This can be deduced by (as all points are support vectors) solving for $\boldsymbol{\alpha}$ with equality constraints rather than inequalities, and then solving for b. Note that this is only true under the assumption that D is invertible (i.e. that $k(x_i,x_i) > 0$ for all i).

3. Methods to reduce large diagonals

The basic idea that we are proposing is very simple indeed. We would like to use a nonlinear transformation to reduce the size of the diagonal elements, or, more generally, to reduce the dynamic range of the Gram matrix entries. The only difficulty is that if we simply do this, we have no guarantee that we end up with a Gram matrix that is still positive definite. To ensure that it is, we can use methods of functional calculus for matrices. In the experiments we will mainly use a simple special case of the below. Nevertheless, let us introduce the general case, since we think it provides a useful perspective on kernel methods, and on the transformations that can be done on Gram matrices.

Let K be a symmetric $m \times m$ matrix with eigenvalues in $[\lambda_{\min}, \lambda_{\max}]$. Consider a continuous function f on $[\lambda_{\min}, \lambda_{\max}]$.

Functional calculus provides a method to define a unique symmetric matrix, denoted by $f(K)$, which has eigenvalues in $[f(\lambda_{\min}), f(\lambda_{\max})]$. This matrix can be computed via a Taylor series expansion in K, or using the eigenvalue decomposition of K: If $K = S^\top DS$ (with D diagonal and S unitary), then $f(K) = S^\top f(D)S$, where $f(D)$ is the diagonal matrix with elements $f(D)_{ii} = f(D_{ii})$.

The convenient property of this procedure is that we can treat functions of symmetric matrices just like functions on $\mathbb{R}$; in particular, we have, for $\alpha \in \mathbb{R}$, and real continuous functions f, g defined on $[\lambda_{\min}, \lambda_{\max}]$, (below, $\sigma(K)$ denotes the spectrum of K)

$$
\begin{aligned}
&(\alpha f + g)(K) = \alpha f(K) + g(K) \\
&(fg)(K) = f(K)g(K) = g(K)f(K) \\
&\|f\|_{\infty,\sigma(K)} = \|f(K)\| \\
&\sigma(f(K)) = f(\sigma(K)).
\end{aligned}
$$

In technical terms, the C^*-algebra generated by K is isomorphic to the set of continuous functions on $\sigma(K)$.

For our problems, functional calculus can be applied in the following way. We start off with a positive definite matrix K which has large diagonal values. We then reduce its dynamic range by elementwise application of a nonlinear function, such as $\varphi(x) = \log(x + 1)$ or $\varphi(x) = \operatorname{sgn}(x) \cdot |x|^p$ with $0 < p < 1$. This will lead to a matrix which may no longer be positive definite. However, it is still symmetric, and hence we can apply functional calculus. As a consequence of $\sigma(f(K)) = f(\sigma(K))$, we just need to apply a function f which maps to $\mathbb{R}_0^+$. This will ensure that all eigenvalues of $f(K)$ are nonnegative, hence $f(K)$ will be positive definite.

One can use these observations to design the following scheme.

For positive definite K,

1. compute the positive definite matrix $A := \sqrt{K}$
2. reduce the dynamic range of the entries of A by applying an elementwise transformation φ, leading to a symmetric matrix A_φ
3. compute the positive definite matrix $K' := (A_\varphi)^2$ and use it in subsequent processing. The entries of K' will be the "effective kernel," which in this case is no longer given in analytic form.

Note that in this procedure, if φ is the identity, then we have $K = K'$.

Experimentally, this scheme works rather well. However, it has one downside: since we no longer have the kernel function in analytic form, our only means of evaluating it is to include all test inputs (not the test labels, though) into the matrix K. In other words, K should be the Gram matrix computed from the observations $x_1, \ldots, x_{m+n}$ where $x_{m+1}, \ldots, x_{m+n}$ denote the test inputs. We thus need to know the test inputs already during training. This setting is sometimes referred to as *transduction* (Vapnik (1998)).

If we skip the step of taking the square root of K, we can alleviate this problem. In that case, the only application of functional calculus left is a rather trivial one, that of computing the square of K. The $m \times m$ submatrix of K^2 which in this case would have to be used for training then equals the Gram matrix when using the empirical kernel map

$$\Phi_{m+n}(x) = (k(x, x_1), \ldots, k(x, x_{m+n}))^\top. \tag{3.1}$$

For the purposes of computing dot products, however, this can approximately be replaced by the empirical kernel map in terms of the training examples only, i.e., by (1.10). The justification for this is that for large $r \in \mathbb{N}$,

$$\frac{1}{r} \langle \Phi_r(x), \Phi_r(x') \rangle \approx \int_{\mathcal{X}} k(x, x'') k(x', x'') dP(x''),$$

where P is assumed to be the distribution of the inputs. Therefore, we have $\frac{1}{m} \langle \Phi_m(x), \Phi_m(x') \rangle \approx \frac{1}{m+n} \langle \Phi_{m+n}(x), \Phi_{m+n}(x') \rangle$. Altogether, the procedure then boils down to simply training an SVM using the empirical kernel map in terms of the training examples and the transformed kernel function $\varphi(k(x, x'))$. This is what we will use in the experiments below unless stated otherwise. Note that a subset of the results of this paper has appeared in (Schölkopf *et al.* (2002)).

4. Experiments

4.1 *Artificial data*

We first constructed a set of artificial experiments which produce kernels exhibiting large diagonals. The experiments are as follows: a string classification problem, a microarray cancer detection problem supplemented with extra noisy features and a toy problem whose labels depend upon hidden variables; the visible variables are nonlinear combinations of those hidden variables.

4.1.1 *String classification*

We considered the following classification problem. Two classes of strings are generated with equal probability by two different Markov models. Both classes of strings consist of letters from the same alphabet of $a = 20$ letters, and strings from both classes are always of length $n = 20$. Strings from the negative class are generated by a model where transitions from any letter to any other letter are equally likely. Strings from the positive class are generated by a model where transitions from one letter to itself (so the next letter is the same as the last) have probability 0.43, and all other transitions have probability 0.03. For both classes the starting letter of any string is equally likely to be any letter of the alphabet. The task then is to predict which class a given string belongs to. To map these strings into a feature space, we used the string subsequence kernel employed by Lodhi *et al.* (2002) for text categorization. As described above, this

kernel is an inner product in a feature space consisting of all subsequences of length l. A subsequence is any ordered sequence of l characters occurring in the text though not necessarily contiguously. In the present application, the subsequences are weighted by an exponentially decaying factor λ of their full length in the text, hence emphasizing those occurrences which are close to contiguous. A direct computation of this feature vector would involve a prohibitive amount of computation, a method of computing the inner product efficiently using a dynamic programming technique is described by Lodhi *et al.* (2002). For our problem we chose the parameters $l = 3$ and $\lambda = \frac{1}{4}$.

We generated 50 such strings and used the string subsequence kernel with $\lambda = 0.25$. We split the data into 25 for training and 25 for testing in 20 separate trials. We measured the success of a method by calculating the mean classification loss on the test sets. Figure 1 shows four strings from the dataset and the computed kernel matrix for these strings (the matrix was rescaled by dividing by the largest entry). Note that the diagonal entries are much larger than the off-diagonals because a long string has a large number of subsequences that are shared with no other strings in the dataset apart from itself. However, information relevant to the classification of the strings is contained in the matrix. This can be seen by computing the mean kernel value between two examples of the positive class which is equal to 0.0003 ± 0.0011, whereas the mean kernel value between two examples of opposite classes is 0.00002 ± 0.00007. Although the numbers are very small, this captures that the positive class have more in common with each other than with random strings (they are more likely to have repeated letters).

string	class
qqbqqnshrtktfhhaahhh	+ve
abajahnaajjjjiiiittt	+ve
sdolncqniflmmpcrioog	−ve
reaqhcoigealgqjdsdgs	−ve

$$K = \begin{pmatrix} 0.6183 & 0.0133 & 0.0000 & 0.0000 \\ 0.0133 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.4692 & 0.0002 \\ 0.0000 & 0.0000 & 0.0002 & 0.4292 \end{pmatrix}$$

Fig. 1. Four strings and their kernel matrix using the string subsequence kernel with $\lambda = 0.25$. Note that the diagonal entries are much larger than the off-diagonals because a long string has a large number of subsequences that are shared with no other strings in the dataset apart from itself.

If the original kernel is denoted as an inner product $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$, then we employ the kernel $k(x, y) = \langle \Phi(x), \Phi(y) \rangle^p$ where $0 < p < 1$ to solve the diagonal dominance problem. We will refer to this kernel as a *subpolynomial* one. When the input data can take negative values we will use $k(x, y) = \text{sgn} \langle \Phi(x), \Phi(y) \rangle * | \langle \Phi(x), \Phi(y) \rangle |^p$ to retain the information given by the sign. If there are no negative values the two choices are equivalent, so we instead write the simpler version.

As this kernel may no longer be positive definite we use the method described in Section 1, employing the empirical kernel map to embed our distance measure into a feature space. Results of using our method to solve the problem of large diagonals is

Table 1. Results on a string classification problem of using the string subsequence kernel (top row) and with a further nonlinear map via an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		classification loss
original k, $k(x,y) = \langle \Phi(x), \Phi(y) \rangle$		0.36 ± 0.018
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 0.25$	0.18 ± 0.021
$k_{emp}(x,y) = \langle \Phi(x), \Phi(y) \rangle^p$	$p = 1$	0.30 ± 0.011
	$p = 0.9$	0.25 ± 0.013
	$p = 0.8$	0.20 ± 0.014
	$p = 0.7$	0.15 ± 0.013
	$p = 0.6$	$\mathbf{0.13 \pm 0.010}$
	$p = 0.5$	0.14 ± 0.008
	$p = 0.4$	0.15 ± 0.010
	$p = 0.3$	0.15 ± 0.008
	$p = 0.2$	0.17 ± 0.010
	$p = 0.1$	0.21 ± 0.013

given in Table 1. In this, and subsequent tables, each algorithm tried is listed, together with the values of its hyperparameters and the resulting mean classification loss and its standard error. The method provides, with the optimum choice of the free parameter, a reduction from a loss of 0.36 ± 0.018 with the original kernel to 0.13 ± 0.001 with $p = 0.6$. Although we do not provide methods for choosing this free parameter, it is straightforward to apply conventional techniques of model selection (such as cross validation) to achieve this goal.

We also performed some further experiments which we will briefly discuss. The subpolynomial kernel adds a kind of nonlinearity to the original kernel, such that if $p = 1$ is overfitting then reducing p can reduce this overfitting, until finally as $p \to 0$, underfitting occurs. To check that the nonlinear mapping really is doing something different to standard nonlinear mappings such as RBF kernels: $k(x,y) = \exp(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2})$, we also measured their test performance. We thus give in our experimental results the validation error score of the optimal RBF kernel (given knowledge of the labels) using the possible values $\sigma = (2^{-10}, 2^{-9}, \ldots, 2^{10})$, i.e. selecting the one with the lowest validation error. Although this value is impossible to choose in real applications it serves to show whether the RBF kernel could help at all: if even the best choice of RBF is still not better than the linear rule then it is clearly not useful. An RBF kernel can help if nonlinear relations between variables can help, but we do not believe it can help with the diagonal dominance problem. In this problem an RBF kernel does improve over the linear one, but not by as much as the subpolynomial kernel (see the table for results), we thus believe in this problem there is an issue of required nonlinearity as well as a problem of large diagonals. We will see in later experiments that the subpolynomial kernel can improve over the linear one even when the RBF kernel only ever makes results worse.

Next, to check that the result is a feature of kernel algorithms, and not something peculiar to SVMs, we also applied the same kernels to another algorithm, kernel 1-nearest neighbor. Using the original kernel matrix yields a loss of 0.43 ± 0.0085 whereas the subpolynomial method again improves the results, using $p = 0.6$ yields 0.22 ± 0.0011

and $p = 0.3$ (the optimum choice) yields 0.17 ± 0.001. Finally, we tried some alternative proposals for reducing the large diagonal effect. We tried using Kernel PCA to extract features as a pre-processing to training an SVM. The intuition behind using this is that features contributing to the large diagonal effect may have low variance and would thus be removed by KPCA. KPCA did improve performance a little, but still worse than the subpolynomial method. The best result was found by extracting 15 features (from the kernel matrix of 50 examples) yielding a loss of 0.23 ± 0.001.

We also tried a "naive" method of subtracting a constant from the diagonal of the kernel matrix, i.e. $K = K - \lambda K^D$ where K is the Gram matrix and K^D is a diagonal matrix where $K^D_{ii} = K_{ii}$. The largest choice of λ before K is no longer positive definite is $\lambda = 0.5$ which yields a loss of 0.25 ± 0.014. Note that in this method, it is not necessary to use the empirical kernel map.

4.1.2 *Microarray data with added noise*

We next considered the microarray classification problem of Alon *et al.* (1999) (see also Guyon *et al.* (2002) for a treatment of this problem with SVMs). In this problem one must distinguish between cancerous and normal tissue in a colon cancer problem given the expression of genes measured by microarray technology. In this problem one does not encounter large diagonals, however we augmented the original dataset with extra noisy features to simulate such a problem. The original data has 62 examples (22 positive, 40 negative) and 2000 features (gene expression levels of the tissues samples). We added a further 10,000 features to the dataset, such that for each example a randomly chosen 100 of these features are chosen to be nonzero (taking a random value between 0 and 1) and the rest are equal to zero. This creates a kernel matrix with large diagonals. The first 4×4 entries of the kernel matrix of a linear kernel on the colon cancer problem before (K) and after (K') adding 10,000 sparse, noisy features. The added features are designed to create a kernel matrix with a large diagonal.

$$K = \begin{pmatrix} 1.00 & 0.41 & 0.33 & 0.42 \\ 0.41 & 1.00 & 0.17 & 0.39 \\ 0.33 & 0.17 & 1.00 & 0.61 \\ 0.42 & 0.39 & 0.61 & 1.00 \end{pmatrix}, \quad K' = \begin{pmatrix} 39.20 & 0.41 & 0.33 & 0.73 \\ 0.41 & 37.43 & 0.26 & 0.88 \\ 0.33 & 0.26 & 31.94 & 0.61 \\ 0.73 & 0.88 & 0.61 & 35.32 \end{pmatrix}.$$

The problem is again an artificial one demonstrating the problem of large diagonals, however this time the feature space is rather more explicit rather than the implicit one induced by string kernels. In this problem we can clearly see the large diagonal problem is really a special kind of feature selection problem. As such, feature selection algorithms should be able to help improve generalizability, unfortunately most feature selection algorithms work on explicit features rather than implicit ones induced by kernels.

Performance of methods was measured using 10-fold cross validation, which was repeated 10 times. Due to the unbalanced nature of the number of positive and negative examples in this data set we measured the error rates using the balanced loss:

$$\ell_{\text{bal}}(y, \hat{y}) = \frac{1}{2}\left(\frac{\#\{\hat{y} : y = 1 \wedge \hat{y} = -1\}}{\#\{y : y = 1\}}\right) + \frac{1}{2}\left(\frac{\#\{\hat{y} : y = -1 \wedge \hat{y} = 1\}}{\#\{y : y = -1\}}\right) \tag{4.1}$$

where y are the true labels and $\hat{y}$ are the estimates. Chance level is a loss of 0.5 regardless of the ratio of positive to negative examples. On this problem (with the added noise) an SVM using the original kernel does not perform better than chance. The same is true of an RBF kernel.

Table 2. Results on a colon cancer classification problem with added noise of using a linear kernel (top row) and an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		balanced loss
original k, $k(x,y) = \langle x,y\rangle$		0.49 ± 0.005
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 1$	0.50 ± 0.000
$k_{emp}(x,y) = \operatorname{sgn}\langle x,y\rangle * \|\langle x,y\rangle\|^p$	$p = 0.95$	0.35 ± 0.017
	$p = 0.9$	0.30 ± 0.017
	$p = 0.8$	0.25 ± 0.018
	$p = 0.7$	$\mathbf{0.22 \pm 0.017}$
	$p = 0.6$	0.23 ± 0.017
	$p = 0.5$	0.25 ± 0.019
	$p = 0.4$	0.28 ± 0.019
	$p = 0.3$	0.29 ± 0.018
	$p = 0.2$	0.30 ± 0.019
	$p = 0.1$	0.31 ± 0.018

The results of using the original kernel, the RBF kernel and the subpolynomial method are given in Table 2. The subpolynomial kernel leads to a large improvement over using the original kernel. Its performance is close to that of an SVM on the original data without the added noise, which in this case is 0.18 ± 0.015.

4.1.3 *Hidden variable problem*

We then constructed an artificial problem where the labels can be predicted by a linear rule based upon some hidden variables. However, the visible variables are a nonlinear combination of the hidden variables combined with noise. The purpose is to show that the subpolynomial kernel is not only useful in the case of matrices with large diagonals: it can also improve results in the case where a *linear* rule already overfits. The data are generated as follows. There are 10 hidden variables: each class $y \in \{\pm 1\}$ is generated by a 10 dimensional normal distribution $N(\mu, \sigma)$ with variance $\sigma^2 = 1$, and mean $\mu = y(0.5, 0.5, \ldots, 0.5)$. We then add 10 more (noisy) features for each example, each generated with $N(0,1)$. Let us denote the 20-dimensional vector obtained this way for example i as h_i. The visible variables x_i are then constructed by taking all monomials of degree 1 to 4 of h_i. It is known that dot products between such vectors can be computed using polynomial kernels (Boser *et al.* (1992)), thus the dot product between two visible variables is

$$k(x_i, x_j) = (\langle h_i, h_j\rangle + 1)^4.$$

We compared the subpolynomial method to a linear kernel and an RBF kernel using balanced 10-fold cross validation, repeated 10 times. The results are shown in Table 3. Again, the subpolynomial kernel gives improved results.

One interpretation of these results is that if we know that the visible variables are polynomials of some hidden variables, then it makes sense to use a subpolynomial transformation to obtain a Gram matrix closer to the one we could compute if we were given the hidden variables. In effect, the subpolynomial kernel can (approximately) extract the hidden variables.

Table 3. Results on the hidden variable problem of using a linear kernel (top row) and an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		classification loss
original k, $k(x,y) = \langle x,y \rangle$		0.26 ± 0.012
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 512$	0.26 ± 0.013
$k_{emp}(x,y) = \operatorname{sgn}\langle x,y \rangle * \mid \langle x,y \rangle \mid^p$	$p = 1$	0.25 ± 0.012
	$p = 0.9$	0.23 ± 0.013
	$p = 0.8$	0.19 ± 0.012
	$p = 0.7$	0.18 ± 0.012
	$p = 0.6$	$\mathbf{0.16 \pm 0.011}$
	$p = 0.5$	$\mathbf{0.16 \pm 0.011}$
	$p = 0.4$	$\mathbf{0.16 \pm 0.011}$
	$p = 0.3$	0.18 ± 0.011
	$p = 0.2$	0.20 ± 0.012
	$p = 0.1$	0.19 ± 0.013

4.2 *Real data*

In the following experiments we apply our methods to real problems.

4.2.1 *Thrombin binding problem*

In the thrombin dataset the problem is to predict whether a given drug binds to a target site on thrombin, a key receptor in blood clotting. This dataset was used in the KDD (Knowledge Discovery and Data Mining) Cup 2001 competition and was provided by DuPont Pharmaceuticals.

In the training set there are 1909 examples representing different possible molecules (drugs), 42 of which bind. Hence the data is rather unbalanced in this respect. Each example has a fixed length vector of 139,351 binary features (variables) in $\{0,1\}$ which describe three-dimensional properties of the molecule. An important characteristic of the data is that very few of the feature entries are nonzero (0.68% of the 1909×139351 training matrix, see (Weston *et al.* (2002)) for further statistical analysis of the dataset). Thus, many of the features somewhat resemble the noisy features that we added on to the colon cancer dataset to create a large diagonal in Section 4.1.2. Indeed, constructing a kernel matrix of the training data using a linear kernel yields a matrix with a mean diagonal element of 1377.9 ± 2825 and a mean off-diagonal element of 78.7 ± 209. We compared the subpolynomial method to the original kernel and an RBF kernel using 8-fold balanced cross validation (ensuring an equal number of positive examples were in each fold). We again used the balanced loss measure given in equation (4.1). The results are given in Table 4. Once again the subpolynomial method provides improved generalization. It should be noted that feature selection and transduction methods have also been shown to improve results, above that of a linear kernel on this problem (Weston *et al.* (2002)).

4.2.2 *Lymphoma classification*

We next looked at the problem of identifying large B-Cell Lymphoma by gene expression profiling (Alizadeh *et al.* (2000)). In this problem the gene expression of 96 samples is measured with microarrays to give 4026 features. Sixty-one of the samples

Table 4. Results on the thrombin binding problem of using a linear kernel (top row) and an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		balanced loss
original k, $k(x,y) = \langle x,y\rangle$		0.30 ± 0.04
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 16$	0.30 ± 0.04
$k_{emp}(x,y) = \langle x,y\rangle^p$	$p = 0.9$	0.24 ± 0.04
	$p = 0.8$	0.24 ± 0.04
	$p = 0.7$	0.18 ± 0.03
	$p = 0.6$	0.18 ± 0.03
	$p = 0.5$	$\mathbf{0.15 \pm 0.03}$
	$p = 0.4$	0.17 ± 0.04
	$p = 0.3$	0.17 ± 0.04
	$p = 0.2$	0.18 ± 0.03
	$p = 0.1$	0.22 ± 0.05

are in classes "DLCL", "FL" or "CLL" (malignant) and 35 are labelled "otherwise" (usually normal). Although the data does not induce a kernel matrix with a very large diagonal it is possible that the large number of features induce overfitting even in a linear kernel. To examine if our method would still help in this situation we applied the same techniques as before, this time using balanced 10-fold cross validation, repeated 10 times, and measuring error rates using the balanced loss. The results are given in Table 5. The improvement given by the subpolynomial kernel suggests that overfitting in linear kernels when the number of features is large may be overcome by applying special feature maps. It should be noted that (explicit) feature selection methods have also been shown to improve results on this problem, see e.g. Weston *et al.* (2001).

4.2.3 *Protein family classification*

We then focussed on the problem of classifying protein domains into superfamilies in the Structural Classification of Proteins (SCOP) database version 1.53 (Murzin *et al.* (1995)). We followed the same problem setting as Liao and Noble (2002): sequences were selected using the Astral database (astral.stanford.edu cite), removing similar sequences using an E-value threshold of 10^{-25}. This procedure resulted in 4352 distinct sequences, grouped into families and superfamilies. For each family, the protein domains within the family are considered positive test examples, and the protein domains outside the family but within the same superfamily are taken as positive training examples. The data set yields 54 families containing at least 10 family members (positive training examples). Negative examples are taken from outside of the positive sequence's fold, and are randomly split into train and test sets in the same ratio as the positive examples. Details about the various families are listed in Table 6, and the complete data set is available at www.cs.columbia.edu/compbio/svm-pairwise. Note that this experimental setup is similar to that used by Jaakkola *et al.* (2000), except the positive training sets do not include additional protein sequences extracted from a large, unlabeled database, which amounts to a kind of "transduction" (Vapnik (1998)) algorithm. We believe that it is this transduction step which may be responsible for much of the success of using the methods described by Jaakkola *et al.* (2000). However, to make a fair comparison of

Table 5. Results on the Lymphoma classification problem of using a linear kernel (top row) and an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows show the results of using the subpolynomial kernel to deal with the large diagonal.

kernel method		balanced loss
original k, $k(x,y) = \langle x, y \rangle$		0.043 ± 0.008
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 4$	0.052 ± 0.008
$k_{emp}(x,y) = \operatorname{sgn}\langle x, y \rangle * \mid \langle x, y \rangle \mid^p$	$p = 1$	0.037 ± 0.007
	$p = 0.9$	0.021 ± 0.005
	$p = 0.8$	0.016 ± 0.005
	$p = 0.7$	$\mathbf{0.015 \pm 0.005}$
	$p = 0.6$	0.022 ± 0.006
	$p = 0.5$	0.022 ± 0.006
	$p = 0.4$	0.042 ± 0.007
	$p = 0.3$	0.046 ± 0.008
	$p = 0.2$	0.083 ± 0.009
	$p = 0.1$	0.106 ± 0.009

kernel methods we do not include this step which could potentially be included in any of the methods. Studying the importance of transduction remains a subject of further research.

An SVM requires fixed length vectors. Proteins, of course, are variable-length sequences of amino acids and hence cannot be directly used in an SVM. To solve this task we used a sequence kernel, called the spectrum kernel, which maps strings into a space of features which correspond to every possible k-mer (sequence of k letters) with at most m mismatches, weighted by prior probabilities (Leslie *et al.* (2002)). In this experiment we chose $k = 3$ and $m = 0$. This kernel is then normalized so that each vector has length 1 in the feature space; i.e.,

$$k(x, x') = \frac{\langle x, x' \rangle}{\sqrt{\langle x, x \rangle \langle x', x' \rangle}}. \tag{4.2}$$

An asymmetric soft margin is implemented by adding to the diagonal of the kernel matrix a value $0.02 * \rho$, where ρ is the fraction of training set sequences that have the same label as the current sequence (see Cortes and Vapnik (1995); Brown *et al.* (2000) for details). For comparison, the same SVM parameters are used to train an SVM using the Fisher kernel (Jaakkola and Haussler (1999); Jaakkola *et al.* (2000), see also Tsuda *et al.* (2002)), another possible kernel choice. The Fisher kernel is currently considered one of the most powerful homology detection methods. This method combines a generative, profile hidden Markov model (HMM) and uses it to generate a kernel for training an SVM. A protein's vector representation induced by the kernel is its gradient with respect to the profile hidden Markov model, the parameters of which are found by expectation-maximization.

For each method, the output of the SVM is a discriminant score that is used to rank the members of the test set. Each of the above methods produces as output a ranking of the test set sequences. To measure the quality of this ranking, we use two different scores: receiver operating characteristic (ROC) scores and the median rate of

Table 6. SCOP families included in the experiments. For each family, the numbers of sequences in the positive and negative training and test sets are listed.

	Positive set		Negative set			Positive set		Negative set	
ID	Train	Test	Train	Test	ID	Train	Test	Train	Test
1.27.1.1	12	6	2890	1444	2.9.1.4	21	10	2928	1393
1.27.1.2	10	8	2408	1926	3.1.8.1	19	8	3002	1263
1.36.1.2	29	7	3477	839	3.1.8.3	17	10	2686	1579
1.36.1.5	10	26	1199	3117	3.2.1.2	37	16	3002	1297
1.4.1.1	26	23	2256	1994	3.2.1.3	44	9	3569	730
1.4.1.2	41	8	3557	693	3.2.1.4	46	7	3732	567
1.4.1.3	40	9	3470	780	3.2.1.5	46	7	3732	567
1.41.1.2	36	6	3692	615	3.2.1.6	48	5	3894	405
1.41.1.5	17	25	1744	2563	3.2.1.7	48	5	3894	405
1.45.1.2	33	6	3650	663	3.3.1.2	22	7	3280	1043
2.1.1.1	90	31	3102	1068	3.3.1.5	13	16	1938	2385
2.1.1.2	99	22	3412	758	3.32.1.1	42	9	3542	759
2.1.1.3	113	8	3895	275	3.32.1.11	46	5	3880	421
2.1.1.4	88	33	3033	1137	3.32.1.13	43	8	3627	674
2.1.1.5	94	27	3240	930	3.32.1.8	40	11	3374	927
2.28.1.1	18	44	1246	3044	3.42.1.1	29	10	3208	1105
2.28.1.3	56	6	3875	415	3.42.1.5	26	13	2876	1437
2.38.4.1	30	5	3682	613	3.42.1.8	34	5	3761	552
2.38.4.3	24	11	2946	1349	7.3.10.1	11	95	423	3653
2.38.4.5	26	9	3191	1104	7.3.5.2	12	9	2330	1746
2.44.1.2	11	140	307	3894	7.3.6.1	33	9	3203	873
2.5.1.1	13	11	2345	1983	7.3.6.2	16	26	1553	2523
2.5.1.3	14	10	2525	1803	7.3.6.4	37	5	3591	485
2.52.1.2	12	5	3060	1275	7.39.1.2	20	7	3204	1121
2.56.1.2	11	8	2509	1824	7.39.1.3	13	14	2083	2242
2.9.1.2	17	14	2370	1951	7.41.5.1	10	9	2241	2016
2.9.1.3	26	5	3625	696	7.41.5.2	10	9	2241	2016

false positives (RFP). The ROC score is the normalized area under a curve that plots true positives as a function of false positives for varying classification thresholds. A perfect classifier that puts all the positives at the top of the ranked list will receive an ROC score of 1, and for these data, a random classifier will receive an ROC score very close to 0. The median RFP score is the fraction of negative test sequences that score as high or better than the median-scoring positive sequence. RFP scores were used by Jaakkola *et al.* (2000) in evaluating the Fisher-SVM method.

We show a family-by-family comparison of the subpolynomial spectrum kernel with the normal spectrum kernel and the Fisher kernel in Fig. 2. The coordinates of each point in the plot are the ROC scores for one SCOP family. The subpolynomial kernel uses the parameter $p = 0.2$. Although the subpolynomial method does not improve performance on every single family over the other two methods, there are only a small number of cases where there is a loss in performance.

Table 7. Results of using the spectrum kernel with $k = 3$, $m = 0$ on the SCOP dataset (top row) and using a further nonlinear map via an RBF kernel (second row). For the RBF kernel the optimal σ is shown, i.e. the one which minimizes the test error. The remaining rows (apart from the last one) show the results of using the subpolynomial kernel to deal with the large diagonal. The last row, for comparison, shows the performance of an SVM using the Fisher kernel.

kernel method		RFP	ROC
original k, $k(\Phi(x), \Phi(y)) = \langle x, y \rangle$		0.1978	0.7516
optimal RBF, $k(x,y) = \exp\left(-\frac{\|\Phi(x)-\Phi(y)\|}{2\sigma^2}\right)$	$\sigma = 0.25$	**0.1287**	**0.8469**
$k_{emp}(x,y) = \langle \Phi(x), \Phi(y) \rangle^p$	$p = 0.5$	0.1697	0.7967
	$p = 0.4$	0.1569	0.8072
	$p = 0.3$	0.1474	0.8183
	$p = 0.2$	0.1357	0.8251
	$p = 0.1$	0.1431	0.8213
	$p = 0.05$	0.1489	0.8156
SVM-FISHER		0.2946	0.6762

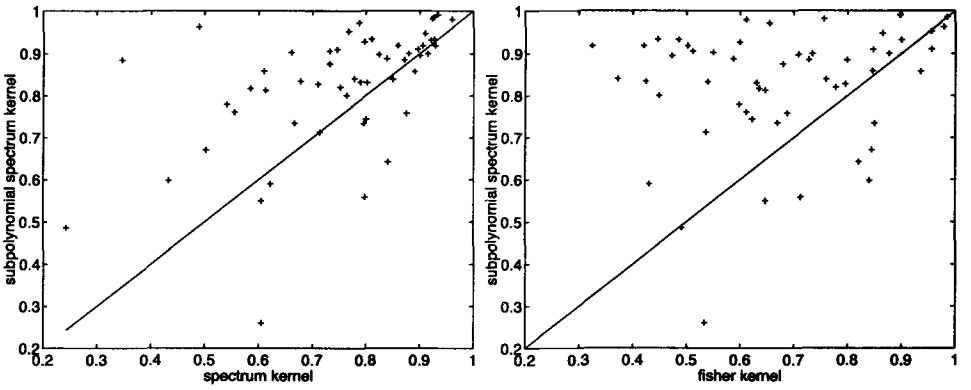

Fig. 2. Family-by-family comparison of the subpolynomial spectrum kernel with the normal spectrum kernel (left), and the Fisher kernel (right). The coordinates of each point in the plot are the ROC scores for one SCOP family. The spectrum kernel uses $k = 3$ and $m = 0$, and the subpolynomial kernel uses $p = 0.2$. Points above the diagonal indicate problems where the subpolynomial kernel performs better than the other methods.

The results of using the spectrum kernel, an RBF kernel, and the subpolynomial kernel applied to the spectrum kernel and the fisher kernel are given in Table 7 and Fig. 2. The mean ROC and RFP scores are superior for the subpolynomial kernel compared to the linear one, but the RBF kernel with the optimal choice of σ performs best overall.

We also compared these methods to using KPCA to extract features from the spectrum kernel as a pre-processing step (results not shown). This only deteriorated the results: the fewer the features extracted, the worse the results. Note that other kinds of explicit feature selection cannot be used in this problem, unless it is possible to integrate the feature selection method into the construction of the spectrum kernel, as the features are never explicitely represented. Note though that the improvements are not

as large as reported in the other experiments (for example, the toy string kernel experiment of Section 4.1.1). We believe this is because this application does not suffer from the large diagonal problem as much as the other problems. This might explain why the RBF kernel outperforms the subpolynomial one. Even without using the subpolynomial method, the spectrum kernel is already superior to the Fisher kernel method. Finally, note that while these results do not represent the record results on this dataset: in (Liao and Noble (2002)), a different kernel (Smith-Waterman pairwise scores) is shown to provide further improvements (mean RFP: 0.09, mean ROC: 0.89). The Smith-Waterman score technique is closely related to the empirical kernel map, where the (non-positive definite) "kernel" is the Smith-Waterman algorithm plus p-value computation. However, this method is much slower to compute. It is also possible to choose other parameters of the spectrum kernel to improve its results. Future work will continue to investigate these kernels.

5. Conclusion

It is a difficult problem to construct useful similarity measures for non-vectorial data types. Not only do the similarity measures have to be positive definite to be useable in an SVM (or, more generally, conditionally positive definite, see e.g. Schölkopf and Smola (2002)), but, as we have explained in the present paper, they should also lead to Gram matrices whose diagonal values are not overly large. It can be difficult to satisfy both needs simultaneously, a prominent example being the much celebrated (but so far not too much used) string kernel. However, the problem is not limited to sophisticated kernels. It is common to all situations where the data are represented as sparse vectors and then processed using an algorithm which is based on dot products.

We have provided a method to deal with this problem. The method's upside is that it can improve performance on kernels that naturally generate matrices with large diagonals (that give almost orthogonal patterns). Its main downside so far is that the precise role and the choice of the function we apply to reduce the dynamic range has yet to be understood, and that its main application, that of string kernels and other kernels for structured objects, have yet to be demonstrated in a real application which truly exhibits the large diagonal effect.

Acknowledgements

We would like to thank Olivier Chapelle, André Elisseeff, and the reviewers for providing some of the data used and for very helpful discussions. We moreover thank Chris Watkins for drawing our attention to the problem of large diagonals.

References

Alizadeh, A. A. *et al.* (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling, *Nature*, **403**, 503–511 (Data available from http://llmpp.nih.gov/lymphoma).

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. and Levine, A. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon cancer tissues probed by oligonucleotide arrays, *Cell Biology*, **96**, 6745–6750.

Berg, C., Christensen, J. P. R. and Ressel, P. (1984). *Harmonic Analysis on Semigroups*, Springer, New York.

Boser, B. E., Guyon, I. M. and Vapnik, V. (1992). A training algorithm for optimal margin classifiers (ed. D. Haussler), *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 144–152, ACM Press, Pittsburgh, Pensylvania.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Furey, T. S., Ares, M. and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data using support vector machines, *Proc. Nat. Acad. Sci. U.S.A.*, **97**(1), 262–267.

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

Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines, *Machine Learning*, **46**, 389–422.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized Additive Models*, Monographs on Statistics and Applied Probability, Vol. 43, Chapman & Hall, London.

Haussler, D. (1999). Convolutional kernels on discrete structures, Tech. Report, UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz.

Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers (eds. M. S. Kearns, S. A. Solla and D. A. Cohn), *Advances in Neural Information Processing Systems 11*, MIT Press, Cambridge, Massachusetts.

Jaakkola, T. S., Diekhans, M. and Haussler, D. (2000). A discriminative framework for detecting remote protein homologies, *Journal of Computational Biology*, **7**, 95–114.

Leslie, C., Eskin, E. and Noble, W. S. (2002). The spectrum kernel: A string kernel for SVM protein classification, *Proceedings of the Pacific Symposium on Biocomputing*, 564–575.

Liao, L. and Noble, W. S. (2002). Combining pairwise sequence similarity and support vector machines for remote protein homology detection, *Proceedings of the Sixth International Conference on Computational Molecular Biology*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C. (2002). Text classification using string kernels, *Journal of Machine Learning Research*, **2**, 419–444.

Murzin, A. G., Brenner, S. E., Hubbard, T. and Chothia, C. (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures, *Journal of Molecular Biology*, **247**, 536–540.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*, MIT Press, Cambridge, Massachusetts.

Schölkopf, B., Weston, J., Eskin, E., Leslie, C. and Noble, W. S. (2002). A kernel approach for learning from almost orthogonal patterns, *Proceedings ECML'2002*, *Helsinki* (to appear).

Tsuda, K. (1999). Support vector classifier with asymmetric kernel function (ed. M. Verleysen), *Proceedings ESANN*, 183–188, D Facto, Brussels.

Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S. and Müller, K. (2002). A new discriminative kernel from probabilistic models (eds. T. Dietterich, S. Becker and Z. Ghahramani), *Advances in Neural Information Processing Systems*, **14**, MIT Press, Cambridge, Massachusetts.

Vapnik, V. (1979). *Estimation of Dependences Based on Empirical Data*, Nauka, Moscow (in Russian) (English translation: Springer Verlag, New York, 1982).

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley, New York.

Watkins, C. (2000). Dynamic alignment kernels (eds. A. J. Smola, P. L. Bartlett, B. Schölkopf and D. Schuurmans), *Advances in Large Margin Classifiers*, 39–50, MIT Press, Cambridge, Massachusetts.

Weston, J., Elisseeff, A. and Schölkopf, B. (2001). Use of the ℓ_0-norm with linear models and kernel methods, Tech. Report, Biowulf Technologies, New York.

Weston, J., Pérez-Cruz, F., Bousquet, O., Chapelle, O., Elisseeff, A. and Schölkopf, B. (2002). Feature selection and transduction for prediction of molecular bioactivity for drug design, http://www.conclu.de/~jason/kdd/kdd.html