

---

# Latent Semantic Kernels

---

Nello Cristianini, John Shawe-Taylor, Huma Lodhi

{NELLO, JOHN, HUMA}@DCS.RHBN.AC.UK

Department of Computer Science,  
Royal Holloway, University of London,  
Egham, Surrey TW20 0EX, UK

## Abstract

Kernel methods like Support Vector Machines have successfully been used for text categorization. A standard choice of kernel function has been the inner product between the vector-space representation of two documents, in analogy with classical information retrieval (IR) approaches.

Latent Semantic Indexing (LSI) has been successfully used for IR purposes, as a technique for capturing semantic relations between terms and inserting them into the similarity measure between two documents. One of its main drawbacks, in IR, is its computational cost.

In this paper we describe how the LSI approach can be implemented in a kernel-defined feature space.

We provide experimental results demonstrating that the approach can significantly improve performance, and that it does not impair it.

## 1. Introduction

Kernel-based learning methods (KMs) are a state-of-the-art class of learning algorithms, whose best known example is Support Vector Machines (SVMs) (Cristianini & Shawe-Taylor, 2000). In this approach, data items are mapped into high-dimensional spaces, where information about their mutual positions (inner products) is used for constructing classification, regression or clustering rules. They are modular systems, formed by a general purpose learning module (eg classification or clustering) and by a data-specific element, called *kernel*, that acts as an interface between the data and the learning machine by providing the mapping into the feature space.

Kernel-based algorithms exploit the information encoded in the inner-product between all pairs of data

items. Somewhat surprisingly, this information is sufficient to run many standard machine learning algorithms, from the Perceptron Convergence algorithm to PCA, from Ridge Regression to nearest neighbour. The advantage of adopting this alternative representation is that often there is an efficient method to compute inner products between very complex, in some cases even infinite dimensional, vectors. Since the explicit representation of feature vectors corresponding to data items is not necessary, KMs have the advantage of accessing feature spaces that would otherwise be either too expensive or too complicated to represent. Strong model selection techniques based on Statistical Learning Theory (Vapnik, 1998) have been developed for such systems in order to avoid overfitting in high dimensional spaces.

It is not surprising that one of the areas where such systems work most naturally is text categorization, where the standard representation of documents is as very high-dimensional vectors, and where standard retrieval techniques are based precisely on the inner-products between vectors. The combination of these two methods has been pioneered by Joachims (Joachims, 1998), and successively explored by several others (Dumais et al., 1997; Leopold & Kindermann, 2001). This approach to documents representation is known as the 'bag of words', and is based on mapping documents to large vectors indicating which words occur in the text. The vectors have as many dimensions as terms in the corpus (usually several thousands), and the corresponding entries are zero if a term does not occur in the document at hand, and positive otherwise. Two documents are hence considered similar if they use (approximately) the same terms. Despite the high dimensionality of such spaces (much higher than the training set size), Support Vector Machines have been shown to perform very well (Joachims, 1998). This paper investigates one possible avenue for extending Joachim's work, by incorporating more information in the kernel.

When used in Information retrieval (IR) this representation is known to suffer from some drawbacks, in particular the fact that semantic relations between terms

are not taken into account. Documents that talk about related topics using different terms are mapped to very distant regions of the feature space. A map that captures some semantic information would be useful, particularly if it could be achieved with a “semantic kernel”, that computes the similarity between documents by also considering relations between different terms.

Using a kernel that somehow takes this fact into consideration would enable the system to extract much more information from documents. One possible approach is the one adopted by (Siolas & d’Alché Buc, 2000), who used a semantic network to explicitly compute the similarity level between terms. Such information is encoded in the kernel, and defines a new metric in the feature space, or - equivalently - a further mapping of the documents into another feature space.

In this paper we propose to use a technique known in Information Retrieval as Latent Semantic Indexing (Deerwester et al., 1990). In this approach, the documents are implicitly mapped into a “semantic space”, where documents that do not share any terms can still be close to each other if their terms are semantically related. The semantic similarity between two terms is inferred by an analysis of their co-occurrence patterns: terms that co-occur often in the same documents are considered as related. This statistical co-occurrence information is extracted by means of a Singular Value Decomposition of the term by document matrix, in the way described in Section 3. We show how this step can be performed implicitly in any kernel-induced feature space, and how it amounts to a ‘kernel adaptation’ or ‘semantic kernel learning’ step. Once we have fixed the dimension of the new feature space, its computation is equivalent to solving a convex optimization problem, so it has just one global maximum that can be found efficiently.

We provide experimental results with text and non-text data showing that this technique can deliver significant improvements on some datasets, and certainly never reduces performance. Then we discuss its advantages, limitations, and its relationships with other methods.

## 2. Kernel Methods for Text

Kernel methods are a new approach to solving machine learning problems. By developing algorithms that only make use of inner products between images of different inputs in a feature space, their application becomes possible to very rich feature spaces provided the inner products can be computed. In this way they avoid the need to explicitly compute the feature vector

for a given input. One of the key advantages of this approach is its modularity: the decoupling of algorithm design and statistical analysis from the problem of creating appropriate function/feature spaces for a particular application. Furthermore, also the design of kernels themselves can be performed in a modular fashion: simple rules exist to combine or adapt basic kernels in order to construct more complex ones, in a way that guarantees that the kernel corresponds to an inner product in some space. The main result of this paper can also be regarded as one such kernel adaptation procedure.

Though the idea of using a kernel defined feature space is not new (Aizerman et al., 1964), it is only recently that its full potential has begun to be realised. The first problem to be considered was classification of labelled examples in the so-called Support Vector Machine (Boser et al., 1992; Cristianini & Shawe-Taylor, 2000), with the corresponding statistical learning analysis described in (Shawe-Taylor et al., 1998). However, this turned out to be only the beginning of the development of a portfolio of algorithms for clustering (Schölkopf et al., 1998) using Principal Components Analysis (PCA) in the feature space, regression (Smola & Schölkopf, 1998), novelty detection (Schölkopf et al., 2000), and ordinal learning (Herbrich et al., 2000). At the same time links have been made between this statistical learning approach, the Bayesian approach known as Gaussian Processes (Opper & Winther, 2000), and the more classical Kriegering known as Ridge Regression (Saunders et al., 1998), hence for the first time providing a direct link between these very distinct paradigms.

In view of these developments it is clear that defining an appropriate kernel function allows one to use a range of different algorithms to analyse the data concerned potentially answering many practical prediction problems. For a particular application choosing a kernel corresponds to implicitly choosing a feature space since the kernel function is defined by

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle, \quad (1)$$

for the feature map  $\phi$ . Given a training set  $S = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , the information available to kernel based algorithms is contained entirely in the matrix of inner products

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

known as the Gram or kernel matrix. This matrix represents in a way a sort of ‘bottleneck’ for the information that can be exploited: by operating on the matrix, one can in fact ‘virtually’ recode the data in a more suitable manner, as we will see.

The solutions sought are linear functions in the feature space

$$f(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}),$$

for some weight vector  $\mathbf{w}$ , where  $'$  denotes the transpose of a vector or matrix. The kernel trick can be applied whenever the weight vector can be expressed as a linear combination of the training points,  $\mathbf{w} = \sum_{i=1}^m \alpha_i \phi(\mathbf{x}_i)$ , implying that we can express  $f$  as follows

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

Given an explicit feature map  $\phi$  we can use equation (1) to compute the corresponding kernel. Often, however, methods are sought to provide directly the value of the kernel without explicitly computing  $\phi$ . We will show how many of the standard information retrieval feature spaces give rise to a particularly natural set of kernels.

Perhaps the best known method of this type is referred to as the polynomial kernel. Given a kernel  $k$  the polynomial construction creates a kernel  $\hat{k}$  by applying a polynomial with positive coefficients to  $k$ , for example consider

$$\hat{k}(\mathbf{x}, \mathbf{z}) = (k(\mathbf{x}, \mathbf{z}) + D)^p,$$

for fixed values of  $D$  and integer  $p$ . Suppose the feature space of  $k$  is  $\mathcal{F}$ , then the feature space of  $\hat{k}$  is indexed by  $t$ -tuples of features from  $\mathcal{F}$ , for  $t = 0, 1, \dots, p$ . Hence, through a relatively small additional computational cost (each time an inner product is computed one more addition and exponentiation is required) the algorithms are being applied in a feature space of vastly expanded expressive power. As an even more extreme example consider the Gaussian kernel  $\tilde{k}$  that transforms the kernel  $k$  as follows:

$$\tilde{k}(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{z}, \mathbf{z}) - 2k(\mathbf{x}, \mathbf{z})}{\sigma^2}\right),$$

whose feature space has infinitely many dimensions.

Given a document, it is possible to associate with it a bag of terms (or bag of words) by simply considering the number of occurrences of all the terms it contains. Typically words are “stemmed” meaning that the inflection information contained in the last few letters is removed.

A bag of words has its natural representation as a vector in the following way. The number of dimensions is the same as the number of different terms in the corpus, each entry of the vector is indexed by a specific

term, and the components of the vector are formed by integer numbers representing the frequency of the term in the given document. Typically such a vector is then mapped into some other space, where the word frequency information is merged with other information (e.g. word importance, where uninformative words are given low or no weight).

In this way a document is represented by a (column) vector  $d$  in which each entry records how many times a particular word stem is used in the document. Typically  $d$  can have tens of thousands of entries, often more than the number of documents. Furthermore, for a particular document the representation is typically extremely sparse, having only relatively few non-zero entries.

In the basic vector-space model (BVSM), a document is represented by a vertical vector  $d$  indexed by all the elements of the dictionary, and a corpus by a matrix  $D$ , whose columns are indexed by the documents and whose rows are indexed by the terms,  $D = [d_1, \dots, d_m]$ . We also call the data matrix  $D$  the “term by document” matrix. We define the “document by document” matrix to be  $G = D'D$ ; the “term by term” matrix to be  $T = DD'$ .

If we consider the feature space defined by the basic vector-space model, the corresponding kernel is given by the inner product between the feature vectors

$$K(d_1, d_2) = \langle d_1, d_2 \rangle = d_1' d_2.$$

In this case the Gram matrix is just the document by document matrix. More generally, we can consider transformations of the document vectors by some mapping  $\phi$ . The simplest case involves linear transformations of the type  $\phi(d) = Pd$ , where  $P$  is any appropriately shaped matrix. In this case the kernels have the form

$$K(d_1, d_2) = d_1' P' P d_2.$$

We will call all such representations Vector Space Models (VSMs). The Gram matrix is in this case given by  $D'P'PD$  that is by definition symmetric and positive definite. The class of models obtained by varying the matrix  $P$  is a very natural one, corresponding as it does to different linear mappings of the standard vector space model, hence giving different scalings and projections. Note that Jiang and Littman (Jiang & Littman, 2000) use this framework to present a collection of different methods, although without viewing them as kernels.

### 3. Latent Semantic Kernels

Latent Semantic Indexing (LSI) (Deerwester et al., 1990) is a technique to incorporate semantic information in the measure of similarity between two documents. We will use it to construct kernel functions. Conceptually, LSI measures semantic information through co-occurrence analysis in the corpus. The technique used to extract the information relies on a Singular Value Decomposition (SVD) of the term by document matrix. The document feature vectors are projected into the subspace spanned by the first  $k$  singular vectors of the feature space. Hence, the dimension of the feature space is reduced to  $k$  and we can control this dimension by varying  $k$ . We can define a kernel for this feature space through a particular choice of the matrix  $P$ , and we will see that  $P$  can be computed directly from the original kernel matrix without direct computation of the SVD in the feature space.

In order to derive a suitable matrix  $P$  first consider the term-document matrix  $D$  and its SVD decomposition

$$D = U\Sigma V'$$

where  $\Sigma$  is a diagonal matrix with the same dimensions as  $D$ , and  $U$  and  $V$  are orthogonal (ie  $U'U = I$ ). The columns of  $U$  are the singular vectors of the feature space in order of decreasing singular value. Hence, the projection operator onto the first  $k$  dimensions is given by  $P = U'_k = I_k U'$ , where  $I_k$  is the identity matrix with only the first  $k$  diagonal elements nonzero and  $U_k$  the matrix consisting of the first  $k$  columns of  $U$ . The new kernel can now be expressed as

$$\begin{aligned} k(d_1, d_2) &= (I_k U' d_1)' (I_k U' d_2) \\ &= d'_1 U I_k U' d_2 = d'_1 P' P d_2. \end{aligned}$$

The motivation for this particular mapping is that it identifies highly correlated dimensions: i.e. terms that co-occur very often in the same documents of the corpus are merged into a single dimension of the new space. This creates a new similarity metric based on context information. In the case of LSI it is also possible to isometrically re-embed the subspace back into the original feature space by defining  $\hat{P}$  as the square symmetric  $(U_k U')' = (U I_k U')$ . This gives rise to the same kernel, since

$$\begin{aligned} k(d_1, d_2) &= (U U'_k d_1)' (U U'_k d_2) \\ &= d'_1 U_k U' U U'_k d_2 = d'_1 P' P d_2. \end{aligned}$$

We can then view  $\hat{P}$  as a term-term similarity matrix making LSI a special case of the semantic smoothing described in Solias and d'Alché-Buc (Solias & d'Alché Buc, 2000). While they need to explicitly work out

all the entries of the term-by-term similarity matrix with the help of a semantic network, however, we can infer the semantic similarities directly from the corpus, using co-occurrence analysis.

What is more interesting for kernel methods is that the same mapping, instead of acting on term-term matrices, can be obtained implicitly by working with the smaller document-document Gram matrix. The original term by document matrix  $D$  gives rise to the kernel matrix

$$K = D' D,$$

since the feature vector for document  $j$  is the  $j$ -th column of  $D$ . The SVD decomposition is related to the eigenvalue decomposition of  $K$  as follows

$$K = D' D = V \Sigma U' U \Sigma V' = V \Sigma^2 V' = V \Lambda V'$$

so that the  $i$ -th column of  $V$  is the eigenvector of  $K$ , with corresponding eigenvalue  $\Lambda_{ii} = \lambda_i = \sigma_i^2$ . The feature space created by choosing the first  $k$  singular values in the LSI approach corresponds to mapping a feature vector  $d$  to the vector  $U I_k U' d$  and gives rise to the following kernel matrix

$$\begin{aligned} \hat{K} &= D' U I_k U' U I_k U' D \\ &= V \Sigma U' U I_k U' U \Sigma V' \\ &= V \Sigma I_k \Sigma V' = V \Lambda_k V' \end{aligned}$$

where  $\Lambda_k$  is the matrix  $\Lambda$  with diagonal entries beyond the  $k$ -th set to zero. Hence, the new kernel matrix can be obtained directly from  $K$  by applying an eigenvalue decomposition of  $K$  and remultiplying the component matrices having set all but the first  $k$  eigenvalues to zero. Hence, we can obtain the kernel corresponding to the LSI feature space without actually ever computing the features. The relations of this computation to kernel PCA (Schölkopf et al., 1999) are immediate. By a similar analysis it is possible to verify that we can also evaluate the new kernel on novel inputs again without reference to the explicit feature space. In order to evaluate the learned functions on novel examples, we must show how to evaluate the new kernel  $\hat{k}$  between a new input  $d$  and a training example,  $\hat{k}(d_i, d)$ . The function we wish to evaluate will have the form

$$\begin{aligned} f(d) &= \sum_{i=1}^m \alpha_i \hat{k}(d_i, d) = \sum_{i=1}^m \alpha_i (U U'_k d_i)' (U U'_k d) \\ &= \sum_{i=1}^m \alpha_i ((U U'_k D)' (U U'_k d))_i \\ &= \sum_{i=1}^m \alpha_i (V \Sigma U' U_k U'_k d)_i \\ &= \sum_{i=1}^m \alpha_i (V \Sigma_k U'_k d)_i = \alpha' V \Sigma_k U'_k d \end{aligned}$$

The expression still, however, involves the feature vector  $d$  which we would like to avoid evaluating explicitly. Consider the vector

$$t = D'd = V\Sigma U'd$$

of inner products between the new feature vector and the training examples in the original space. These inner products can be evaluated using the original kernel. But now we have

$$I_k V't = I_k \Sigma U'd = \Sigma_k U'd = \Sigma_k U'_k d,$$

showing that we can evaluate  $f(d)$  as follows

$$f(d) = \alpha' V \Sigma_k U'_k d = \alpha' V I_k V't.$$

Hence to evaluate  $f$  on a new example, we first create a vector of the inner products in the original feature space and then take its inner product with the precomputed row vector  $\alpha' V I_k V'$ . None of this computation involves working directly in the feature space.

The combination of the LSK technique with the polynomial or Gaussian construction opens up the possibility of performing LSI in very high dimensional feature spaces, for example indexed by tuples of terms. Experiments applying this approach are reported in the experimental section of this paper. If we think of the polynomial mapping as taking conjunctions of terms, we can view the LSK step as a soft disjunction, since the projection links several different conjunctions into a single concept. Hence, the combination of the polynomial mapping followed by an LSK step produces a function with a form reminiscent of a disjunctive normal form.

Alternatively one could perform the LSK step before the polynomial mapping (by just applying the polynomial mapping to the entries of the Gram matrix obtained after the LSK step), obtaining a space indexed by tuples of concepts. Here the function obtained will be reminiscent of a conjunctive normal form. We applied this approach to the Ionosphere data but obtained no improvement in performance. We conjecture that the results obtained will depend strongly on the fit of the style of function with the particular data.

The main drawback of all such approaches is the computational complexity of performing an eigenvalue decomposition on the kernel matrix. Although the matrix is smaller than the term by document matrix it is usually no longer sparse. This makes it difficult to process training sets much larger than a few thousand examples. We will present in the next section techniques that get round this problem by evaluating an approximation of the LSK approach.

## 4. Algorithmic Techniques

All the experiments were performed using the eigenvalue decomposition routine provided with Numerical Recipes in C (Press, 1992).

The complete eigen-decomposition of the Kernel matrix is an expensive step, and where possible one should try to avoid it when working with real world data. More efficient methods can be developed to obtain or approximate the LSK solution.

We can view the LSK technique as one method of obtaining a low rank approximation of the kernel matrix. Indeed the projection on to the first  $k$  eigenvalues is the rank  $k$  approximation which minimises the norm of the resulting error matrix. But projection onto the eigensubspaces is just one method of obtaining a low-rank approximation.

We have also developed an approximation strategy, based on the Gram-Schmidt decomposition. A similar approach to unsupervised learning is described by Smola et al. (Smola et al., 1999).

The projection is built up as the span of a subset of (the projections of) a set of  $k$  training examples. These are selected by performing a Gram-Schmidt orthogonalisation of the training vectors in the feature space. Hence, once a vector is selected the remaining training points are transformed to become orthogonal to it. The next vector selected is the one with the largest residual norm. The whole transformation is performed in the feature space using the kernel mapping to represent the vectors obtained. We refer to this method as the G-S algorithm<sup>1</sup>. As with the LSK method it is parametrised by the number of dimensions selected.

### 4.1 Implicit Dimensionality Reduction

An interesting solution to the problem of approximating the Latent Semantic solution is possible in the case in which we are not directly interested in the low-rank matrix, unlike in the information retrieval case, but we only plan to use it as a kernel in conjunction with an optimization problem of the type:

$$\text{minimize } W(\alpha) = c + q\alpha + \frac{1}{2}\alpha^T H\alpha$$

where  $H$  is the Hessian, obtained by pre- and post-multiplying the Gram matrix by the diagonal matrix containing the  $\{+1, -1\}$  labels,

$$H = YKY, \quad \text{where } Y = \text{diag}(y_i).$$

<sup>1</sup>The full description of this algorithm is omitted due to lack of space

Note that  $H$  and  $K$  have the same eigenvalues since if  $Kx = \lambda x$ , then  $HYx = \lambda Yx$ . It is possible to easily (and cheaply) modify the Gram matrix so as to obtain nearly the same solution that one would obtain by using a (much more expensive) low rank approximation.

The minimum of this error function occurs at the point  $\alpha^*$  which satisfies  $q + H\alpha^* = 0$ . If the matrix  $H$  is replaced by  $H + \lambda I$  then the minimum moves to a new point  $\tilde{\alpha}$  which satisfies  $q + H\tilde{\alpha} + \lambda\tilde{\alpha} = 0$ . Let us consider the expansion of  $H$  in its eigenbasis:  $Hu_j = \mu_j u_j$  and the expansions of  $\alpha^*$  and  $\tilde{\alpha}$  in the same basis:

$$\alpha^* = \sum \alpha_i^* u_i \quad \tilde{\alpha} = \sum \tilde{\alpha}_i u_i.$$

Substituting into the above formulae and equating coefficients of the  $i$ -th eigenvalue gives

$$\alpha_i^* \mu_i = \tilde{\alpha}_i (\mu_i + \lambda) \quad \text{implying that} \quad \tilde{\alpha}_i = \frac{\mu_i}{\mu_i + \lambda} \alpha_i^*.$$

The fraction in the above equation is a squashing function, approaching zero for values of  $\mu \ll \lambda$  and approaching 1 for  $\mu \gg \lambda$ . In the first case  $\tilde{\alpha}_j \approx 0$ , while in the second case  $\tilde{\alpha}_j \approx \alpha_j^*$ . The overall effect of this map, if the parameter  $\lambda$  is chosen carefully in a region of the spectrum where the eigenvalues decrease rapidly, is to effectively project the solution onto the space spanned by the eigenvectors of the larger eigenvalues.

From an algorithmic point of view this is much more efficient than explicitly performing the low-rank approximation by computing the eigenvectors.

This derivation does not only provide a cheap approximation algorithm for the latent semantic kernel. It also highlights an interesting connection between this algorithm and the soft margin algorithm for noise tolerance, that also can be obtained by adding a diagonal to the kernel matrix (Shawe-Taylor & Cristianini, 2000). But note that there are several approximations in this view since for example the SVM solution is a constrained optimisation, where the  $\alpha_i$ 's are held within the interval  $[0, C]$ . In this case the effect may be very different if the support vectors are nearly orthogonal to the eigenvectors corresponding to large eigenvalues. The fact that the procedure is distinct from a standard soft margin approach is borne out in the experiments that are described in the next section.

## 5. Experimental Results

We tried the system both on text and on non-text data, in order to demonstrate the general applicability of the method, and to test its effectiveness under different

conditions. The results were generally positive, but in some cases the improvements are not significant or not worth the additional computation. In other cases there is a significant advantage in using the Latent Semantic kernels, and certainly their use never hurts performance.

As a first example, we present a non-text Ionosphere data set from the UCI repository. We considered 100 random splits in training and testing data (90% and 10% of the total set of 351 points) and plotted the performance of LSK after a kernel mapping by means of polynomial kernels of degree 2, 3 and 4. In all cases, the test error was greatly reduced when the dimension of the feature space was reduced. When the number of dimensions was too small, the performance rapidly deteriorated. We applied a 1-norm soft margin SVM (box constraint) in all cases with the soft margin parameter tuned on one split for the full feature space and kept fixed for the reduced feature spaces and other splits on the full space. The results for Ionosphere are summarized in Figure 1 and in Table 1 (mean and standard deviation of the test error for the best dimension for each kernel).

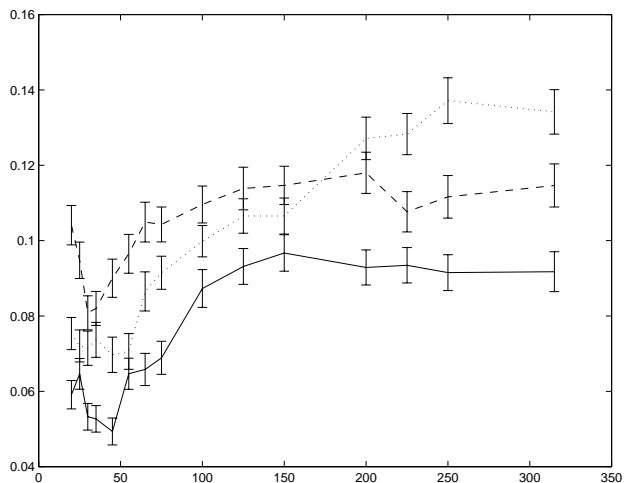
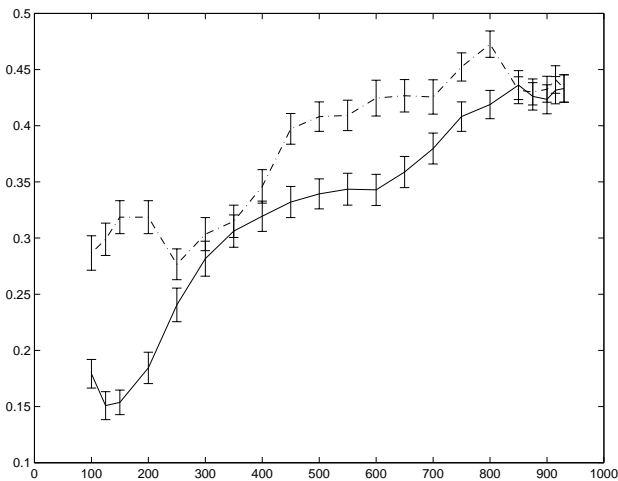


Figure 1. Generalization error for polynomial kernels of degrees 2,3, 4 on Ionosphere data (averaged over 100 random splits) as a function of the dimension of the feature space.

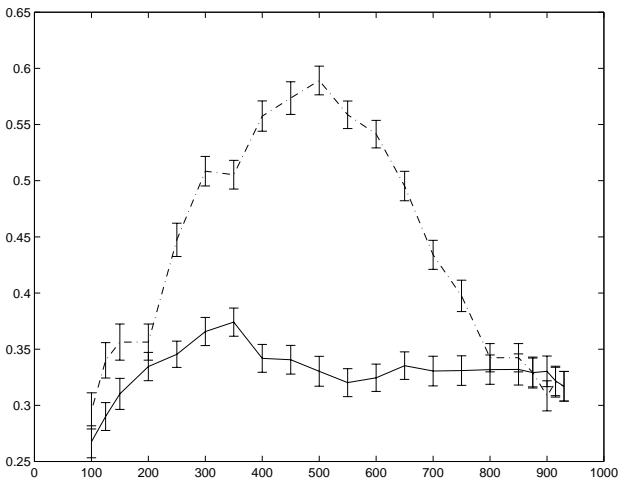
Table 1. Ionosphere Results for the dimension giving the best performance (LSK) compared with using the full feature space (SVM)

DEGREE	LSK		SVM	
	MEAN	SD	MEAN	SD
1	0.155	0.057	0.155	0.057
2	0.049	0.036	0.092	0.053
3	0.081	0.045	0.115	0.057
4	0.070	0.047	0.134	0.059

As examples of text datasets (or corpora) we considered Medline and Reuters. For Medline we considered the most frequent category (Query23) and performed 100 random splits so as to always have 24 positives in the training set (and 15 in the test set) for Query 23, and 22 positives (15 in the test set) for Query 20. We also tested the Gram-Schmidt (GSK) algorithm. For the LSK and GSK algorithms Table 2 and Figure 2 show the mean and the standard deviation of the F1 performance measure given by  $2pr/(p+r)$ , where  $p$  is the precision and  $r$  the recall. Note that the optimal value of F1 is 1. Finally, we report in Table 3 the



(a)



(b)

Figure 2. F1 Values (higher values imply better quality) for different projection dimensions for the Medline data: (a) degree 1 kernel for LSK (unbroken line) and GSK methods (dotted line) for query 20, (b) same for query 23

results for 5 categories of Reuters, where LSK did not yield significant advantages. We report the average performance of SVMs and LSKs over the 5 categories

Table 2. Medline Results for the dimension giving the best performance (LSK) compared with using the full feature space (SVM)

DEGREE	LSK		SVM	
	MEAN	SD	MEAN	SD
1	0.682	0.092	0.622	0.109
2	0.397	0.135	0.361	0.141

(top 5 Categories (Earn, Acq, Crude, Grain, Money)) and also averaged over 10 random splits, using linear kernels. Table 3 gives the mean and standard deviation of the F1 performance measure (averaged as described above).

Table 3. Reuters Results for the dimension giving the best performance (LSK) compared with using the full feature space (SVM)

DEGREE	LSK		SVM	
	MEAN	SD	MEAN	SD
1	0.874	0.04	0.867	0.038

## 6. Conclusion

We have introduced and demonstrated a general method for incorporating semantic information in a kernel. Although the experiments we have performed only tested the simplest uses of the method, the results were positive.

The idea of constructing automatically a feature set that takes into consideration the properties of the corpus at hand is a very promising direction in text categorization and in pattern recognition in general. Although relying on assumptions, our method addresses that problem.

Much work remains to be done, both on the algorithmic and on the theoretical side, to completely understand in what situations this method is more likely to deliver good performance.

## Acknowledgements

The authors would like to thank Thorsten Joachims and Chris Watkins for useful discussions. Our work was supported by EPSRC grant number GR/N08575 and by the European Commission through the ES-PRIT Working Group in Neural and Computational Learning, NeuroCOLT2, Nr. 27150.

## References

- Aizerman, M., Braverman, E., & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25, 821–837.
- Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (pp. 144–152). ACM Press.
- Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 39(1–407). 41(6).
- Dumais, S. T., Letsche, T. A., Littman, M. L., & Landauer, T. K. (1997). Automatic cross-language retrieval using latent semantic indexing. *AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*.
- Herbrich, R., Obermayer, K., & Graepel, T. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*. MIT Press.
- Jiang, F., & Littman, M. L. (2000). Approximate dimension equalization in vector-based information retrieval. *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan-Kaufman.
- Joachims, T. (1998). Text categorization with support vector machines. *Proceedings of European Conference on Machine Learning (ECML)*.
- Leopold, E., & Kindermann, J. (2001). Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*. to appear.
- Opper, M., & Winther, O. (2000). Gaussian processes and SVM: Mean field and leave-one-out. *Advances in Large Margin Classifiers*. MIT Press.
- Press, W. H. (1992). *Numerical recipes in c: the art of scientific computing*. Cambridge University Press.
- Saunders, C., Gammernann, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables. *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann.
- Schölkopf, B., Mika, S., Smola, A., Rätsch, G., & Müller, K.-R. (1998). Kernel PCA pattern reconstruction via approximate pre-images. *Proceedings of the 8th International Conference on Artificial Neural Networks* (pp. 147 – 152). Springer Verlag.
- Schölkopf, B., Smola, A. J., & Müller, K. (1999). Kernel principal component analysis. *Advances in Kernel Methods – Support Vector Learning* (pp. 327–352). MIT Press.
- Schölkopf, B., Williamson, R., Smola, A., & Shawe-Taylor, J. (2000). SV estimation of a distribution’s support. *Neural Information Processing Systems*. forthcoming.
- Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., & Anthony, M. (1998). Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44, 1926–1940.
- Shawe-Taylor, J., & Cristianini, N. (2000). Margin distribution and soft margin. *Advances in Large Margin Classifiers* (pp. 349–358). Cambridge, MA: MIT Press.
- Siolas, G., & d’Alché Buc, F. (2000). Support vectors machines based on a semantic kernel for text categorization. *Proceedings of the International Joint Conference on Neural Networks, IJCNN*. Como: IEEE.
- Smola, A., & Schölkopf, B. (1998). A tutorial on support vector regression. *Statistics and Computing*. Invited paper, in press.
- Smola, A. J., Mangasarian, O. L., & Schölkopf, B. (1999). *Sparse kernel feature analysis* (Technical Report 99-04). University of Wisconsin, Data Mining Institute, Madison.
- Vapnik, V. (1998). *Statistical learning theory*. Wiley.