, , 1?? ()

Latent Semantic Kernels

NELLO CRISTIANINI nello@cs.rhul.ac.uk

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

JOHN SHAWE-TAYLOR john@cs.rhul.ac.uk

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

HUMA LODHI huma@cs.rhul.ac.uk

Department of Computer Science,
Royal Holloway, University of London
Egham, Surrey TW20 OEX, 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) [3]. 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 (e.g. classification or clustering) and by a data-specific element, called the *kernel*, that acts as an interface between the data and the learning machine by defining 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 Principal Components Analysis (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 [26] 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 [10], and successively explored by several others [6, 11]. 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 [10]. This paper investigates one possible avenue for extending Joachims' 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 [23], where a semantic network is used 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 (LSI) [4]. 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 of eigenvalue decomposition, so it has just one global maximum that can be found efficiently. Since eigenvalue decomposition can become expensive for very large datasets we develop an approximation technique based on the Gram-Schmidt orthogonalisation procedure. In practice this method can actually perform better than the LSI method.

We provide experimental results with text and non-text data showing that the techniques can deliver significant improvements on some datasets, and certainly never reduce performance. Then we discuss their advantages, limitations, and their 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, 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 feature 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 [1], 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 [2, 3], with the corresponding statistical learning analysis described in [20]. However, this turned out to be only the beginning of the development of a portfolio of algorithms for clustering [17] using Principal Components Analysis (PCA) in the feature space, regression [24], novelty detection [19], and ordinal learning [7]. At the same time links have been made between this statistical learning approach, the Bayesian approach known as Gaussian Processes [13], and the more classical Krieging known as Ridge Regression [16], 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

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

for the feature map ϕ. Given a training set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, 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 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.

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 ϕ 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 ϕ. 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 $t = 0, 1, \ldots, p$. 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.

3. Vector Space Representations

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, $D = [d_1, ..., d_m]$. We also call the data matrix D the "term by document" matrix. We define the "document by document" matrix to be $G = D'D$ and 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

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

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 ϕ. 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 [9] use this framework to present a collection of different methods, although without viewing them as kernels. Throughout the rest of the paper we will use P to refer to the matrix defining the VSM. We will describe a number of different models in each case showing how an appropriate choice of P realises it as VSM.

Basic Vector Space Model

The Basic Vector Space Model (BVSM) was introduced in 1975 by Salton et al. [15] (and used as a kernel by Joachims [10]) and uses the vector representation with no further mapping. In other words the VSM matrix $P = I$ in this case. The performance of retrieval systems based on such a simple representation is surprisingly good. Since the representation of each document as a vector is very sparse, special techniques can be deployed to facilitate the storage and the computation of dot products between such vectors.

A common map P is obtained by considering the importance of each term in a given corpus. The VSM matrix is hence a diagonal, whose entries $P(i,i)$ are the weight of the term i. Several methods have been proposed, and it is known that this has a strong influence on generalization [11]. Often $P(i,i)$ is a function of the inverse document frequency $\mathrm{idf}_i = \frac{m}{d(\mathrm{term})}$, that is the total number of documents in the corpus divided by the number of documents that contain the given term. So if for example a word appears in each document, it would not be regarded as a very informative one. Its distance from the uniform distribution is a good estimation of its importance, but better methods can be obtained by studying the typical term distributions within documents and corpora. The simplest method for doing this is just given by $P(i,i) = \log(\mathrm{idf}_i)$. Other measures can be obtained from information theoretic quantities, or from empirical models of term frequency. Since these measures do not use label information, they could also be estimated from an external, larger unlabelled corpus, that provides the background knowledge to the system.

As described in the previous section as soon as we have defined a kernel we can apply the polynomial or Gaussian construction to increase its expressive power. Joachims [10] and Dumais et al. [5] have applied this technique to the basic vector space model for a classification task with impressive results. In particular, the use of polynomial kernels can be seen as including features for each tuple of words up to the degree of the chosen polynomial.

One of the problems with this representation is that it treats terms as uncorrelated, assigning them orthogonal directions in the feature space. This means that it can only cluster documents that share many terms. But in reality words are correlated, and sometimes even synonymous, so that documents with very few common terms can potentially be on closely related topics. Such similarities cannot be detected by the BVSM. This raises the question of how to incorporate information about semantics into the feature map, so as to link documents that share *related* terms?

One idea would be to perform a kind of document expansion, adding to the expanded version all synonymous (or closely related) words to the existing terms. Another, somehow similar, method would be to replace terms by concepts. This information could potentially be gleaned from external knowledge about correlations, for example from a semantic network. There are, however, other ways to address this problem. It is also possible to use statistical information about term-term cor-

relations derived from the corpus itself, or from an external reference corpus. This approach forms the basis of Latent Semantic Indexing.

In the next subsections we will look at two different methods, in each case showing how they can be implemented directly through the kernel matrix, without the need to work explicitly in the feature space. This will allow them to be combined with other kernel techniques such as the polynomial and Gaussian constructions described above.

Generalised Vector Space Model

An early attempt to overcome the limitations of BVSMs was proposed by Wong et al. [27] under the name of Generalised VSM, or GVSM. A document is characterised by its relation to other documents in the corpus as measured by the BVSM. This method aims at capturing some term-term correlations by looking at co-occurrence information: two terms become semantically related if they co-occur often in the same documents. This has the effect that two documents can be seen as similar even if they do not share any terms. The GVSM technique can provide one such metric, and it is easy to see that it also constitutes a kernel function.

Given the term by document data matrix D, the GVSM kernel is given by

$$K(d_1, d_2) = (D'd_1)(D'd_2) = d_1'DD'd_2.$$

The matrix DD' is the term by term matrix and has a nonzero ij entry if and only if there is a document in the corpus containing both the i-th and the j-th terms. So two terms co-occurring in a document are considered related. The new metric takes this co-occurrence information into account.

The documents are mapped to a feature space indexed by the documents in the corpus, as each document is represented by its relation to the other documents in the corpus. For this reason it is also known as a dual space method [22]. In the common case when there are less documents than terms, the method will act as a bottle-neck mapping forcing a dimensionality reduction. For the GVSM the VSM matrix P has been chosen to be D' the document by term matrix.

Once again the method can be combined with the polynomial and Gaussian kernel construction techniques. For example the degree p polynomial kernel would have features for each $(\leq p)$-tuple of documents with a non-zero feature for a document that shares terms with each document in the tuple. To our knowledge this combination has not previously been considered with either the polynomial or the Gaussian construction.

Semantic Smoothing for Vector Space Models

Perhaps a more natural method of incorporating semantic information is by directly using an external source, like a semantic network. In this section we briefly describe

one such approach. Siolas and d'Alché-Buc [23] used a semantic network (Word-net [12]) as a way to obtain term-similarity information. Such a network encodes for each word of a dictionary its relation with the other words in a hierarchical fashion (e.g. synonym, hypernym, etc). For example both the word 'husband' and 'wife' are special cases of their hypernym 'spouse'. In this way, the distance between two terms in the hierarchical tree provided by Wordnet gives an estimation of their semantic proximity, and can be used to modify the metric of the vector space when the documents are mapped by the bag-of-words approach.

Siolas and d'Alché-Buc [23] have included this knowledge into the kernel by hand-crafting the entries in the square VSM matrix P. The entries $P_{ij} = P_{ji}$ expresses the semantic proximity between the terms i and j. The semantic proximity is defined as the inverse of their topological distance in the graph, that is the length of the shortest path connecting them (but some cases deserve special attention). The modified metric then gives rise to the following kernel

$$k(d_1, d_2) = d_1' P' P d_2 = d_1' P^2 d_2$$

or to the following distance

$$\begin{aligned} \|Pd_1 - Pd_2\|^2 &= \|P(d_1 - d_2)\|^2 \\ &= (d_1 - d_2)' P^2 (d_1 - d_2). \end{aligned}$$

Siolas and d'Alché-Buc used this distance in order to apply the Gaussian kernel construction described above, though a polynomial construction could equally well be applied to the kernel.

Siolas and d'Alché-Buc used a term-term similarity matrix to incorporate semantic information resulting in a square matrix P. It would also be possible to use a concept-term relation matrix in which the rows would be indexed by concepts rather than terms. For example one might consider both 'husband' and 'wife' examples of the concept 'spouse'. The matrix P would in this case no longer be square symmetric. Notice that GVSMs can be regarded as a special case of this, when the concepts correspond to the documents in the corpus, that is a term belongs to the i-th 'concept' if it occurs in document d_i.

4. Latent Semantic Kernels

Latent Semantic Indexing (LSI) [4] 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 VSM 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 Σ 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 [23]. 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 $UI_kU'd$ and gives rise to the following kernel matrix

$$\begin{aligned}\hat{K} &= D'UI_kU'UI_kU'D \\ &= V\Sigma U^{'}UI_kU'U\Sigma V^{'} \\ &= V\Sigma I_k\Sigma V^T = V\Lambda_k V^T\end{aligned}$$

where Λ_k is the matrix Λ 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 [18] 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

$$\begin{aligned}f(d) &= \sum_{i=1}^{m}\alpha_i\hat{k}(d_i,d) = \sum_{i=1}^{m}\alpha_i(UU_k'd_i)'(UU_k'd) \\ &= \sum_{i=1}^{m}\alpha_i\left((UU_k'D)'(UU_k'd)\right)_i \\ &= \sum_{i=1}^{m}\alpha_i\left(V\Sigma U'U_kU_k'd\right)_i \\ &= \sum_{i=1}^{m}\alpha_i\left(V\Sigma_kU_k'd\right)_i = \alpha'V\Sigma_kU_k'd\end{aligned}$$

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

$$I_kV't = I_k\Sigma U'd = \Sigma_kU'd = \Sigma_kU_k'd,$$

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

$$f(d) = \alpha'V\Sigma_kU_k'd = \alpha'VI_kV't.$$

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.

5. Algorithmic Techniques

All the experiments were performed using the eigenvalue decomposition routine provided with Numerical Recipes in C [14].

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 onto 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. [25].

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 GK algorithm. Table 1 gives complete pseudo-code for extracting the features in the kernel defined feature space. As with the LSK method it is parametrised by the number of dimensions T selected.

Table 1. The GSK Algorithm

Given a kernel k, training set $d_1, \ldots, d_m$ and number T:

```
for i = 1 to m do
    norm2[i] = k(d_i, d_i);
for j = 1 to T do
    i_j = argmax_i(norm2[i]);
    index[j] = i_j;
    size[j] = sqrt(norm2[i_j]);
    for i = 1 to m do
        feat[i,j] = ( k(d_i, d_{i_j}) - sum_{t=1}^{j-1} feat[i,t] * feat[i_j,t] ) / size[j];
        norm2[i] = norm2[i] - feat(i,j) * feat(i,j);
    end;
end;
return feat[i,j] as the j-th feature of input i;
```

To classify a new example x:

```
for j = 1 to T do
    newfeat[j] = ( k(d, d_{i_j}) - sum_{t=1}^{j-1} newfeat[t] * feat[i_j,t] ) / size[j];
end;
return newfeat[j] as the j-th feature of the example x;
```

5.1. 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:

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

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

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

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

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

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

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

The fraction in the above equation is a squashing function, approaching zero for values of $\mu_i \ll \lambda$ and approaching 1 for $\mu_i \gg \lambda$. In the first case $\widetilde{\alpha}_i \approx 0$, while in the second case $\widetilde{\alpha}_i \approx \alpha_i^*$. The overall effect of this map, if the parameter λ 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 not only provides a cheap approximation algorithm for the latent semantic kernel. It also highlights an interesting connection between this algorithm and the 2-norm soft margin algorithm for noise tolerance, that also can be obtained by adding a diagonal to the kernel matrix [21]. But note that there are several approximations in this view since for example the SVM solution is a constrained optimisation, where the α_i's are constrained to be positive. 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.

6. Experimental Results

We empirically tested the proposed methods 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 or Gram-Schmidt kernels, and certainly their use never hurts performance.

6.1. Experiments on Text Data

This section describes a series of systematic experiments performed on text data. We selected two text collections, namely Reuters and Medline that are described below.

Datasets

Reuters21578 We conducted the experiments on a set of documents containing stories from Reuters news agency, namely the Reuters data-set. We used Reuters-21578, the newer version of the corpus. It was compiled by David Lewis in 1987 and is publicly available at

http://www.research.att.com/lewis.

To obtain a training set and test set there exists different splits of the corpus. We used the Modified Apte ("ModeApte") split. The "ModeApte" split comprises 9603 training and 3299 test documents. A Reuters category can contain as few as 1 or as many as 2877 documents in the training set. Similarly a test set category can have as few as 1 or as many as 1066 relevant documents.
Medline1033 The Medline1033 is the second data-set which was used for experiments. This dataset comprises of 1033 medical documents and 30 queries obtained from National library of medicine. We focused on query23 and query20. Each of these two queries contain 39 relevant documents. We selected randomly 90% of the data for training the classifier and 10% for evaluation, while always having 24 relevant documents in the training set and 15 relevant documents in the test set. We performed 100 random splits of this data.

Experiments

The Reuters documents were preprocessed. We removed the punctuation and the words occurring in the stop list and also applied Porter stemmer to the words. We weighted the terms according to a variant of the $tfidf$ scheme. It is given by $\log(1+tf) * log(m/df)$, here tf represents term frequency, df is used for the document frequency and m is the total number of documents. The documents have unit length in the feature space.
We preprocessed the Medline documents by removing stop words and punctuation and weighted the words according to the variant of $tfidf$ described in the preceding paragraph. We normalised the documents so that no bias can occur because of the length of the documents. For evaluation we used the F1 performance measure. It is given by $2pr/(p+r)$, where p is the precision and r is the recall.
The first set of experiment was conducted on a subset of 3000 documents of Reuters21578 data set. We selected randomly 2000 documents for training and the remaining 1000 documents were used as a test set. We focused on the top

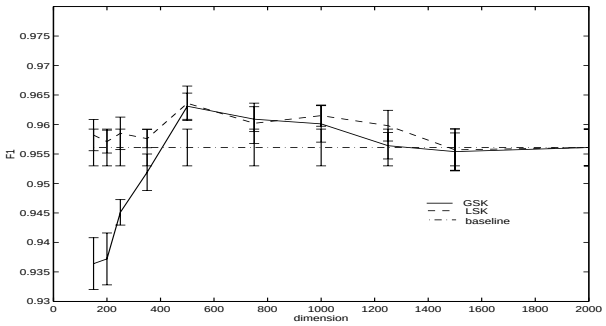

Figure 1. Generalisation performance of SVM with GSK, LSK and linear kernel for earn.

5 Reuters categories (earn, acq, money-fx, grain, crude). We trained a binary classifier for each category and evaluated its performance on new documents. We repeated this process 10 times for each category. We used an SVM with linear kernel for the baseline experiments. The parameter C that controls the trade off between error and maximisation of margin was tuned by conducting preliminary experiments. We chose the optimal value by conducting experiments on ten splits of one category. We ran an SVM not only in the reduced feature space but also in a feature space that has full dimension. The value of C that showed the best results in the full space was selected and used for all further experiments. For the Medline1033 text corpus we selected the value of C by conducting experiments on one split of the data. We ran an SVM in a feature space that has full dimension. The optimal value of C that showed best results was selected. Note that we did not use that split for further experiments. This choice does not seem perfect but on the basis of our experimental observation on Reuters, we conclude that this method gives an optimal value of C.

The results of our experiments on Reuters are shown in Figures 1 through 4. Note that these results are averaged over 10 runs of the algorithm. We started with a small dimensional feature space. We increased the dimensionality of the feature space in intervals by extracting more features.

These figures demonstrate that the performance of the LSK method is comparable to the baseline method. The generalisation performance of an SVM classifier varies by varying the dimensionality of the semantic space. By increasing the value of k, F1 numbers rise reaching a maximum and then falls to a number equivalent to the baseline method. However this maximum is not substantially different from the baseline method. In other words sometimes we obtain only a modest gain by incorporating more information into a kernel matrix.

Figure 6 and Figure 7 illustrate the results of experiments conducted on the two Medline1033 queries. These results are averaged over 100 random runs of the algorithm. For these experiments we start with a small number of dimensions. The dimensionality was increased in intervals by extracting more features. The results

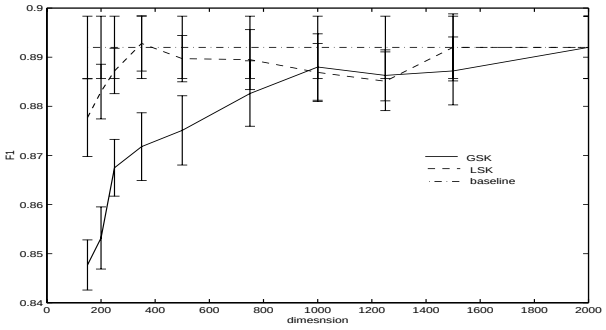

Figure 2. Generalisation performance of SVM with GSK, LSK, and linear kernel for acq.

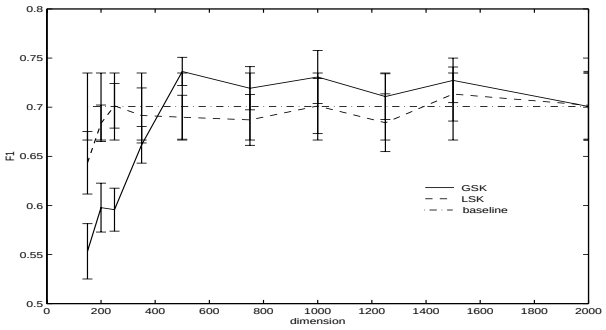

Figure 3. Generalisation performance of SVM with GSK, LSK and linear kernel for money-fx.

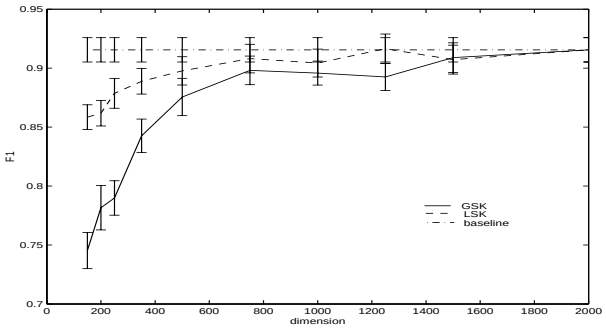

Figure 4. Generalisation performance of SVM with GSK, LSK and linear lkernel for grain.

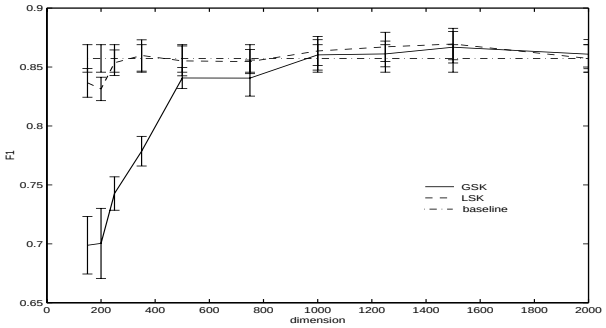

Figure 5. Generalistaion performance of SVM with GSK, LSK and linear kernel for crude.

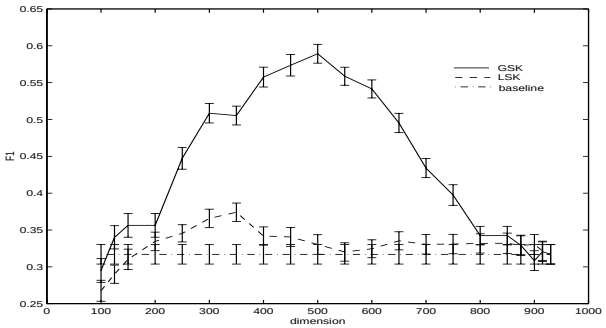

Figure 6. Generalisation performance of SVM with GSK, LSK and linear kernel for query23.

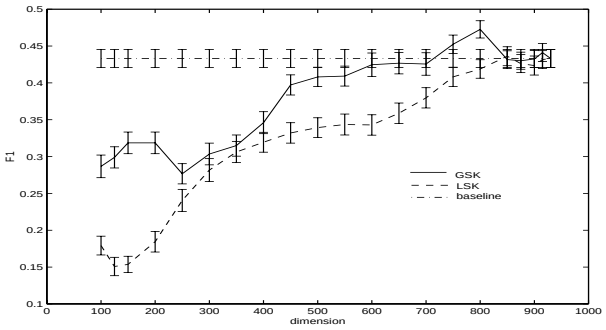

Figure 7. Generalisation performance of SVM with GSK, LSK and linear kernel for query20.

Table 2. F1 numbers for varying dimensions of feature space for a SVM classifier with LSK and SVM classifier with linear kernel (baseline) for ten Reuters categories

category	k			baseline
	100	200	300	
earn	0.962	0.958	0.961	0.959
acq	0.789	0.855	0.848	0.859
money-fx	0.62	0.673	0.635	0.6
grain	0.664	0.661	0.67	0.727
crude	0.431	0.558	0.576	0.575
trade	0.568	0.683	0.66	0.657
interest	0.478	0.497	0.5	0.517
ship	0.422	0.544	0.565	0.565
wheat	0.514	0.51	0.556	0.624
corn	0.194	0.1	0.133	0.133
micro-avg	0.786	0.815	0.815	0.819

for query23 are very encouraging showing that the LSK has a potential to show a substantial improvement over the baseline method. Thus the results (Reuters and Medline1033) show that in some cases there can be improvements in performance, while for others there can be no significant improvements.

Our results on Reuters and Medline1033 datasets demonstrates that GSK is a very effective approximation strategy for LSK. In most of the cases the results are approximately the same as LSK. However it is worth noting that in some cases such as Figure 6, GSK may show substantial improvement not only over the baseline method but also over LSK.

Hence the results demonstrate that GSK is a good approximation strategy for LSK. It can improve the generalisation performance over LSK as is evident from the results on the Medline data. It can extract informative features that can be very useful for classification. GSK can achieve a maximum at a high dimension in some situations. This phenomenon may cause practical limitations for large data sets. We have addressed this issue and developed a generalised GSK algorithm for text classification.

Furthermore, we conducted another set of experiments to study the behaviour of and SVM classifier with a semantic kernel and an SVM classifier with a linear kernel in a scenario where a classifier is learnt using a small training set. We selected randomly 5% of the training data (9603 documents). We focused on the top 10

categories (earn, 144), (acq, 85), (money-fx, 29), (grain, 18), (crude, 16), (trade, 28), (interest, 19), (ship, 12), (wheat, 8), (corn, 6). Note that the number of relevant documents are shown with the name of the categories. A binary classifier was learnt for each category and was evaluated on the full test set of (3299) documents. C was tuned on one category.
F1 numbers obtained as a results of these experiments are reported in Table 2. Micro-averaged F1 numbers are also given. We set the value of $k = 100, 200, 300$. It is to be noted that there is gain for some categories, but that there is loss in performance for others. It is worth noting that an SVM classifier trained with a semantic kernel can perform approximately the same as the baseline method even with 200 dimensions. These results demonstrate that the proposed method is capable of performing reasonably well in environments with very few labelled documents.

6.2. Experiments on Non-text Data

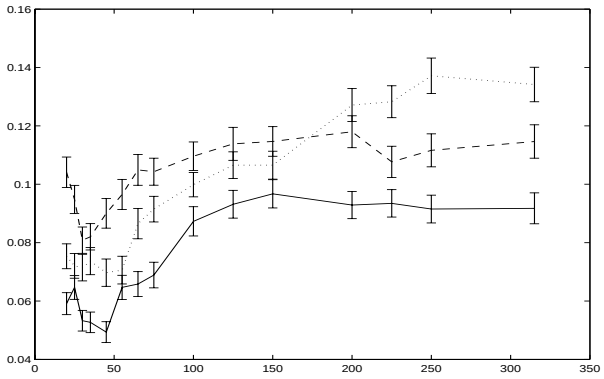

Figure 8. Generalization error for polynomial kernels of degrees 2,3, 4 on Ionosphere data (averaged over 100 random splits) as a function of the dimension of the feature space.

Now we present the experiments conducted on the non-text Ionosphere data set from the UCI repository. Ionosphere contains 34 features and 315 points. We measured the gain of the the LSK by comparing its performance with an SVM with polynomial kernel.
The parameter C was set by conducting preliminary experiments on one split of the data keeping the dimensionality of the space full. We tried $C = 0.01, 0.1, 1, 2, 4, 10$. The optimal value that demonstrated minimum error was chosen. This value was used for all splits and for the reduced feature space. Note that the split of the data used for tuning the parameter C was not used for further experiments.
The results are shown in Figure 8. These results are averaged over 100 runs. We begin experiments by setting k to a small value. We increased the dimensionality of the space in intervals. The results show that test error was greatly reduced when the

dimension of the feature space was reduced. The curves also demonstrate that the classification error of an SVM classifier with semantic kernel reaches a minimum. It makes some peaks and valleys before showing results equivalent to the baseline method. These results demonstrate that the proposed method is so general that it can be applied to domains other than text. It has a potential to improve the performance of a SVM classifier by reducing the dimension. However in some cases it can show no gain and may not be successful in reducing the dimension.

7. A Generalised Version of GSK algorithm for Text Classification

In this section we present a generalised version of the GSK algorithm. This algorithm arose as a result of experiments reported in Section 6. Some other preliminary experiments also contributed to the development of the algorithm.
The GSK algorithm presented in the previous section extracts features relative to the documents but irrespective of their relevance to the category. In other words, features are not computed with respect to the label of a document. Generally the category distribution is very skewed for text corpora. This establishes a need to bias the feature computation towards the relevant documents. In other words, if we can introduce some bias in this feature extraction process, the computed features can be more useful and informative for text classification.
The main goal of developing the generalised version of the GSK algorithm is to extract few but more informative features, so that when fed to a classifier it can show high effectiveness in a low number of dimensions.
To achieve the goal described in the preceding paragraph we propose the algorithm shown in Figure 9. GSK is an iterative procedure that greedily selects a document at each iteration and extracts features. At each iteration the criterion for selecting a document is the maximum residual norm. The generalised version of GSK algorithm focuses on relevant documents by placing more weight on the norm of relevant documents.
The algorithm transforms the documents into a new (reduced) feature space by taking a set of documents. As input an underlying kernel function, number T and bias B are also fed to the algorithm. The number T specifies the dimension of the reduced feature space, while B gives the degree to which the feature extraction is biased towards relevant documents.
The algorithm starts by measuring the norm of each document. It concentrates on relevant documents by placing more weight on the norm of these documents. As a next step a document with a maximum norm is chosen and features are extracted relative to this document. This process is repeated T times. Finally the documents are transformed into a new T dimensional space. The dimension of the new space is much smaller than the original feature space. Note that when there is enough positive data available for training, equal weights can be given both to relevant and irrelevant documents.
The generalised version of the GSK algorithm provides a practical solution of the problem that may occur with the GSK-algorithm. This algorithm may show good

Require: A kernel k, training set $\{(d_1, y_1), \ldots, (d_n, y_n)\}$, bias B and number T
for $i = 1$ to n **do**
norm2$[i] = k(d_i, d_i)$;
end for
for $j = 1$ to T **do**
for $i = 1$ to n **do**
if $(y_i == +1)$ **then**
biasednorm2$[i] = B *$ norm2$[i]$;
else
biasednorm2$[i]$ = norm2$[i]$;
end if
end for
$i_j = \text{argmax}_i(\text{biasednorm2}[i])$;
index$[j] = i_j$;
size$[j] = \sqrt{\text{norm2}[i_j]}$;
for $i = 1$ to n **do**
feat$[i, j] = \left(k(d_i, d_{i_j}) - \sum_{t=1}^{j-1} \text{feat}[i, t] * \text{feat}[i_j, t]\right) / \text{size}[j]$;
norm2$[i] =$ norm2$[i] -$ feat$(i, j) *$ feat(i, j);
end for
end for
return feat$[i, j]$ as the j-th feature of input i;
To classify a new example d:
for $j = 1$ to T **do**
newfeat$[j] = \left(k(d, d_{i_j}) - \sum_{t=1}^{j-1} \text{newfeat}[t] * \text{feat}[i_j, t]\right) / \text{size}[j]$;
end for
return newfeat$[j]$ as the j-th feature of the example d;

Figure 9. A Generalised Version of GSK Algorithm

generalisation at high dimension when there is not enough training data. In that scenario the generalised version of the GSK-algorithm shows similar performance at lower dimensions. The complete pseudo-code of the algorithm is given in Figure 9.

8. Experiments with Generalised GSK-algorithm

We employed the generalise the GSK algorithm to transform the Reuters documents into a new reduced feature space. We evaluated the proposed method by conducting experiments on the full Reuters data set. We used the ModeApte version and performed experiments on 90 categories that contain at least one relevant document both in the training set and test set. In order to transform documents into a new space, two free parameters T (dimension of reduced space) and B (bias) need to be tuned. We analysed the generalistion performance of an SVM classifier with respect to B by conducting a set of experiments on 3 Reuters categories. The results of these experiments are shown in Table 3. For this set of experiments we set the dimensionality of space (T) to 500 and varied B. The results demonstrate that the extraction of features in a biased environment can be more informative and useful when there is insufficient training data. On the basis of these experiments we selected an optimal value of B for our next set of experiments. Note that we selected the optimal value of C by conducting preliminary experiments on one Reuters category.
We set the value of $T = 500, 1000$. The results of this set of experiments are given in Table 4. We have given F1 value for 500, and 1000 dimensional space. Micro-averaged F1 values are also shown in the table. In order to learn a SVM classifier we used SVM^{light} [8] for the experiments described in this section.
These results show that the generalised GSK algorithm can be viewed as a substantial dimensionality reduction technique. Our observation is that the proposed method shows results that are comparable to the baseline method at a dimensionality of 500. Note that for the baseline method we employed an SVM with a linear kernel. It is to be noted that after 500 dimensionality there is a slow improvement in generalisation performance of the SVM. The micro-averaged F1 values for an SVM with generalised GSK is 0.822 (at 500 dimensions), whereas the micro-averaged F1 number for an SVM with linear kernel is 0.854. These results show that the performance of the proposed technique is comparable to the baseline method.
These results show that the generalised GSK algorithm is a practical approximation of LSK. If the learning algorithm is provided with enough positive training data, there is no need to bias the feature extraction process. However, when the learning algorithm does not have enough positive training data, an SVM may only show good performance at high dimensionality leading to practical limitations. However the introduction of bias towards relevant documents will overcome this problem, hence making it a technique that can be applied to large data sets.

Table 3. F1 numbers for acq, money-fx and wheat for different values of B.

B	acq	money-fx	wheat
1.0	0.922	0.569	0.707
1.05	0.913	0.678	0.851
1.1	0.864	0.695	0.855
1.15	0.864	0.723	0.846
1.2	0.864	0.756	0.846
1.4	0.864	0.754	0.844
1.6	0.864	0.751	0.862
1.8	0.864	0.755	0.853
2.0	0.864	0.748	0.846
2.2	0.864	0.752	0.855
2.4	0.864	0.756	0.846
2.6	0.864	0.748	0.846
2.8	0.864	0.752	0.846
3.0	0.864	0.756	0.846
4.0	0.864	0.755	0.864
6.0	0.864	0.752	0.857
10.0	0.864	0.731	0.857

Table 4. F1 numbers for top-ten Reuters categories..

Category	T 500	T 1000	baseline
earn	0.977	0.979	0.982
acq	0.923	0.934	0.948
money-fx	0.755	0.754	0.775
grain	0.894	0.902	0.93
crude	0.872	0.883	0.880
trade	0.733	0.763	0.761
interest	0.627	0.654	0.691
ship	0.743	0.747	0.797
wheat	0.864	0.851	0.87
corn	0.857	0.869	0.895
micro-avg (10)	0.903	0.909	0.919
micro-avg (90)	0.822	0.836	0.854

9. Conclusion

The paper has studied the problem of introducing semantic information into a kernel based learning method. The technique was inspired by an approach known as Latent Semantic Indexing borrowed from Information Retrieval. LSI projects the data into a subspace determined by choosing the first singular vectors of a singular value decomposition. We have shown that we can obtain the same inner products as those derived from this projection by performing an equivalent projection onto the first eigenvectors of the kernel matrix. Hence, it is possible to apply the same technique to any kernel defined feature space whatever its original dimensionality. We refer to the derived kernel as the Latent Semantic Kernel (LSK).

We have experimentally demonstrated the efficacy of the approach on both text and non-text data. For some datasets substantial improvements in performance were obtained using the method, while for others little or no effect was observed. As the eigenvalue decomposition of a matrix is relatively expensive to compute, we have also considered an iterative approximation method that is equivalent to projecting onto the first dimension derived from a Gram-Schmidt othogonalisation of the data. Again we can perform this projection efficiently in any kernel defined feature space and experiments show that for some datasets the so-called Gram-Schmidt Kernel (GSK) is more effective than the LSK method.

Despite this success, for large imbalanced datasets such as those encountered in text classification tasks the number of dimensions required to obtain good performance grows quite large before relevant features are drawn from the small number of positive documents. This problem is addressed by biasing the GSK feature selection procedure in favour of positive documents hence greatly reducing the number of dimensions required to create an effective feature space.

The methods described in the paper all have a similar flavour and have all demonstrated impressive performance on some datasets. The question of what it is about a dataset that makes the different semantic focusing methods effective is not fully understood and remains the subject of ongoing research.

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 ESPRIT Working Group in Neural and Computational Learning, NeuroCOLT2, Nr. 27150. KerMIT. and the 1st Project 'Kernel methods for images and text', KerMIT, Nr. 1st-2000-25431.

References

1. M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
2. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.
3. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.
4. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, pages 391–407, 1990. 41(6).
5. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management*, 1998.
6. S. T. Dumais, T. A. Letsche, M. L. Littman, and T. K. Landauer. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Spring Symposuim on Cross-Language Text and Speech Retrieval*, 1997.
7. R. Herbrich, K. Obermayer, and T. Graepel. Large margin rank boundaries for ordinal regression. In A.J. Smola, P. Bartlett, B. Schölkopf, and C. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.
8. T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel - Methods Support Vector Learning*. MIT Press, 1999.
9. Fan Jiang and Michael L. Littman. Approximate dimension equalization in vector-based information retrieval. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan-Kauffman, 2000.
10. T. Joachims. Text categorization with support vector machines. In *Proceedings of European Conference on Machine Learning (ECML)*, 1998.
11. Leopold, Edda, and Kinderman. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, to appear.
12. G. Miller, R. Beckwith, C. Fellbaum, D Gross, and K. Miller. Five papers on wordnet. Technical report, Stanford University, 1993.
13. M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out. In A.J. Smola, P. Bartlett, B. Schölkopf, and C. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 2000.
14. William H. Press. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.
15. G. Salton, A. Wang, and C.S. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620, 1975.
16. C. Saunders, A. Gammermann, and V. Vovk. Ridge regression learning algorithm in dual variables. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*. Morgan Kaufmann, 1998.
17. B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors,

Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing, pages 147 – 152. Springer Verlag, 1998.

18. B. Schölkopf, A. J. Smola, and K. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 327–352. MIT Press, 1999.
19. B. Schölkopf, R.C. Williamson, A.J. Smola, and J. Shawe-Taylor. SV estimation of a distribution's support. In *Neural Information Processing Systems*, 2000. forthcoming.
20. J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.
21. J. Shawe-Taylor and N. Cristianini. Margin distribution and soft margin. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 349–358, Cambridge, MA, 2000. MIT Press.
22. P. Sheridan and J.P. Ballerini. Experiments in multilingual information retrieval using the spi-der system. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 58–65. Association of Computing Machinery, 1996.
23. G. Siolas and F. d'Alché Buc. Support vectors machines based on a semantic kernel for text categorization. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, Como, 2000. IEEE.
24. A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 1998. Invited paper, in press.
25. A. J. Smola, O. L. Mangasarian, and B. Schölkopf. Sparse kernel feature analysis. Technical Report 99-04, University of Wisconsin, Data Mining Institute, Madison, 1999.
26. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
27. S.K.M. Wong, W. Ziarko, and P.C.N. Wong. Generalized vector space model in information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–25. ACM, 1985.