Kernel Methods for Exploratory Pattern Analysis: A Demonstration on Text Data

Tijl De Bie[1] and Nello Cristianini[2]

[1] K.U.Leuven, ESAT-SCD
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
tijl.debie@esat.kuleuven.ac.be
www.esat.kuleuven.ac.be/~tdebie
[2] U.C.Davis, Statistics Dept.
360 Kerr Hall, One Shields Ave., Davis, CA 95616, USA
nello@support-vector.net
www.kernel-methods.net

Abstract. Kernel Methods are a class of algorithms for pattern analysis with a number of convenient features. They can deal in a uniform way with a multitude of data types and can be used to detect many types of relations in data. Importantly for applications, they have a modular structure, in that any kernel function can be used with any kernel-based algorithm. This means that customized solutions can be easily developed from a standard library of kernels and algorithms. This paper demonstrates a case study in which many algorithms and kernels are mixed and matched, for a cross-language text analysis task. All the software is available online.

1 Introduction

Kernel Methods (KMs) offer a very general framework for performing pattern analysis on many types of data. In this paper we focus on text data, where text is chosen as an example of non-numeric data, and we demonstrate the versatility of this approach by performing cluster analysis, classification, correlation analysis and visualization on this data. What is more important, we do this by using different representations of our data defined by different kernel functions, as will be explained below. Overall, the purpose of this work is to make clear how different components can be combined together, to easily produce a wide variety of data analysis algorithms.

The main idea of kernel methods is to embed the dataset $S \subseteq X$ into a (possibly high dimensional) vector space $\Re^N$, and then to use linear pattern analysis algorithms to detect relations in the embedded data. Linear algorithms are extremely efficient and well-understood, both from a statistical and computational perspective. The embedding map is denoted here by $\phi : X \rightarrow \Re^N$, and it is understood that X can be any set.

An important point is that the embedding does not need to be performed explicitly: we do not actually need the coordinates of all the image vectors of

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 16–29, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the data in the embedding space $\Re^N$, we can perform a number of algorithms just knowing their relative positions in it. To be more accurate, if we know all the pairwise inner products $\langle \phi(x), \phi(z) \rangle$ between image vectors for all pairs of datapoints $x, z \in X$, we can perform most linear pattern discovery methods known from multivariate statistics and machine learning without ever needing the coordinates of such data points.

This point is important because it turns out that it is often easy to compute the inner product in the embedding space, even when the dimensionality N is high and so the coordinate vectors would be very large. It is often possible to find a (cheaply computable) function that returns the inner product between the images of any two data points in the feature space, and we call it a kernel. Formally, if we have data $x, z \in S \subseteq X$ and a map $\phi : X \to \Re^N$, we call kernel a function such that

$$K(x, z) = \langle \phi(x), \phi(z) \rangle$$

for every $x, z \in \Re^N$. As mentioned above, x and z can be elements of any set, and in this case study they will be text documents. On the other hand, their image $\phi(x)$ is a vector in $\Re^N$. The matrix $K_{ij} = K(x_i, x_j)$ is called the kernel matrix. Armed with this tool, we can look for linear relations in very high dimensional spaces at a very low computational cost. If the map ϕ is non-linear, then this will provide us with an efficient way to discover non-linear relations in the data, by using well understood linear algorithms in a different space. What is even more powerful, is that if X is not a vector space itself, the use of kernels enables us to operate on generic entities with essentially algebraic tools.

The kernel matrix contains sufficient information to run many classic and new linear algorithms in the embedding space, including Support Vector Machines (SVM), Fisher's Linear Discriminant (FDA), Partial Least Squares (PLS), Ridge Regression (RR), Principal Components Analysis (PCA), K-means and Spectral Clustering (SC), Canonical Correlation Analysis (CCA), Novelty Detection (ND) and many others. We refer the reader to [11, 9, 13, 3, 10, 1, 12] for more information on these methods, to [2] for a tutorial on kernel methods based on eigenvalue problems (PCA, CCA, PLS, FDA and SC), and to [16, 15] for two nice examples of the use of kernel methods in real life problems. Owing to the level maturity already achieved in these algorithmic domains, recently the focus of kernel methods research is shifting towards the design of kernels defined on general data types (such as strings, text, nodes of a graph, trees, graphs,...). Major issues in kernel design are its expressive power and its efficiency of evaluation [5, 7, 14, 8, 6].

Since by now a wide variety of kernel functions has been developed, each equivalent to a specific embedding function, the set of kernel methods has culminated into a complete toolbox to deal with real life machine learning and exploratory data analysis problems. Here we demonstrate this idea by using a variety of algorithms in combination with different text and string kernels on the articles of the Swiss constitution, which is available in 4 languages: English, French, German and Italian. What is interesting for this demonstration: the constitution is divided into several groups of articles, each group under a

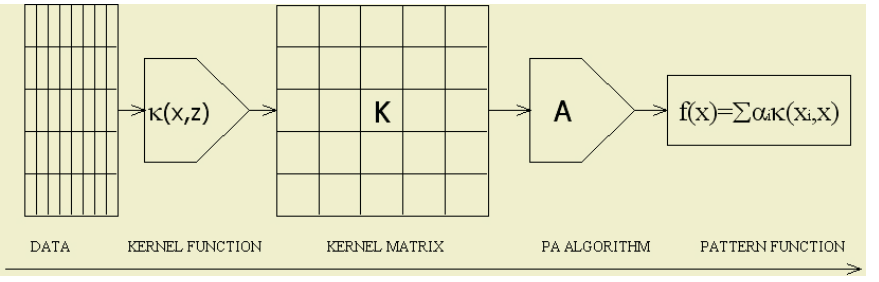

Fig. 1. A sketch of the modularity inherent in kernel-based algorithms: the data is transformed into a kernel matrix, by using a kernel function; then the pattern analysis algorithm uses this information to find the suitable relations, which are all written in the form of a linear combination of kernel functions.

different so-called 'Title' (in the English translation). All data can be found online at `www.admin.ch/ch/e/rs/c101.html`. A few articles were omitted in this case study (some because they do not have an exact equivalent in the different languages, 2 others because they are considerably different in length than the bulk of the articles), leaving a total 195 articles per language. The texts are processed by removing punctuation and stop words followed by stemming (where stop word removal and stemming are performed in a language specific way).

Ultimately, the aim of this simple case study is to exhibit how the inherent modularity of kernel methods makes them perfectly suited for fast and efficient deployment in a wide variety of tasks.

The entire matlab demo and the data used in this case-study, including scripts to remove punctuation and stop words and a stemming tool, are freely available online at `www.kernel-methods.net`, together with more free software. The pseudo-code and the detailed description of each algorithm and kernel used in this demo are described in the new book [12].

2 Pattern Algorithms

We briefly list here the algorithms that we will demonstrate. Given their large number and the space constraints of this article, it is impossible to even outline the theory behind them, so we refer the interested readers to the book [12]. All of these algorithms can work in any kernel-induced feature space, and all are amenable to statistical analysis based on Statistical Learning Theory. What we want to emphasize here is how all can be used as modules of a system, where any algorithm can be combined with any kernel, enabling practitioners to rapidly develop and test a large quantity of general purpose algorithms, and customize them by selecting the appropriate algorithms and kernels. Importantly, all algorithms (with the exception of k-means clustering) reduce to optimizing a convex function or to solving an eigenvalue problem.

Classification. One of the most classic tasks in pattern recognition is that of classification (or discrimination in the statistics literature, and categorization in text analysis). The goal is to find a function of the data that can be used to correctly assign a data item (e.g. a document) to one of a finite set of categories. A classic statistical method is Fisher's Linear Discriminant Analysis (FDA), and a classic method from machine learning is the Support Vector Machine algorithm (SVM) [3]. Both algorithms aim at finding a separating hyperplane in the embedding space, and differ in the properties of such hyperplane. In the first case (FDA) the hyperplane is chosen to maximize the proportion of the between class variance over the within class variance orthogonal to this hyperplane; in the second case (SVM) the hyperplane is chosen to maximize the margin.

Clustering. A second classic application in pattern recognition is the task of partitioning the samples in coherent groups. A common method for clustering vectorial data is K-means clustering. However K-means can be applied in a kernel induced feature space as well, making it applicable to virtually any kind of data using the kernel trick. As an alternative to K-means, we will demonstrate a more recently developed clustering technique known as spectral clustering (SC). This method is based on a cheap processing of the kernel matrix, followed by a simple eigenvalue problem.

Factor Analysis. When data is high dimensional (such as e.g. in text and bioinformatics applications), often the interesting information contained by the data can be explained by a number of underlying *factors* much smaller than this dimensionality. Depending on what is assumed to be interesting in a particular problem, different linear methods have been developed in multivariate statistics to extract these factors. The best known of these is principal component analysis (PCA), that finds a low dimensional projection of the data capturing as much of its variance as possible. Another method called canonical correlation analysis (CCA) can be used when we have two or more instantiations of the data that are all assumed to contain the relevant factors. CCA then proceeds by identifying those directions along which the data shows a large correlation between the different spaces. Both PCA and CCA can naturally be combined with kernels making it possible to identify hidden non-linear factors as well, or even factors explaining non-vectorial data such as text, trees, graphs,... For a survey on these methods based on eigenvalue problems, see [2].

3 Kernel Functions

All algorithms listed in the previous section are originally developed to be applied to vectorial data. However, for many other types of data it is possible to explicitly or implicitly construct a feature space capturing relevant information from this data. Unfortunately even when it can be expressed explicitly, often this feature space is so high dimensional that the algorithms can not be used in

their original form for computational reasons. However, as pointed out above, many of these algorithms can be reformulated into a kernel version. These kernel versions directly operate on the kernel matrix instead of on the feature vectors. For many data types, methods have been devised to efficiently evaluate these kernels, avoiding the explicit construction of the feature vectors. In this way, the introduction of kernels defined for a much wider variety of data structures significantly extended the application domain of these algorithms.

In this section we briefly discuss the various kernels we will demonstrate in this case study. All kernels used here are text kernels, and we always normalized them. For a detailed description we refer the reader to [12].

Bag of Words Kernel. A text document can be represented by the words occurring in it, without considering the order in which the words appear. Of course this is a less complete representation than the texts themselves, but for many practical problems this is sufficient. Consider the complete dictionary of words occurring in all texts. Then each text document x could be represented by a *bag of words* feature vector $\phi(x)$. The entries in this vector are indexed by the words in the vocabulary, and equal to the number of times the corresponding word occurs in the given text. Then, the bag of words kernel between two texts is defined as the inner product of their bag of words vectors: $K(x, z) = \langle \phi(x), \phi(z) \rangle$. Of course the feature vectors are usually sparse (since texts are usually much smaller than the dictionary size), and some care has to be taken to efficiently implement the bag of words kernel.

Figure (2) contains an image of the bag of words kernel on all articles (of all languages)[1]. One can distinguish a block structure, corresponding to the 4 languages. In these blocks, one can see some substructure in the articles, roughly corresponding to the Titles, Chapters, Sections... the articles are arranged in. This substructure reappears in all languages to some extent.

K-mer Kernel. Another – more generally applicable – class of kernels is the class of k-mer kernels [8]. For each document a feature vector is constructed indexed by all possible length-k strings (k-mer) of the given alphabet; the value of these entries is equal to the number of times this substring occurs in the given text. The kernel between two texts is then computed in the usual way, as the inner product of their corresponding feature vectors. Note that this kernel is therefore applicable to string data, also where no words can be distinguished, such as in DNA sequences. On the other hand, its power is generally less than a bag of words kernel wherever this can be used, such as on natural language. K-mer kernels capture the order $k-1$ Markov properties of the texts, which are specific to natural languages. Therefore, even for small k they are quite powerful already in distinguishing different languages.

[1] To avoid a completely black picture except for a bright diagonal, before visualizing the diagonal is subtracted from the kernel. This is necessary because text kernels generally have a very heavy diagonal.

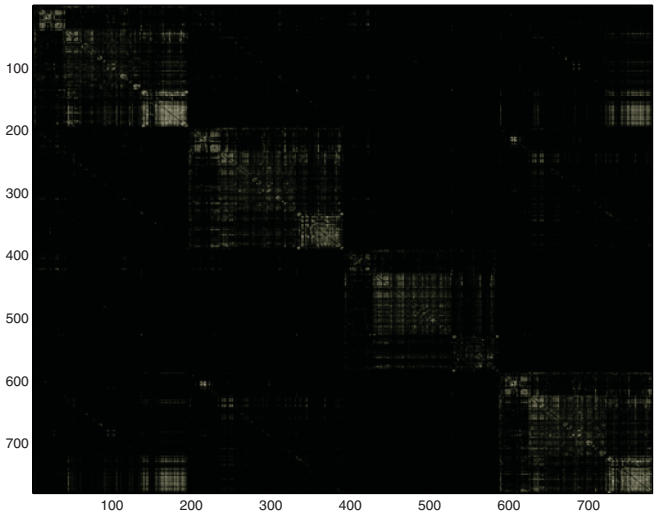

Fig. 2. A visualization of the full bag of words kernel matrix after normalization.Note that it is obvious from the figure that we have 4 distinct groups of texts, corresponding to the 4 different languages.

Note that the length of the feature vector is exponential in k, therefore a naive implementation would be prohibitively expensive for larger k. However efficient algorithms have been devised allowing the computation of this kernel for large scale problems [8].

Figures (3,4) contain the full 2-mer and 4-mer kernels. Figure (3) contains the part of the 4-mer kernel that corresponds to the English and French texts. Figure (5) above left, shows the same but now on the same articles artificially made noisy. Clearly the structure fades away.

Restricted Gappy K-mer Kernel. For noisy data, the k-mer kernel may be a bit too conservative in the sense that, even though two documents may be similar, still they don't share many k-mers. In that case, one may consider using a restricted gappy k-mer kernel. Consider feature vectors with entries corresponding to all possible k-mers again. Now, every entry is made equal to the number of k-mers up to (k+g)-mers in the text, that contain a (not necessarily contiguous) subsequence of length k equal to the k-mer of this specific entry. Here g is a parameter indicating the maximum number of gaps allowed. For details we refer the reader to [8], where an efficient way to evaluate such kernels is described. Figure (5) left below contains the restricted gappy 4-mer kernel on the same noisy texts.

Wildcard K-mer Kernels. This kernel adopts a different approach to deal with noisy texts. Now we use a feature space where each dimension corresponds

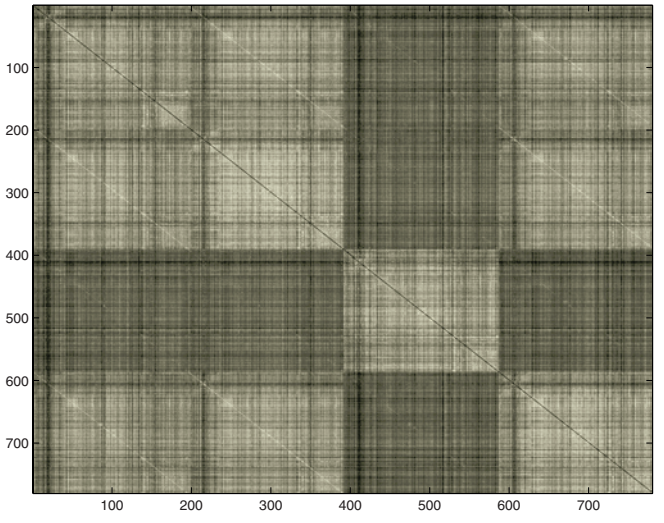

Fig. 3. The 2-mer kernel matrix after normalization. Again the cluster structure can be seen, however it is less clear than from the bag of words kernel. This is not surprising: a 2-mer kernel only takes into account 1st order Markov properties in the texts, making them probably less suitable for natural language applications. Note that the third group of texts –corresponding to the German language–, sticks out however, indicating that the 1st order Markov properties of German are significantly different from those of the other languages considered.

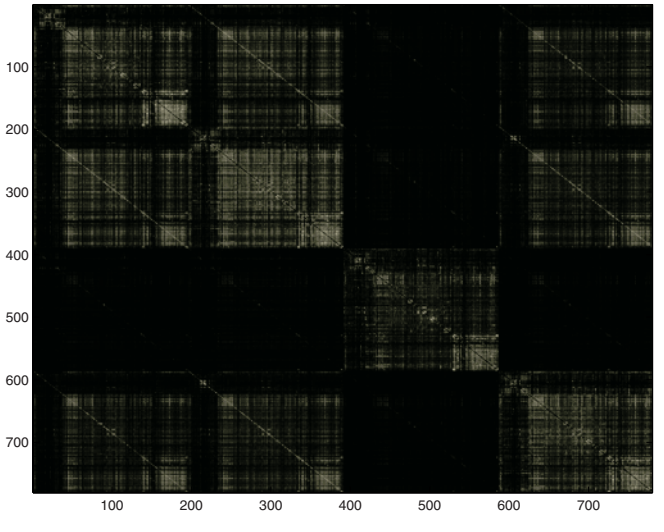

Fig. 4. The 4-mer kernel matrix after normalization. One can see that the distinction between the different languages is more clear now than for the 2-mer kernel.

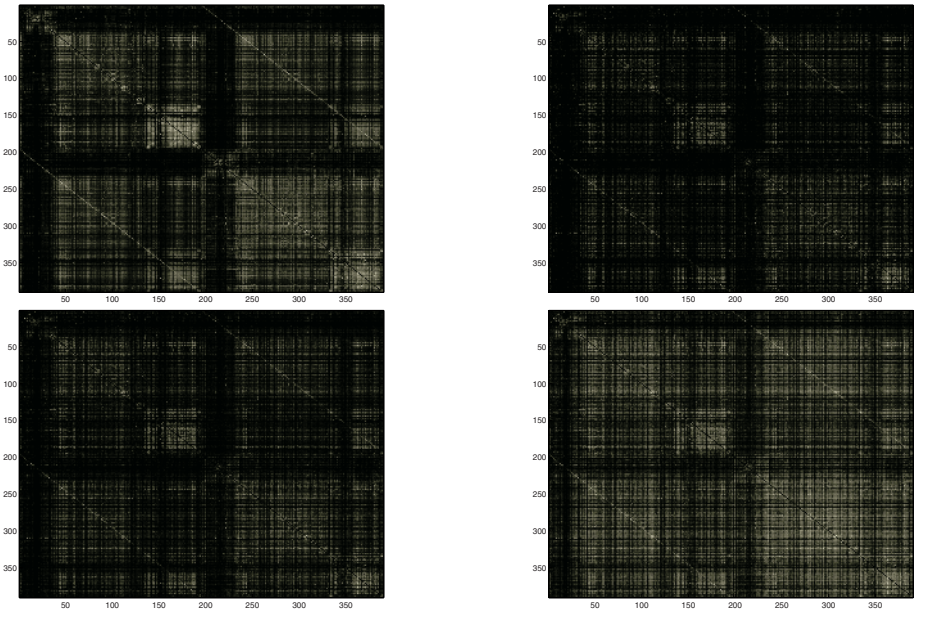

Fig. 5. Above left, the part of the normalized 4-mer kernel matrix corresponding to the English and French texts only is shown. Above right, the normalized 4-mer kernel matrix on the noisy English and French texts is depicted. One can see that the pattern has faded away a bit. The normalized restricted gappy 4-mer kernel matrix on the noisy English and French texts is shown below on the left. This kernel explicitly tries to take the noise influence into account. It is not immediately obvious from the figure, however experiments will show an improvement in performance of algorithms using this kernel over using the simple 4-mer kernel. The normalized wildcard 4-mer kernel matrix on the noisy English and French texts is shown below on the right. This kernel provides an alternative way to deal with noisy data. Also with this kernel algorithms will be shown to perform better than with the simple 4-mer kernel on the noisy data.

to a k-mer of the alphabet augmented with a wildcard. The number of wildcards in these k-mers is restricted by a parameter m. Then every feature is equal to the number of matches to this k-mer found in the text. Again, [8] describes an efficient way to evaluate such kernels. Figure (5) right below contains the wildcard 4-mer kernel on the same noisy texts.

4 A Case Study: Swiss Constitution Corpus

Thus far we have given an exposition of a wide variety of linear algorithms in machine learning that can be kernelized, and of different kernels applicable to text data. In what follows, we will show how each of these kernels can be used in the different algorithms. This inherent modularity in kernel methods is of major importance. Since for most types of data relevant kernels that can be evalu-

Table 1. Classification error rates on the noise free data, averaged over 100 randomizations with balanced 80/20 splits. The 2-mer kernel is used.

	English vs French	English vs German	English vs Italian
SVM	0.82 ± 0.05	0.03 ± 0.03	0.43 ± 0.04
FDA	5.0 ± 0.2	0 ± 0	1.2 ± 0.1

ated efficiently have been proposed in literature, the application domain is vast. Furthermore, this modularity has obvious advantages in software engineering.

4.1 Classification

Technical Notes. For FDA, we always took 1 for the regularization parameter. For the SVM we used the new-SVM formulation, and the regularization parameter ν was always chosen equal to 0.1.

Note that the ambition here is not to optimally tune the parameters: the main goal is to show how the modularity of kernel methods allows to apply a large library of algorithms to non-vectorial data; not to benchmark these algorithms.

Classifying Articles in Their Respective Language Classes

Noise Free. The first task we consider is the classification of texts into their respective language classes. The kernel we use here is the 2-mer kernel. We considered 3 binary classification problems, discriminating English texts from the texts in other languages (averaged over 100 random balanced splits in training (80%) and test sets (20%)). Error rates are in table 1.

English and French are hardest to distinguish based on the 2-mer kernel, which is probably due to many loan words present in English, recently adopted from French. Also, English and Italian are not perfectly distinguished (probably due to the same fact, and due to the fact that many English words have a Roman origin). German sticks out most clearly, which is to be expected. SVM's seem to have a better performance on this dataset.

Noisy, English versus French. Now let us consider the classification problem 'English vs French' in some greater detail. What happens if we add noise to the texts? We study this by artificially modifying the text by randomly deleting or altering 1/4th of the letters. Table 2 contains the average classification error rates for different kernels, along with the standard deviation on the estimated average, over 100 randomizations.

Note that the 4-mer kernel performs better that the 2-mer kernel. We can further improve the performance by using the restricted gappy and the wildcard 4-mer kernels.

Somewhat surprisingly FDA on the noisy data performed significantly worse with the restricted gappy as compared to the standard 4-mer kernel. However, clearly the method of choice here is SVM, which improves when using the restricted gappy or wildcard kernels.

Table 2. Classification error rates for different kernels on the noisy data, averaged over 100 randomizations with balanced 80/20 splits.

	2-mer	4-mer	gappy 4-mer	wildcard 4-mer
SVM, No noise	0.82 ± 0.05	0.42 ± 0.03	-	-
SVM With noise	2.89 ± 0.09	1.53 ± 0.07	1.28 ± 0.06	1.29 ± 0.06
FDA, No noise	5.0 ± 0.2	1.4 ± 0.1	-	-
FDA With noise	18.0 ± 0.5	8.4 ± 0.3	8.7 ± 0.3	10.9 ± 0.3

Table 3. Adjusted Rand index performances for spectral clustering and K-means clustering of the documents. The ideal clustering is clustering per language.

	bow	2-mer	4-mer
Spectral clustering	0.966 ± 0	0.437 ± 0	0.337 ± 0
K-means	0.38 ± 0.04	0.17 ± 0.03	0.26 ± 0.04

4.2 Clustering

Having shown that kernel methods allow to do classification in various ways, we will now show also clustering can be performed on data such as text in this case study. We will consider two methods: K-means clustering and spectral clustering.

Clustering the Articles in Their Language Clusters

Spectral Clustering. We cluster the articles of all languages, and check how well they are clustered into their respective language clusters. To assess the performance we use the adjusted Rand index [4], which is 1 for perfect clustering and has an expected value of 0 for random clustering. The final step consists of K-means on the eigenvectors, the clustering corresponding to the minimal K-means cost is taken over 10 starting values, chosen as described in [10].

K-means. Similarly, we perform kernel K-means on the documents. After 100 random initializations of K-means, the one with the best K-means cost is taken, and its adjusted Rand index is computed.

The results are summarized in table 3. The numbers in the table are averages over 10 runs along with the standard deviations on these averages. Note that spectral clustering (virtually) always returns the same optimal value (very small standard deviation), i.e. it is quite independent of the starting values in the K-means iterations, whereas K-means does not.

Somewhat surprisingly the 2-mer kernel performs better than the 4-mer kernel with the spectral clustering. As expected the best performance is achieved with the bow-kernel. The spectral method outperforms K-means in all cases.

Clustering the Articles into Coherent Groups

The articles in the constitution are organized into groups, called 'Titles'. Can we use clustering to automatically categorize the articles into their Titles?

Table 4. Adjusted Rand indices for spectral clustering of the English articles into the chapters they appear in.

		bow	2-mer	4-mer
English	Spectral clustering	0.326 ± 0	0.231 ± 0	0.328 ± 0.001
	K-means	0.24 ± 0.02	0.24 ± 0.02	0.27 ± 0.02
French	Spectral clustering	0.372 ± 0	0.206 ± 0	0.340 ± 0
	K-means	0.23 ± 0.03	0.17 ± 0.01	0.30 ± 0.02
German	Spectral clustering	0.559 ± 0	0.136 ± 0	0.241 ± 0
	K-means	0.13 ± 0.02	0.12 ± 0.01	0.19 ± 0.02
Italian	Spectral clustering	0.508 ± 0.001	0.214 ± 0	0.308 ± 0
	K-means	0.26 ± 0.02	$0.0.19 \pm 0.01$	0.31 ± 0.03

Spectral Clustering and K-means. See table 4 for the adjusted Rand scores achieved on this clustering problem for the different languages, kernels and methods. The performances are much less than for clustering articles into their language classes. This is of course to be expected: now the number of samples is smaller, and the distinction between languages is an objective criterion, while the distinction between Titles in the constitution is man-made and thus subjective in nature. Still, the performance is well above what a random clustering would do.

4.3 Factor Analysis

As a last type of applications discussed in this paper, we consider two methods for doing factor analysis: principal component analysis and canonical correlation analysis. Again, even though these techniques are originally developed to analyze vectorial data, the kernel trick allows us to apply them in a kernel induced feature space on a wide variety of data types. We demonstrate the methods here on the text data of our case study.

PCA. PCA is an algorithm to project the data in a lower dimensional space such that as much of the variance as possible is captured. The first two principal components are shown in figure 6. It can be seen that in this case indeed the directions of large variance seem to visualize some interesting cluster structure in the data.

CCA. With CCA one is able to capture information that is in common between several information sources. In this case, we have the same information in different languages. Since we are in fact interested in the semantic meaning of the articles, and not in the particularities of the languages, using CCA can be a good idea. Indeed, the division of the constitution articles into groups (the 'Titles' as they are called in the constitution) has something to do with their semantic context, and not with their particularities due to the language in which they are written.

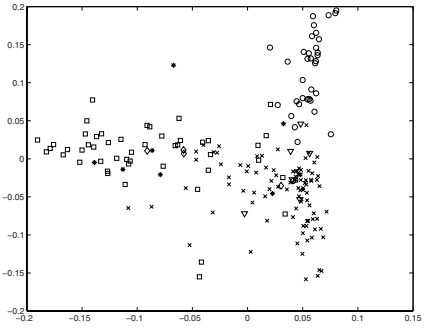

Fig. 6. First two principal components of the bag of words kernels for the English articles, as obtained by doing PCA. Articles from different chapters are represented by a different symbol.

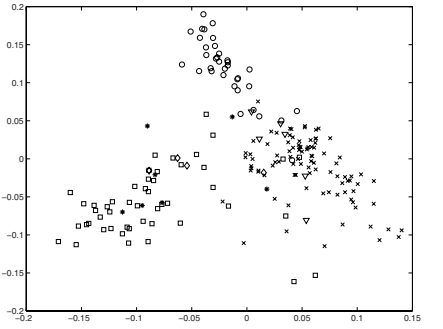

Fig. 7. First two canonical components of the bag of words kernels for the English articles, as obtained by doing CCA with the other languages. Articles from different chapters are represented by a different symbol.

We can use it here as a visualization tool: find two semantically interesting directions in the high dimensional feature space of the articles, and plot the components of the articles along these directions in the 2D plane[2]. The result can be seen in figure (7).

Apart from dimensionality reduction, CCA can also be used for cross-language text retrieval. For more information we refer the reader to the relevant literature [16].

Comparison of PCA with CCA. If we compare figure (6), where only one language is used, with figure (7), where the other languages are used to *supervise* the dimensionality reduction to some extent, we can see that the cluster structure is slightly more apparent when using CCA. We can assess this by computing the between class variance divided by the total variance (BCV/TV) in the subspaces found by PCA and CCA respectively. The larger this number, the better the class separation. The results for subspaces from 1 dimension up to 10 dimensions are shown in figure 8. Clearly CCA performs better than PCA, indicating that the different languages effectively supervise each other when selecting relevant dimensions in CCA.

5 Conclusion

We have demonstrated with a case study some of the most appealing features of kernel methods for pattern analysis: their modular design, the possibility of

[2] Note that training the regularization parameter is an issue here, and done by permutation analysis (the difference between the sum of the maximal correlations between the actual problem and a permuted version is maximized).

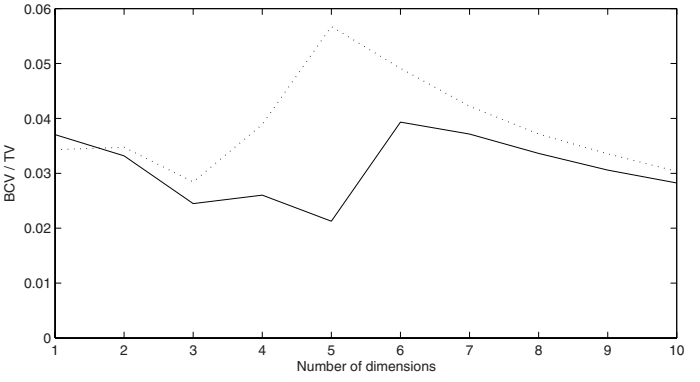

Fig. 8. Between class variance divided by total variance (BCV/TV) for PCA (full line) and CCA (dotted line) as a function of the dimension of the subspace (equivalently: the number of factors selected).

naturally using them for exploratory data analysis and rapid deployment, and their capability of operating seamlessly on non-numeric data. The theoretical details which are absent in this paper can be found in [12], and all the software and data are available at `www.kernel-methods.net`.

Acknowledgments

The authors thank John Shawe-Taylor for useful discussions and Manju Pai for contributing to part of the software development. TDB is a Research Assistant with the Fund for Scientific Research – Flanders (F.W.O.–Vlaanderen).

References

1. F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.
2. T. De Bie, N. Cristianini, and R. Rosipal. Eigenproblems in pattern recognition. In E. Bayro-Corrochano, editor, *Handbook of Computational Geometry for Pattern Recognition*. Springer-Verlag, 2004.
3. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, U.K., 2000.
4. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, page 193–218, 1985.
5. T. Jaakkola, M. Diekhans, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999.
6. H. Kashima, K. Tsuda, and A. Inokuchi. Kernel methods in computational biology. In B. Schoelkopf, K. Tsuda, and J.P. Vert, editors, *Handbook of Computational Geometry for Pattern Recognition*. Springer-Verlag, 2004.

7. R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.
8. C. Leslie and R. Kuang. Fast kernels for inexact string matching. In *Conference on Learning Theory and Kernel Workshop (COLT 2003)*, 2003.
9. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.
10. A. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
11. B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.
12. J. Shawe-Taylor and N. Cristianini. *Kernel methods for Pattern Analysis*. Cambridge University Press, Cambridge, U.K., 2004.
13. D.M.J. Tax and R.P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.
14. K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10):2397–2414, 2002.
15. J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel cca, 2003.
16. A. Vinokourov, N. Cristianini, and J. Shawe-Taylor. Inferring a semantic representation of text via cross-language correlation analysis, 2002.