On Problem-Oriented Kernel Refining*

Parrado-Hernández, E., Arenas-García, J., Mora-Jiménez, I.,
and Navia-Vázquez, A.
Dept. Signal Theory and Communications, Univ. Carlos III de Madrid
Avda Universidad 30, 28911-Leganés, Madrid, Spain

September 10, 2002

Abstract

Much attention has been recently devoted to those machine learning procedures known as kernel methods, the Support Vector Machines being an instance of them. Their performance heavily depends on the particular 'distance measurement' between patterns, function also known as 'kernel', which represents a dot product in the projection space. Although some attempts are being made to 'a priori' decide which kernel function is more suitable for a problem, no definite solution for this task has been found yet, since choosing the best kernel very often reduces to a selection among different possibilities by a cross-validation process. In this paper we propose a method for solving classification problems relying on the *ad-hoc* determination of a kernel for every problem at hand, i.e., a problem-oriented kernel design method. We iteratively obtain a semiparametric projecting function of the input data into a space which has an appropriately low dimension to avoid overfitting and complexity explosion of the resulting machine, but being powerful enough to solve the classification problems with good performance. The performance of the proposed method is illustrated using standard databases, and we further discuss its suitability for developing problem-oriented feature extraction procedures.

Keywords: kernel design, support vector machine, compact architectures, growing

1 Introduction

Kernel Methods (KM) have become a powerful technique within the machine learning community due to their ability to build non-linear versions of classical linear algorithms in an easy and elegant way [15]. The main idea underlying KM consists in projecting the input space onto a higher dimension space and applying in that space the corresponding linear algorithm. Support Vector Machines (SVMs) [16] stand as the most popular members of KM, and implement linear classifiers in the projected space that yield nonlinear decision boundaries in the input space. SVM based classifiers are able to determine maximal margin boundaries, what leads to excellent generalization capabilities. Examples of other Kernel-based algorithms are Nonlinear PCA [13], SVM Regressors [17], among several others.

An open issue in KM literature focuses on the design of kernel functions that provide the best conditions for a particular problem to be solved. Common kernels in the literature are

*This work has been partially supported by Spain CAM grant CAM-07T/0046/2000.

the linear, sigmoidal, polynomial, or Gaussian ones. The latter are considered to behave well in general, since they imply a projection onto an infinite dimension space, where any linear algorithm can then be successfully applied (for instance, any classification problem can be solved with minimal error, since enough degrees of freedom are always available; another issue is that of correct generalization, though, which is guaranteed by the hyperparameter selection and the capacity control imposed by the Support Vector Machine learning method). Special attention should be paid to those kernels specifically designed to deal with a particular -non numerical- input signal, such as those proposed for text classification [6] or in genomic processing [4]. We will not cover these issues here, since we assume that numerical inputs are available.

The commonly used kernel functions (Gaussian and polynomial) tend to project data onto very high dimension spaces, such that a tendency to overfitting exists. Support Vector Machines tend to limit this effect by controlling the capacity of the machine through the maximization of the margin. In any case, a large number of support vectors is usually found (every SV represents an axis in the projection space to represent new data), which does not benefit the generalization nor the complexity control of the machine. If we are mainly interested in solving a decision problem, we may ask the projection function to build the minimal representation of the input patterns in the projection space, by focusing from the beginning on the relevant data to solve the problem. Since the boundary is not known beforehand, an incremental procedure is needed. In this sense, the aim of this paper is to explore a method for constructing a problem oriented kernel (POKER) that better fits the characteristics of the problem at hand, and that also provides a more compact solution for the final classifier.

The outline of the paper is as follows: Section 2 introduces the formulation of the training algorithm for SVCs with semiparametric kernels and Section 3 presents an iterative scheme to construct those problem-oriented kernels leading to compact machines. Experimental results in several benchmark problems (both synthetic and real) are shown in Section 4 and finally, Section 5 collects the main conclusions of this paper and identifies lines for future research.

2 Building SVCs with semiparametric kernels

We will consider here a binary classification problem in which we have a training dataset $\{\vec{x}_1, \cdots, \vec{x}_l\}$ belonging to an input space $\mathcal{R}^n$ and the corresponding targets $y_i \in \{-1, +1\}, i = 1, \cdots, l$. Nonlinear SVM projects input data onto a higher dimension space $\mathcal{F}$, by means of a function $\vec{\phi}(\vec{x})$ and solves the problem in $\mathcal{F}$ with a linear classifier

$$\hat{y} = sign(f(\vec{x})) = sign\left(\vec{w}^T \vec{\phi}(\vec{x}) + b\right) \tag{1}$$

where $\vec{w}$ is a linear combination of certain $\vec{\phi}(\vec{x}_i)$ (those with $\alpha_i > 0$, called support vectors in SVM terminology):

$$\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{\phi}(\vec{x}_i) \tag{2}$$

Since the equations to solve the linear classifier only involve inner products in $\mathcal{F}$, there is no need to explicitly compute $\vec{\phi}(\vec{x})$, providing we can calculate the inner product $\left\langle \vec{\phi}(\vec{x}_i), \vec{\phi}(\vec{x}_j) \right\rangle$ from vectors $\vec{x}_i, \vec{x}_j$ [3]. Mercer's theorem [16] identifies certain functions or kernels associated to those projections whose evaluation on a pair of vectors $\vec{x}_i, \vec{x}_j$ equals the inner product in $\mathcal{F}$. This way:

$$\mathrm{k}(\vec{x}_i, \vec{x}_j) = \left\langle \vec{\phi}(\vec{x}_i), \vec{\phi}(\vec{x}_j) \right\rangle \tag{3}$$

By means of this kernel trick we can even consider infinite dimension spaces $\mathcal{F}$ as long as we know the corresponding kernel function (this is the case of the widely used Gaussian kernel). In consequence, suitable projection design has been converted into suitable kernel design. Under SVM formulation (see by instance [3, 15, 16]) the classifier is expressed according to

$$\hat{y} = sign(f(\vec{x})) = sign\left(\sum_{i=1}^{NSV} \alpha_i y_i \mathrm{k}(\vec{x}, \vec{x}_i) + b\right) \tag{4}$$

The above summatory incorporates only the kernel terms belonging to some of the training patterns (the support vectors), but usually this number is very large (in some problems even 20% -40% of the database). Another inconvenience of this approximation is that we can not analyze vectors directly in $\mathcal{F}$, what could be interesting in other tasks different from that of classification (such as feature selection). Some works [7, 8, 9, 15] have tried to get more compact solutions (i.e., with less kernel evaluations per pattern) by assuming the following expansion of the weight vector:

$$\vec{w} = \sum_{i=1}^{R} \beta_i \vec{\phi}(\vec{x}_i) \tag{5}$$

Doing so, the classifier evaluation involves the calculation of just R kernel functions for each pattern (usually $R \ll NSV$).

$$f(\vec{x}) = \sum_{i=1}^{R} \beta_i \mathrm{k}(\vec{x}, \vec{x}_i) + b \tag{6}$$

However, the projection and kernel functions have not changed at all and, by using this formulation, the goal is to find the β_i that best approximates the exact SVM solution in $\mathcal{F}$; in fact, these coefficients can be exactly calculated by first solving the complete SVM problem and then solving a pseudoinverse problem (as shown in [15]). Since the projection space has not changed in this new approach, this method still closes opportunities for a further data analysis in $\mathcal{F}$.

In this paper we follow a different approach and focus on designing a mapping function $\vec{\phi}(\vec{x})$ of finite dimension which allows both to control machine complexity and to operate in the projection space, while still achieving a good solution for the classification problem in terms of maximal margin. To do so, we propose to reinterpret equation (6) by introducing the following projection function that maps data onto a different space $\mathcal{F}_{PO}$ (Problem Oriented, PO) [1]:

$$\begin{aligned} \vec{\phi}_{PO}(\vec{x}) &= [\mathrm{k}(\vec{x}, \vec{x_1}), \cdots, \mathrm{k}(\vec{x}, \vec{x_R})]^T \\ \mathrm{k}_{PO}(\vec{x}, \vec{z}) &= \sum_{i=1}^{R} \mathrm{k}(\vec{x}, \vec{x}_i)\mathrm{k}(\vec{z}, \vec{x}_i) \end{aligned} \tag{7}$$

$\vec{\phi}_{PO}(\vec{x})$ is a projection function that incorporates some information about the projection of data in space $\mathcal{F}$ (in fact, $\mathcal{F}_{PO}$ as defined in equation (7) is a subspace of $\mathcal{F}$), but it also opens the possibility of grouping together information obtained from different spaces $\mathcal{F}_k$ with different kernel functions (Multi Kernel (MK) mapping function):

$$\vec{\phi}_{MK}(\vec{x}) = [\mathrm{k}'_1(\vec{x}, \vec{x}_1), \cdots, \mathrm{k}'_R(\vec{x}, \vec{x}_R)]^T$$

[1]Note that the components of $\vec{\phi}_{PO}(\vec{x})$ are calculated with the kernel function corresponding to $\mathcal{F}$.

$$\mathrm{k}_{MK}(\vec{x},\vec{z}) = \sum_{i=1}^{R} \mathrm{k}'_i(\vec{x},\vec{x}_i)\mathrm{k}'_i(\vec{z},\vec{x}_i) \tag{8}$$

The above definition of $\vec{\phi}_{PO}(\vec{x})$ and $\mathrm{k}_{PO}(\vec{x},\vec{y})$ enables us to obtain the exact SVM solution in $\mathcal{F}_{PO}$ (instead of the approximate one in $\mathcal{F}$ given by equation (6)). Moreover, since $\mathcal{F}_{PO}$ is a finite (and relatively low) dimension space, we can also analyze patterns projected in $\mathcal{F}_{PO}$, what would open up a promising line for feature selection algorithms from SVM classifiers as we discuss in Section 4.

The remaining of this section is devoted to review the formulation of the parametric kernel SVC so that it can be solved in a computationally efficient fashion with WLS-SVC [7], that also enables the application of growing schemes for the construction of a problem oriented architecture (see Section 3).

Let us now consider a linear classification function in $\mathcal{F}_{PO}$ [2]

$$\hat{y} = sign(f(\vec{x})) = sign\left(\vec{w}^T\vec{\phi}_{PO}(\vec{x}) + b\right) \tag{9}$$

where $\vec{w}$ and b result from the minimization of

$$L_p = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l}\xi_i \tag{10}$$

subject to constraints

$$\left.\begin{array}{r} y_i\left(\vec{\phi}_{PO}(\vec{x}_i)\vec{w} + b\right) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{array}\right\} \forall i = 1,\cdots,l \tag{11}$$

with respect to $\vec{w}$, b and ξ_i. After the incorporation of constraints into the functional with Lagrangian multipliers $\alpha_i \geq 0$ and $\eta_i \geq 0$ respectively, and by introducing the variables

$$e_i = y_i - \left(\vec{\phi}_{PO}(\vec{x}_i)\vec{w} + b\right) \tag{12}$$

$$a_i = \frac{2\alpha_i}{1 - y_i\left(\vec{\phi}_{PO}(\vec{x}_i)\vec{w} + b\right)} = \frac{2\alpha_i y_i}{e_i} \tag{13}$$

we can rewrite (10) as

$$L_p = \frac{1}{2}\|\vec{w}\|^2 + \frac{1}{2}\sum_{i=1}^{l} a_i e_i^2 + \sum_{i=1}^{l}\left(C - \alpha_i - \eta_i\right)\xi_i \tag{14}$$

which, apart from the last term that vanishes at the minimum, is a Weighted Least Squares functional. Taking derivatives with respect to $\vec{w}$, and b, and forcing them to be 0 at the solution, we get the following matrix expressions:

$$\begin{array}{r} \vec{w} + \mathbf{\Phi}\mathbf{D}_a\mathbf{\Phi}^T\vec{w} + \mathbf{\Phi}\vec{a}b = \mathbf{\Phi}\mathbf{D}_a\vec{y} \\ \vec{a}^T\mathbf{\Phi}^T\vec{w} + \vec{a}^T b = \vec{a}^T y \end{array} \tag{15}$$

[2]Note that in this case all vectors $\vec{\phi}_{PO}(\vec{x}_i)$ and w can be explicitly calculated

where

$$\mathbf{\Phi} = \left[\vec{\phi}_{PO}(\vec{x}_1), \cdots, \vec{\phi}_{PO}(\vec{x}_l)\right]$$
$$\vec{a} = [a_1, \cdots, a_l]^T$$
$$(\mathbf{D}_a)_{ij} = a_{ij}\delta(i-j)$$

We can now group both equations into a single system of linear equations

$$\begin{bmatrix} \left(\mathbf{\Phi}\mathbf{D}_a\mathbf{\Phi}^T + \mathbf{I}\right) & \mathbf{\Phi}\vec{a} \\ \vec{a}^T\mathbf{\Phi}^T & \vec{a}^T\vec{1} \end{bmatrix} \begin{bmatrix} \vec{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}\mathbf{D}_a\vec{y} \\ \vec{a}^T\vec{y} \end{bmatrix} \tag{16}$$

Note that (16) is an equation in $\vec{w}$, b and $\vec{a}$ that can not be solved simultaneously for all these variables. Thus, we use an iterative scheme consisting of 2 steps; after randomly initializing $\vec{w}$ and b:

- First, we calculate the corresponding a_i values by using (13)[3].
- Then, we consider the a_i values as constants and use (16) to update $\vec{w}$ and b.

This procedure continues until the minimum of (14) is reached. This type of iterated WLS scheme has asymptotic convergence as shown in [10]. The proof is based on the "Convergence of Block Coordinate Descent" theorem [1].

The extension of this algorithm to the Multi Kernel case is immediate, by just taking Φ to be

$$\mathbf{\Phi} = \left[\vec{\phi}_{MK}(\vec{x}_1), \cdots, \vec{\phi}_{MK}(\vec{x}_l)\right]$$

3 Growing Scheme for designing a Problem Oriented Kernel

In this section we present an iterative scheme for constructing a POKER that incorporates the knowledge about the decision boundary that has been learned by the classifier in the previous iterations. This growing algorithm, named Growing Support Vector Classifiers (GSVC), has already been proposed in [8, 9] to construct compact semi-parametric approximations to Support Vector Classifiers. The main idea underlying POKER-GSVC is to start with a very reduced machine (a few centroids for each class) and to iteratively add new centroids in order to achieve better representations of the decision boundary. At each iteration, the partial classification boundary is determined through the solution of an WLS-SVC problem [7]. Therefore, the POKER-GSVC is initialized with $M/2$ (typically $M = 2$ or 4) centroids from each class picked up at random from the input data. Then, the initial projection is computed as

$$\vec{\phi}_{PO}(\vec{x}) = [\mathrm{k}(\vec{x}, \vec{c_1}), \cdots, \mathrm{k}(\vec{x}, \vec{c_M})]^T \tag{17}$$

and after the application of WLS-SVC, the first classifier is given by

$$\hat{f}(\vec{x}) = \mathrm{sign}\left(\sum_{k=1}^{M} w_k \mathrm{k}(\vec{x}, \vec{c}_k)\right) \tag{18}$$

[3]Computation of α_i is straightforward, please refer to [7] for details.

Now it is necessary to increase the complexity of the projection $\vec{\phi}_{PO}$ to improve the decision boundary by incorporating some new centroids to the architecture of the classifier. According to SVM principles, the SVs are the critical input data that determine the boundary, i.e. all the information needed to construct the decision boundary that minimizes the structural risk is contained in these samples. WLS-SVC solves an optimal semiparametric approximation to a SVM classifier, so it also identifies the data points that support the initial boundary. Therefore, the criterion to select the new centroids consists in choosing them from the current SVs. In addition, to obtain good projections, orthogonality is largely preserved by selecting as new centroids samples lying far apart from those already forming part of the classifier. This procedure incurs in no additional cost since the value of the already computed components of $\vec{\phi}_{PO}$ can be used as a measure of distance for this maximal orthogonality purpose. After this update, projection $\vec{\phi}_{PO}$ is enhanced by centroids $\vec{c_1}^1$ to $\vec{c_N}^1$ (with N not necessarily equal to M) and given by

$$\begin{aligned} \vec{\phi}_{PO}(\vec{x})^{new} &= [\vec{\phi}_{PO}^T(\vec{x}), \vec{\phi}_{PO}^T(\vec{x})^{inc}]^T, \\ \text{with } \vec{\phi}_{PO}(\vec{x})^{inc} &= [\mathrm{k}(\vec{x}, \vec{c_1}^1), \cdots, \mathrm{k}(\vec{x}, \vec{c_N}^1)]^T \end{aligned} \tag{19}$$

The growing stage continues with the application of WLS-SVC and the whole procedure is iterated until the quality of projection $\vec{\phi}_{PO}$ is observed to worsen because of overfitting. To detect this fact, a cross-validation procedure is applied after each iteration to determine whether to stop or continue incorporating centroids. This stopping criterion, carried out on a separate subset of the training patterns that are not fed into WLS-SVC, has proved to endow the machine with excellent generalization capabilities [9]. Moreover, it is worth remarking the efficiency of POKER-GSVC in terms of computational resources since there is no need to recalculate from scratch the $\mathbf{\Phi}$ matrix after each iteration. Furthermore, WLS-SVC provides a classifier of maximum margin since it actually solves a SVM problem.

4 Experimental work

In this section, the proposed POKER-GSVC algorithm is first applied to two bidimensional datasets in order to illustrate the problem oriented built kernels, as well as serving for an initial discussion on the implicit feature extraction performed by the method. Then, we present results on several real datasets extracted from the UCI Machine Learning repository [2] to evaluate its performance against other SVM technologies (classical SVM).

In both cases we use gaussian kernels for all the coordinates of the problem oriented mapping function $\vec{\phi}_{PO}(\vec{x})$:

$$\mathrm{k}(\vec{x}, \vec{c_i}) = \exp\left(-\frac{\|\vec{x} - \vec{c_i}\|^2}{2\sigma^2}\right) \tag{20}$$

4.1 Graphical interpretation of the 'ad hoc' built kernels

We use here two simple bidimensional datasets to graphically illustrate the kernel designed by the POKER-GSVC for these problems. The first one is that of separating two classes drawn in concentric circles. The results have been detailed in Figure 1 below.

Note how the obtained boundary perfectly represents the maximal margin solution (Figure 1 (a)), equivalent to the solution which could have been obtained with a 2nd-degree polynomial

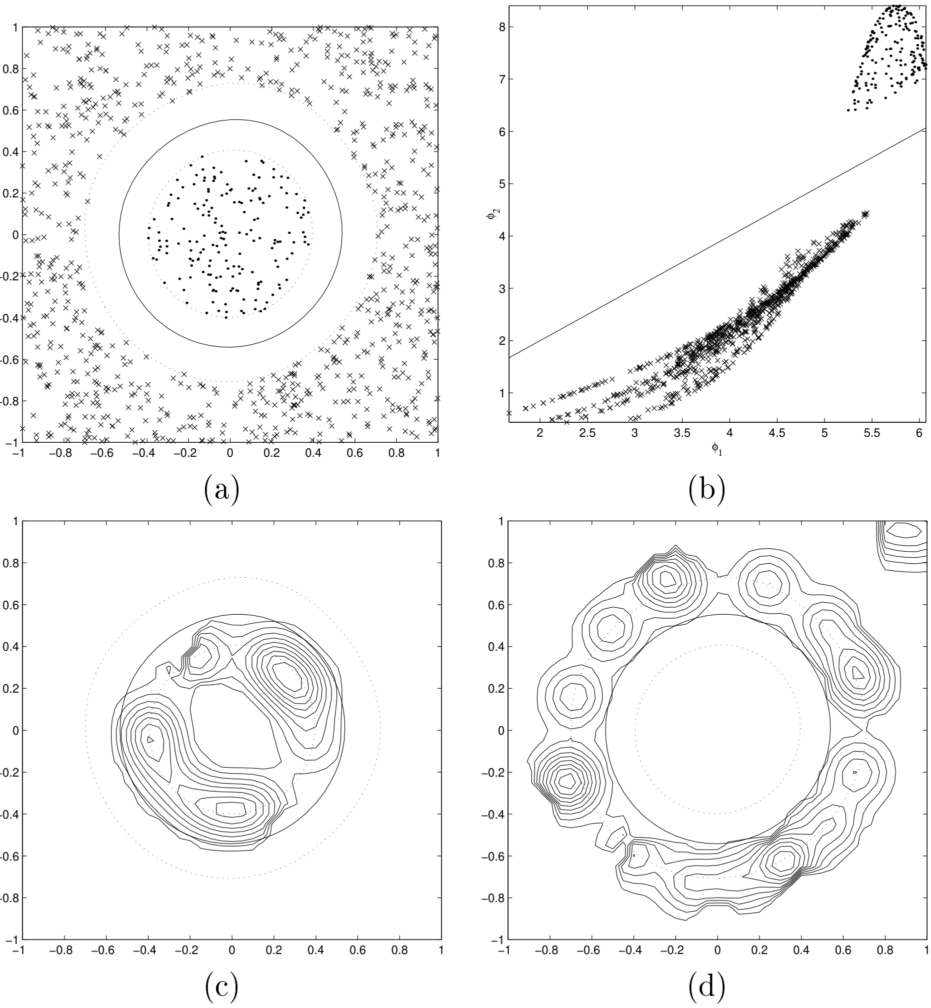


Figure 1: Results of the POKER-GSVC on the concentric circle problem. The obtained boundary is shown in (a), the representation of patterns in (ϕ_1, ϕ_2) space, where they become linearly separable (b), as well as the area of influence of feature 1 (c) and 2 (d) in input space.

kernel (known to be optimal for this particular problem). Therefore, POKER-GSVC is able to -semiparametrically- approximate (without 'a priori' information about the problem to be solved) the best kernel for the problem at hand. A trivial feature extraction would be that of separating between 'positive' and 'negative' contributions to the decision, what results in a mapping of input patterns into a bidimensional space (ϕ_1, ϕ_2), shown in Figure 1 (b). The problem is linearly separable in that space, and the maximal margin is also preserved. Preliminary representation of these two extracted features can be observed in Figure 1 (c) and (d), for ϕ_1 and ϕ_2 respectively: these features tend to concentrate their influence range nearby the boundary. Possibly, further processing of the projected data could lead to finer grain feature analysis, but that exceeds the scope of this paper. Analogous results have been represented for Ripley's dataset [11] in Figure 2, where similar comments could be done. Note, however, that the problem is no longer separable, which is reflected in both input space (a) and projected (ϕ_1, ϕ_2) space (b), as well as in the extended range of influence of features ϕ_1 and ϕ_2 (c) and (d).

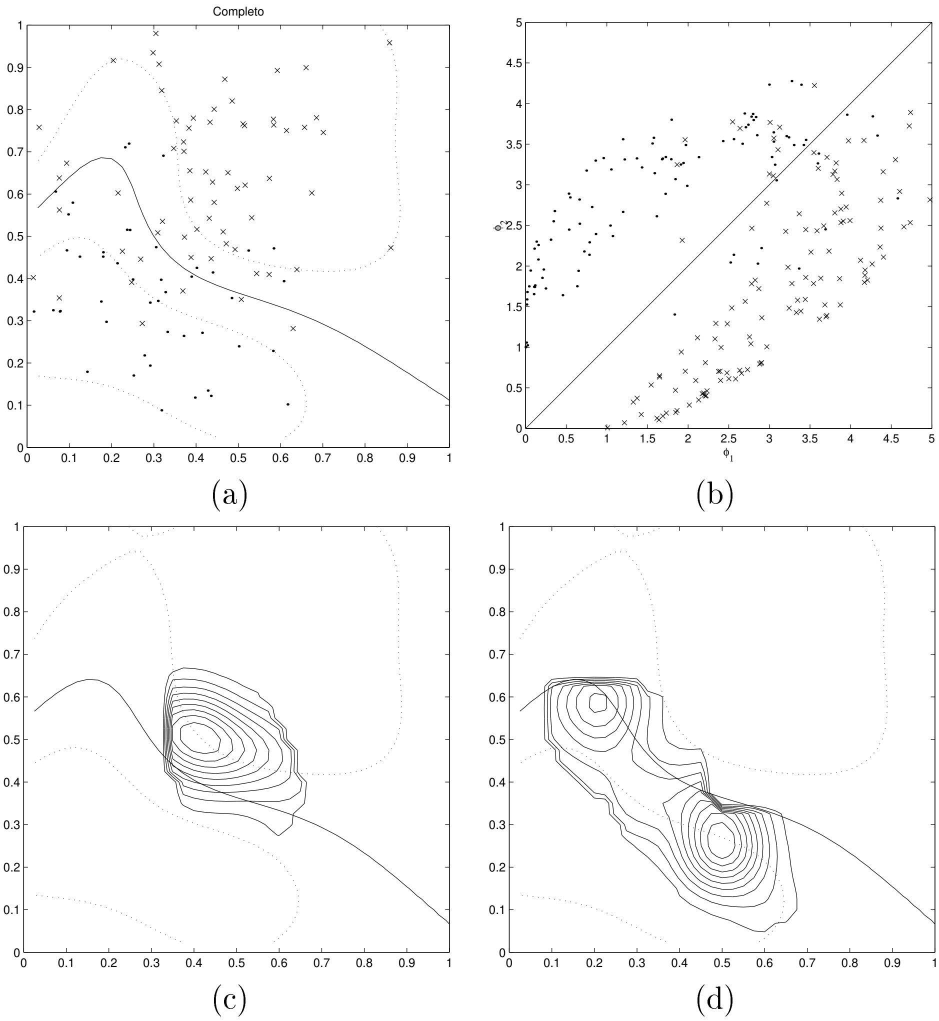


Figure 2: Results of the POKER-GSVC on Ripley's dataset. The obtained boundary is shown in (a), the representation of patterns in (ϕ_1, ϕ_2) space, where they become linearly separable (b), as well as the area of influence of feature 1 (c) and 2 (d) in input space.

4.2 Other experiments

We have selected for performance comparison a binary synthetic dataset "kwok" presented in [5] and several benchmark problems from the UCI Machine Learning Repository [2]: "diabetes", "waveform", "image" and "abalone". The last dataset has been converted into a two-class problem according to [12].

Table 1 shows the test classification error (CE) and machine size obtained by POKER-GSVC and classical SVM.

For both classifiers, we have selected the value of σ (gaussian kernel parameter) using a five-fold cross-validation method and exploring a range of 10 values between 0.1 and 10. All results in the table correspond to the average value and standard deviation obtained through 20 experiments. POKER-GSVC beats classical SVM in all cases but "kwok".

Column *machine size* refers to the number of kernel evaluations required to classify a pattern. We observe POKER-GSVC sizes are usually an order of magnitude or so below those of classical SVM (only in "image" problem sizes are similar).

	Test CE (%)		Machine Size		#Train Patt.
Problem	POKER-GSVC	SVM	POKER-GSVC	SVM	
abalone	**20.48**±0.6	24.04	**70.8**±24.46	1184	2507
waveform	**8.56**±0.3	8.9	**87.86**±23.89	892	4000
image	**3.82**±0.66	3.9	**186**±78.05	271	1848
diabetes	**23**±1.4	25	**19.6**± 4.6	295	576
kwok	12.09±0.37	**11.8**	**23.8**±6.8	122	500

Table 1: Performance comparison between POKER-GSVC and SVM in several benchmark problems. Test classification error (in percentage) is shown, as well as final machine sizes and the number of training patterns for each problem.

In addition to the advantages related to classification error and machine size, POKER-GSVC projects data into a space where vector images can be explicitly calculated. We illustrate the potential feature representation and extraction capabilities with projections onto feature space (ϕ_1, ϕ_2) for some of the datasets (Figure 3). It seems clear that further processing of these somewhat 'aggregated' two features could lead to more detailed features, for instance by applying Vector Quantization methods on this space or by differently aggregating kernels into more than two features. We propose this topic as future work, though.

5 Concluding remarks and future work

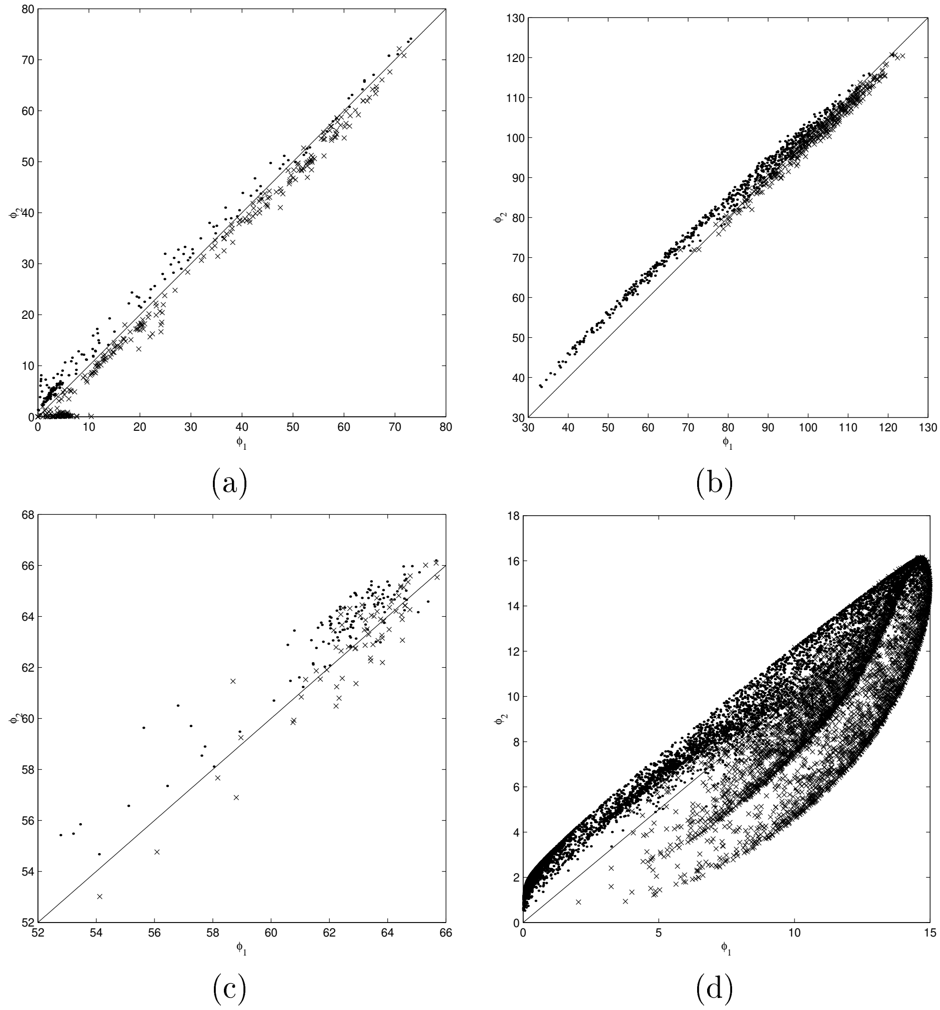
We have proposed a problem-oriented kernel extraction method, relying on a growing procedure to build Support Vector Classifiers. Experimental results reveal the good generalization capabilities of the method and the compactness of the resulting machines in terms of number of nodes. Furthermore, the graphical analysis of the implicit feature extraction carried out by the POKER-GSVC method shows that it actually extracts features that concentrate on the areas close to the decision boundary, or, alternatively interpreted, that the method is able to build 'ad hoc' kernels for a given problem. Preliminary analysis of the projection onto a bidimensional feature space indicates that it would be possible to further process these features to extract a finer grain feature representation, for instance by performing different groupings in the semiparametric kernels built by the algorithm. Furthermore, this method is directly extensible to multiclass problems: we believe that exploiting this information could lead to feature extraction methods that find across-category common features, and which may help in many cases to obtain interpretability in complex classification tasks.

References

[1] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont MA, 1999.

[2] C. L. Blake, and C. J. Merz, UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science, 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[3] C. J. C. Burges, "A tutorial on Support vector machines for pattern recognition," *Data Mining and knowledge discovery*, vol. 2, pp. 121-167, 1998.



(a) (b) (c) (d)

Figure 3: Projections of input patterns onto feature space (ϕ_1, ϕ_2) for Image (a), Waveform (b), Diabetes (c) and Kwok (d) datasets. Patterns of both classes are plotted using different symbols.

[4] T. Jaakkola, M. Diekhans and D. Haussler, "Using the Fisher kernel method to detect remote protein homologies," in *Proc. of the Seventh International Conference on Intelligent Systems for Molecular Biology*, 1999.

[5] J. T. Kwok, "Moderating the Output of Support Support Vector Machine Classifiers," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1018-1031, 1999.

[6] H. Lodhi, J. Shawe-Taylor, N. Cristianini and C.J.C.H. Watkins "Text Classification using String Kernels," in *Proc. Neural Information Processing Systems*, pp. 563-569, 2000.

[7] A. Navia-Vázquez, F. Pérez-Cruz, A. Artés-Rodríguez and A. R. Figueiras-Vidal, "Weighted Least Squares Training of Support Vector Classifiers leading to Compact and Adaptive Schemes," *IEEE Transactions on Neural Networks*, vol 12, no 5, pp. 1047-1059, 2001.

[8] E. Parrado-Hernández, I. Mora-Jiménez and A. Navia-Vázquez, "Growing Support Vector Classifiers via Architecture Boosting," in *Proc. of Learning'00*, 2000.

[9] E. Parrado-Hernández, I. Mora-Jiménez, J. Arenas-García, A. R. Figueiras-Vidal and A. Navia-Vázquez, "Growing Support Vector Classifiers with controlled complexity," submitted to *Pattern Recognition*.

[10] F. Pérez-Cruz, C. Bousoño-Calzón and A. Artés-Rodríguez, "Convergence of the IRWLS Procedure to the Support Vector Machine Solution," submitted to *IEEE Trans. on Neural Networks*.

[11] B. D. Ripley, "Neural networks and related methos for classification (with discussion)," *J. Roy. Statist. Soc. Series B*, vol. 56, pp. 409-456, 1994.

[12] A. Ruiz and P. E. López-de-Teruel, "Nonlinear Kernel-Based Statitiscal Pattern Analysis," *IEEE Transactions on Neural Networks*, vol.12, no 1, pp. 16-32, 2001.

[13] B. Schölkopf, "Nonlinear Component Analysis in a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[14] 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.

[15] B. Schölkopf and A.J. Smola, *Learning with kernels*, MIT Press, Cambridge MA, 2002.

[16] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

[17] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pp. 281-287, MIT Press, Cambridge, MA, 1997.