Selection of Basis Functions Guided by the L2 Soft Margin

Ignacio Barrio, Enrique Romero, and Lluís Belanche

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. Support Vector Machines (SVMs) for classification tasks produce sparse models by maximizing the margin. Two limitations of this technique are considered in this work: firstly, the number of support vectors can be large and, secondly, the model requires the use of (Mercer) kernel functions. Recently, some works have proposed to maximize the margin while controlling the sparsity. These works also require the use of kernels. We propose a search process to select a subset of basis functions that maximize the margin without the requirement of being kernel functions. The sparsity of the model can be explicitly controlled. Experimental results show that accuracy close to SVMs can be achieved with much higher sparsity. Further, given the same level of sparsity, more powerful search strategies tend to obtain better generalization rates than simpler ones.

1 Introduction

Margin maximization has proven a good approach for classifi cation tasks, and the Support Vector Machine (SVM) [1] is a state-of-the-art technique that shows very good performance. Given a training set $\{x_i, t_i\}_{i=1}^N$ where $x_i \in \mathbb{R}^D$ and $t_i \in \{+1, -1\}$, linear SVMs fi nd a hyperplane that maximizes the margin in the input space. Nonlinearities can be added to these linear SVMs by mapping the input data with a set of basis functions into a feature space, with as many dimensions as basis functions, where the plane is found. The most usual way to add nonlinearities is by means of kernel functions.

An advantage of using (Mercer) kernels in SVMs is that an induced set of basis functions can be used without the need to explicitly consider them. The kernel is usually designed to serve as a good similarity function for a given problem [2]. This advantage sometimes involves a drawback, since the most natural similarity function for a given problem may not satisfy Mercer's condition, therefore designing a specifi c kernel, besides requiring some expertise, may reduce the quality of the function [3].

Another drawback of SVMs is that, although the solution is sparse in the number of support vectors, this number tends to grow with the number of data [4]. This is problematic for applications requiring high classifi cation speed, since the cost of each prediction is proportional to the number of support vectors.

A number of methods have been proposed to overcome these drawbacks while maintaining identical functional form to SVMs [5–7]. Basically, these methods consider a fi nite set of basis functions (features) and they select a subset of those basis functions so that the solution is sparse in the feature space. These works have shown that very good performance can be achieved without the requirement of kernel functions. However these methods do not maximize the Euclidean margin.

In order to maximize the margin and find a sparser representation than SVMs, a new constraint can be added to the SVM training problem [8], so that the number of *expansion vectors*[1] is preset by the user. This approach is similar to finding a "constrained" feature space where the margin of the training data is maximized. Another method [9] has been proposed to select the expansion vectors with forward selection while maximizing the L_2 soft margin (penalizing the square of slack variables). Both works require the use of kernels.

The objective of this work is to find sparse models that maximize the margin whithout the requirement of using kernels. To overcome the requirement of kernels, unlike classical SVMs, we will consider an explicit finite set of basis functions (features). Then we could compute the parameters of a linear SVM where the inputs to the SVM are the inputs of the problem non-linearly mapped into the feature space, but the solution of this linear SVM would not be sparse.

In order to find sparse models, similar to [9], we propose the use of *Search Strategies Guided by the L_2 soft margin* (SSGL2) to select a subset from a set of candidate basis functions without the requirement of using kernels. Although the optimization problem posed is minimized when the whole set of basis functions is included, we seek a tradeoff between margin maximization and subset size. The solution is sparse in the feature space. In fact, we are trying to solve a feature selection problem where each feature is the output of a basis function. The search strategies used require the addition or removal of one basis function at a time. Based on an algorithm recently proposed to train linear L_2 soft margin SVMs [10], we can make use of the current solution to efficiently select the next basis function to add or remove.

Experiments using kernel functions show that, with a much reduced subset of basis functions, the model can be competitive and perform very similar to a linear SVM with the whole set of basis functions as features and to classical nonlinear SVMs. This way, the cost of predictions is much lower. Regarding the search strategies, more complex ones require fewer basis functions to achieve good performance than simpler ones.

This work is organized as follows. In section 2 we mention some alternatives to SVMs that tackle the aforementioned drawbacks. In section 3, we briefly review a method to solve linear L_2 soft margin SVMs. In section 4 we enumerate two popular search strategies. In section 5 we propose the SSGL2 for the selection of basis functions. An experimental study is carried out in section 6, comparing the SSGL2 to other methods. Finally, we draw some conclusions in section 7.

2 Related Methods

A number of methods have been proposed to select a subset of basis functions from a set of candidates without the need of being kernels. These candidates are usually, but not necessarily, centered at the input data. Some of these methods are the Relevance Vector Machine (RVM) [5], 1-norm SVMs (1-NSVM) [7] or Kernel Matching Pursuits (KMP)

[1] *Expansion vectors* is the term used in [8] to refer to the vectors associated with each kernel evaluation in the model.

[6]. On the one hand, RVM and 1-NSVM consider the whole set of basis functions and minimize a cost function that makes most of the weights become zero. On the other hand, KMP follows a search process to select a subset of basis functions that minimize some cost function (usually the sum-of-squares error). None of these methods maximizes the Euclidean margin.

Two methods have recently been proposed to maximize the margin using a reduced number of expansion vectors. Both methods require the use of kernels. The first method [8] adds a new constraint to the SVM problem so that the number of expansion vectors is preset by the user. The new problem is non-convex and a local minimum is found with a gradient based algorithm.

The second method [9] uses forward selection to select a subset of vectors that maximize the L_2 soft margin or, equivalently, optimize

$$\min_{\beta} f(\beta) = \frac{\lambda}{2}\beta^T K \beta + \frac{1}{2}\sum_{i\in I(\beta)} (y_i(\beta) - t_i)^2, \tag{1}$$

where K is the $N \times N$ kernel matrix and $y_i(\beta) = \sum_{j=1}^{N} \beta_j K(x_i, x_j)$ and $I(\beta) = \{i : t_i y_i(\beta) < 1\}$ is the active set. The bias term is omitted for simplicity. The selection technique is based on previous work on optimizing linear L_2 soft margin SVMs [10].

3 Optimizing Linear L_2 Soft Margin SVMs

Linear SVMs find a hyperplane that maximizes the margin in the input space $\mathbb{R}^D$. The objective is to find the parameters $\mathrm{w} = (\omega_1, \omega_2, .., \omega_D)^T$ that solve the problem:[2]

$$\min_{\mathrm{w}} \frac{1}{2}\|\mathrm{w}\|^2 + \frac{C}{2}\sum_{i=1}^{N} \xi_i^2$$
$$\text{subject to} \quad t_i(\mathrm{w}^T x_i) \geq 1 - \xi_i \quad \forall i \in \{1, .., N\}, \tag{2}$$

where C is a regularization parameter and $x_i = (x_{i1}, x_{i2}, .., x_{iD})^T$ are the input vectors. The output function is $y(x) = \mathrm{w}^T x$.

This problem has an equivalent formulation as

$$\min_{\mathrm{w}} f(\mathrm{w}) = \frac{\lambda}{2}\mathrm{w}^T \mathrm{w} + \frac{1}{2}\sum_{i\in I(\mathrm{w})} (\mathrm{w}^T x_i - t_i)^2, \tag{3}$$

where $\lambda = 1/C$ and $I(\mathrm{w}) = \{i : t_i \mathrm{w}^T x_i < 1\}$. The active set in the final solution, $I(\mathrm{w})$, corresponds to the support vectors. Note that f is a piecewise quadratic function strictly convex, thus it has a unique minimizer. Furthermore, f is continuously differentiable [10]. An iterative algorithm has been proposed [10] to optimize (3). Basically, each iteration k consists of two steps. In the first one, the optimal parameters $\overline{\mathrm{w}}$ for the current active set $I(\mathrm{w}_k)$ can be found as a regularized least squares solution. In the second step a line search is done, following $\mathrm{w}_{k+1} = \mathrm{w}_k + \delta^*(\overline{\mathrm{w}} - \mathrm{w}_k)$, to decrease the "full" objective function f. The algorithm has been theoretically shown to converge to the solution of (2) in a finite number of steps.

[2] For the sake of simplicity we will treat the bias term as part of the parameters, w.

4 Search Strategies

A search process has four main elements: an objective function which directs the search, a search strategy that decides how to continue exploring new states, an initial state where the search starts from and a stopping criterion. Two popular search strategies are:

– **PTA(l, r):** Plus l and Take Away r [11]. At every step, l elements are added one at a time (always the one that maximizes the objective function) and then r elements are removed one at a time (always the one that, after removing it, the objective function is maximized). When $l > r$, it is an increasing method, and when $l < r$, it is a decreasing one. Note that **Forward Selection (FS)** is PTA(1,0).

– **SFFS:** Sequential Forward Floating Selection [12] was originally designed for feature selection. At every step, an element is added and then zero or more elements are removed one at a time while the value of the objective function is better than the best value achieved until this moment with the same number of elements.

5 Search Strategies Guided by the L_2 Soft Margin

If we transform the input data into a new feature space $\phi(x) = (\phi_1(x), \phi_2(x), ..\phi_m(x))^T$, the hyperplane that maximizes the margin in this new space can be found by extending (3):

$$\min_{\mathrm{w}} f(\mathrm{w}) = \frac{\lambda}{2}\mathrm{w}^T\mathrm{w} + \frac{1}{2}\sum_{i \in I(\mathrm{w})} (y_i(\mathrm{w}) - t_i)^2, \qquad (4)$$

where $\mathrm{w} = (\omega_1, \omega_2, .., \omega_m)^T$, $y_i(\mathrm{w}) = \mathrm{w}^T\phi(x_i)$ and $I(\mathrm{w}) = \{i : t_i y_i(\mathrm{w}) < 1\}$ is the active set. Given a set of M candidate basis functions, we could tackle the classifi cation problem by including in the model all of them (*i.e.* $m = M$) and solving (4). Since the model with the whole set of candidate basis functions has more flexibility than all the other subsets, the loss function (4) is minimized further. Although that solution can obtain good generalization rates, the cost of the predictions is very high when M is large. In order to reduce the cost, we could select a random subset of $m < M$ basis functions, obtanining a Reduced SVM [13, 14], but that subset could be very suboptimal.

In order to obtain good generalization at a low prediction cost, we intend to fulfi ll a tradeoff between the maximization of the L_2 soft margin and the number of basis functions. As a practical approach to achieve this goal we propose to use a search process, following some search strategy (section 4) guided by the maximization of the L_2 soft margin or, equivalently, the minimization of (4). We will refer to these methods as *Search Strategies Guided by the L_2 soft margin* (SSGL2). More powerful search strategies like SFFS should reduce (4) more than simpler ones like FS. Therefore, a better generalization is expected. However, they will also require more additions and removals, so the learning stage will be slower.

The L_1 soft margin is more commonly used with SVMs than the L_2 soft margin. One of the reasons for the common use of the L_1 soft margin is that it produces sparser solutions in the number of support vectors. This reason does not affect this work, because we are working in the feature space and the sparsity in the number of features is explicitly controlled during the search process. We propose to maximize the L_2 soft margin for computational convenience.

Note that the SSGL2 are guided by (4), and not by (1) as proposed in [9]. The main difference is that the regularization term is simpler in (4) and only depends on the values of the coeffi cients. This allows the use of non-kernel basis functions.

5.1 Implementation

The search strategies described in section 4 only require the addition and removal of elements (the "best" element at every step). Following is the pseudocode for addition and removal of basis functions with SSGL2 based on the algorithm proposed in [10] and described in section 3.

```
AddBestBasisFunction (a subset of $m$ basis functions $\phi$, the parameters $\mathrm{w}$,
  a set of candidate basis functions $\{\varphi_i\}$)
 1. for each candidate basis function $\varphi_i$ not in $\phi$ do
 2.    set $\phi_+$ the subset obtained by adding $\varphi_i$ to $\phi$ and compute the parameters $\overline{\mathrm{w}}$
         for the current active set $I(\mathrm{w})$ as a regularized least squares solution
 3.    $\mathrm{w}_+$ := LineSearch($\phi_+, \mathrm{w}, \overline{\mathrm{w}}$)
 4.    compute (4) for $\phi_+, \mathrm{w}_+$
 5. end for
 6. set $\phi$, $\mathrm{w}$ the subset and parameters obtained by adding to $\phi$ the $\varphi_i$ that
      minimizes (4) in the previous loop
 7. w:=TrainSVMKeerthiDeCoste($\phi$, $\mathrm{w}$)
 8. return ($\phi, \mathrm{w}$)
end AddBestBasisFunction

RemoveWorstBasisFunction (a subset of $m$ basis functions $\phi$, the parameters $\mathrm{w}$)
 9. for each basis_function $\phi_i$ in $\phi$
10.    set $\phi_-$ the subset obtained by removing $\phi_i$ from $\phi$ and compute the parameters $\overline{\mathrm{w}}$
         that minimize (4) for the current $I(\mathrm{w})$ as a regularized least squares solution
11.    set $\mathrm{w}'$ the parameters obtained by removing $\omega_i$ from $\mathrm{w}$
12.    $\mathrm{w}_-$ := LineSearch($\phi_-, \mathrm{w}', \overline{\mathrm{w}}$)
13.    compute (4) for $\phi_-, \mathrm{w}_-$
14. end for
15. set $\phi$, $\mathrm{w}$ the subset and parameters obtained by removing from $\phi$ the $\phi_i$
      that minimizes (4) in the previous loop
16. w:=TrainSVMKeerthiDeCoste($\phi$, $\mathrm{w}$)
17. return ($\phi, \mathrm{w}$)
end RemoveWorstBasisFunction

TrainSVMKeerthiDeCoste (a subset of basis functions $\phi$,
  the parameters $\mathrm{w}$)
18. while not convergence do
19.    compute the parameters $\overline{\mathrm{w}}$ that minimize (4) for the current $I(\mathrm{w})$
20.    w:=LineSearch($\phi, \mathrm{w}, \overline{\mathrm{w}}$)
21. end while
22. return($\mathrm{w}$)
end TrainSVMKeerthiDeCoste
```

For each candidate basis function, for efficiency reasons, the optimal parameters are approximated (steps 2-3 or 10-12 in the pseudocode) by performing only one iteration of the algorithm in section 3: the regularized least squares stage is computed incrementally (steps 2 or 10) and the cost of the line search (steps 3 or 12) is $O(N \log N)$ [10]. After a basis function has been included, the whole algorithm in section 3 is run until convergence (steps 7 or 16).

Next we explain how to perform steps 2 and 3 incrementally. Suppose the current model has m basis functions and the active set $I(\mathrm{w})$ has N_{SV} elements and denote $d : \{1, .., N_{SV}\} \rightarrow \{1, .., N\}$ the function that given an index k in the active set returns the corresponding index in the training set. Denote Φ the $N_{SV} \times m$ design matrix with elements $\Phi_{kj} = \phi_j(x_{d(k)})$ and denote ϕ_* the column vector corresponding to a new basis function with elements $\phi_{*k} = \phi_{m+1}(x_{d(k)})$. Let $\Sigma = (\lambda I + \Phi^T \Phi)^{-1}$. Under a fixed active set, Σ^{-1} is a partitioned matrix. In that case, we can compute the parameters after adding ϕ_* (step 2) as $\overline{\mathrm{w}} = ((\mathrm{w} - \omega_* \Sigma \Phi^T \phi_*)^T, \omega_*)^T$ where $\omega_* = (\lambda + \phi_*^T \phi_* - \phi_*^T \Phi \Sigma \Phi^T \phi_*)^{-1}$. Similarly, we can compute the parameters decrementally after removing ϕ_i (step 10) as $\overline{\mathrm{w}} = \mathrm{w} - \omega_i \Sigma_{ii}^{-1} \Sigma_i$. The parameter $\overline{\omega}_i$, which is now zero, should be removed.

The cost of an addition is $O(N_{SV} \times m \times M + N \times \log N \times M)$, where M is the number of candidate basis functions. The cost of a removal is $O(N_{SV} \times m^2 + N \times \log N \times m)$ —note that we have to compute (4) for each candidate removal (step 13 in the pseudocode).

5.2 Stopping Criteria

The addition of more basis functions, when using the appropiate regularization parameter, is usually beneficial for the accuracy of the model, but some heuristic stopping criterion should be used to decide when to stop the search.

A first stopping criterion comes when the requirements of memory are strict and precise: the size of the subset is fixed beforehand, and when the model first reaches that number of basis functions, the process is stopped.

However, in many cases, the user will prefer the method to automatically select the number of basis functions. For those cases, we propose this second stopping criterion: let F_n be the error (4) of the best model found with n basis function; given some k, ϵ, if the current model has $m + k$ basis functions and $F_m - F_{m+k} < \epsilon N$ the process is stopped. By modifying k and ϵ one can roughly control the size of the subset.

6 Experimental Study

The goal of the following experiments is threefold: to observe the size of the subsets required to achieve good results, to confirm that more powerful SSGL2 produce better solutions than simpler ones and to compare the performance of SSGL2 to other related methods.

Although the use of kernels is not necessary in the proposed model, in order to compare the SSGL2 to the SVM, $M = N$ candidate Radial Basis Functions (RBF) centered at the training set input vectors were considered. We used $\varphi_i(x) = \exp(-\gamma \|x - x_i\|^2)$. Furthermore, a bias candidate, $\varphi_0(x) = 1$, was used.

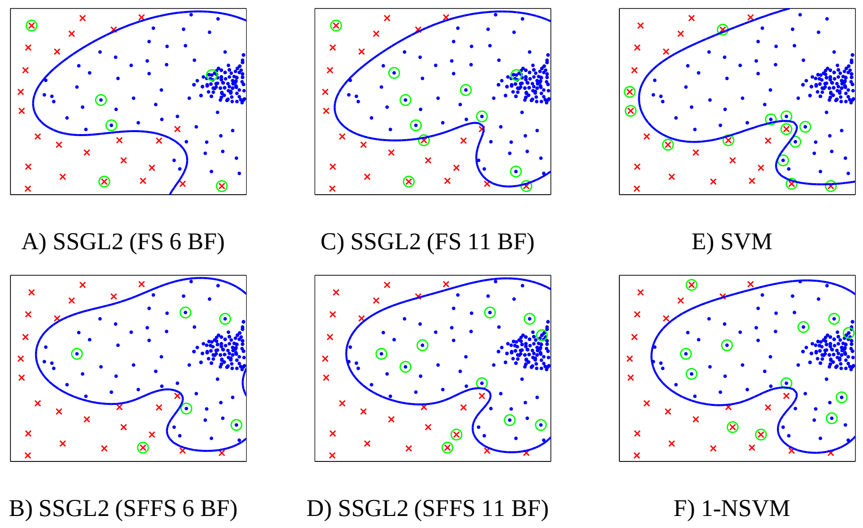

Fig. 1. Basis functions selected by different methods on the toy problem.

6.1 Toy Example

In a first experiment, we created a toy data set with two input dimensions and some nonlinearities. C was set to a high value: 10000.

Figure 1 shows the centers of the basis functions selected in the solutions of different methods. The decision boundary generated is also shown. Figures 1A to 1D correspond to SSGL2, concretely FS and SFFS. FS required 11 basis functions to classify all the data correctly, while SFFS required only 6 (both solutions are shown in the figure). For illustrative purposes, Figure 1E shows the solution of a L_1 soft margin SVM using Gaussian kernels (the selected basis functions are the support vectors), and Figure 1F shows the solution of a 1-NSVM. The classical SVM selects points that are close to the decission surface, but that is not the case for the SSGL2 and 1-NSVM.

6.2 Benchmark Comparisons

We used 12 classification benchmark problems, provided by G. Rätsch and available at http://ida.first.fraunhofer.de/projects/bench. These data sets are split in 100 training/test partitions. The results reported in this work show averages over the first 10 partitions.

Settings The input vectors were linearly scaled to $[-1, +1]$. The RBF width was the same for all the methods: we set $\gamma = (0.3D)^{-1}$, following [15], where D is the dimension of the input vectors.

We used the second stopping criterion described in section 5.2. We set $k = 10$ and $\epsilon = 0.005$. These values were chosen after some preliminary experiments. Furthermore, if the model reached 100 basis functions, the process was also stopped.

Table 1. For each data set: average accuracy obtained in the test sets (upper row) and number of basis functions of the model (lower row).

Data set	N	D	FS	PTA(2,1)	SFFS	ABF	SVM	1-NSVM	SpSVM-2
Banana	400	2	89.0	89.0	89.0	89.1	89.0	88.7	88.9
			29.3	27.2	22.0	400	99.3	18.8	19.8
Breast Cancer	200	9	72.9	72.5	72.4	71.8	70.7	69.7	71.9
			43.6	44.3	42.6	200	130	43.6	7.6
Diabetis	468	8	76.7	76.9	76.8	76.5	75.2	75.8	75.8
			32.3	28.5	25.0	468	267	31.7	18.3
Flare-Solar	666	9	66.5	66.3	66.2	65.4	66.5	66.3	66.6
			24.6	24.2	21.6	666	494	13.9	11.6
German	700	20	76.3	76.8	76.7	76.8	76.3	76.5	76.2
			52.2	46.8	44.1	700	459	108	43.6
Heart	170	13	79.1	79.6	79.0	79.0	79.8	78.7	80.0
			64.4	62.8	56.6	170	116	39.9	22.6
Image	1300	18	96.5	96.6	96.3	96.7	97.2	96.6	96.6
			65.9	55.9	35.6	1300	147	71.7	96
Ringnorm	400	20	97.8	97.7	97.6	98.0	97.5	97.8	97.7
			16.2	15.9	15.1	400	86.7	18.7	15.1
Titanic	150	3	77.8	77.8	77.8	77.8	77.8	77.9	77.3
			23.4	22.7	21.4	150	74.6	8.8	5.3
Waveform	400	21	89.8	89.6	89.5	89.6	89.6	89.1	89.4
			38.6	41.3	37.3	400	134	39.1	15.1
Splice	1000	60	86.8	86.6	86.4	88.6	88.9	85.1	85.1
			99.6	95.8	90.6	1000	841	540	86.1
Twonorm	400	20	97.2	97.1	97.1	97.2	97.3	97.0	96.9
			40.8	43.9	38.9	400	155	15.9	11.6

In order to choose the regularizer parameter, C, a 5-fold cross-validation process was performed on each training set partition. Values for C were tried ranging from 2^{-4} to 2^{14} multiplying by 2^2 (that is, $\{2^{-4}, 2^{-2}, 2^0, .., 2^{12}, 2^{14}\}$).

Models We run three SSGL2 (FS, PTA(2,1) and SFFS) and compared them to a linear (in the explicit feature space) L_2 soft margin SVM, where the features ϕ_i are all the candidate basis functions (ABF). In other words, ABF is a model minimizing (4) with $m = N$ (plus the bias term). We also compared them to classical L_1 soft margin SVM with Gaussian kernels (labelled SVM) and to 1-NSVM.

Finally, we compared them to SVMs with reduced complexity [9] (labelled SpSVM-2). We used the code provided at http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/primal/ by O. Chapelle. The whole set of candidates was considered for addition at each iteration. The number of basis functions (up to 100) for this method was selected with 5-fold cross-validation.

Results For each task, the upper row in Table 1 shows the percentages of correctly classifi ed test data (the average over the 10 partitions) and the lower row shows the

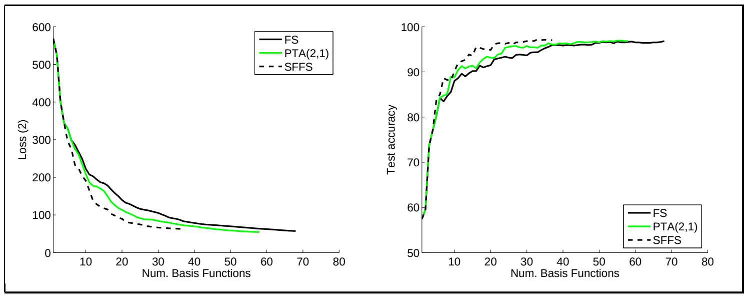

Fig. 2. The vertical axes show L_2 soft margin loss (4) (left) and test accuracy (right) on a partition of the *Image* data set. The horizontal axes show the number of basis functions of the model.

Table 2. Average number of additions plus removals required by the SSGL2.

	Ban.	Bre.	Dia.	Fla.	Ger.	Hea.	Ima.	Rin.	Tit.	Wav.	Spl.	Two.
FS	29	44	32	25	52	64	66	16	23	39	100	41
PTA(2,1)	82	133	86	73	140	188	168	48	68	124	287	132
SFFS	269	272	180	118	412	456	540	83	115	264	1066	168

number of basis functions used by each method. The number of training data N and the number of input variables D for each task are also shown. The SSGL2 obtained, in most cases, very similar accuracy than ABF while using only a subset of basis functions. Their performance was also very similar to SVM, 1-NSVM and SpSVM-2. The SSGL2 found more compact models than SVM and similar to 1-NSVM and SpSVM-2. Comparing the search strategies of SSGL2, SFFS usually required a lower number of basis functions to obtain similar accuracy than FS and PTA(2,1).

Figure 2 shows the performance of the models found by the SSGL2 on the first partition of the *Image* data set. The plot on the left shows the L_2 soft margin loss (4) in the training set, while the plot on the right shows the accuracy of the models on the test set. Similar results were obtained for the rest of the data sets. Given the same number of basis functions, the models found by SFFS performed better than those found by FS. Again, we can see that the number of basis functions of the final solution is lower for SFFS than for FS. In contrast, the number of additions and removals required by SFFS was much higher than by FS (see Table 2). PTA(2,1) was in an intermediate position.

7 Conclusions and Future Work

A method has been described to select a subset of basis functions from a set of candidates by means of a search process guided by the L_2 soft margin. Being an explicit search, we can explicitly control the sparsity of the solution.

In the experiments, the SSGL2 found compact and competitive models. SFFS found very good subsets but it required a high number of operations. PTA(2,1) and FS required much less operations but their subsets were not so good. Choosing the search strategy implies a tradeoff between accuracy and computational cost. In orther to satisfy this tradeoff, other search strategies may be considered.

The SSGL2 can be extended in two ways that kernel methods cannot (or at least not so easily). First, any similarity function can be used without the restriction to be a kernel function. Second, a set of candidate basis functions (and a model) can contain different sorts of basis functions.

Acknowledgments

This work was supported by the Consejo Interministerial de Ciencia y Tecnología (CICYT), under projects CGL2004-04702-C02-02 and TIN2006-08114.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag (1995)
2. B. Schölkopf, K. Tsuda, J.P.V.: Kernel methods in computational biology. MIT Press (2004)
3. Balcan, M.F., Blum, A.: On a theory of learning with similarity functions. In: Proceedings of the 23rd International Conference on Machine Learning, ACM Press (2006) 73–80
4. Steinwart, I.: Sparseness of support vector machines. Journal of Machine Learning Research **4** (2003) 1071–1105
5. Tipping, M.: Sparse Bayesian learning and the relevance vector machine. Journal of Machine Learning Research **1** (2001) 211–244
6. Vincent, P., Bengio, Y.: Kernel matching pursuit. Machine Learning **48**(1-3) (2002) 165–187
7. Bradley, P.S., Mangasarian, O.L.: Feature selection via concave minimization and support vector machines. In: 15th International Conf. on Machine Learning, Morgan Kaufmann (1998) 82–90
8. Wu, M., Schölkopf, B., Bakir, G.: Building sparse large margin classifiers. In: 22nd International Conf. on Machine learning, ACM Press (2005) 996–1003
9. Keerthi, S., Chapelle, O., DeCoste, D.: Building Support Vector Machines with Reduced Classifier Complexity. Journal of Machine Learning Research **8** (2006) 1–22
10. Keerthi, S., DeCoste, D.: A modified finite Newton method for fast solution of large scale linear SVMs. Journal of Machine Learning Research **6** (2005) 341–361
11. Kittler, J.: Feature selection and extraction. In Young, Fu, eds.: Handbook of Pattern Recognition and Image Processing. Academic Press (1986)
12. Pudil, P., Novovičov´a, J., Kittler, J.: Floating Search Methods in Feature Selection. Pattern Recognition Letters **15**(11) (1994) 1119–1125
13. Lee, Y.J., Mangasarian, O.L.: Rsvm: Reduced support vector machines. In: SIAM International Conference on Data Mining. (2001)
14. Lin, K.M., Lin, C.J.: A study on reduced support vector machines. IEEE Transactions on Neural Networks **14**(6) (2003) 1449–1559
15. Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V.: Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing **45**(11) (1997) 2758–2765