Feature Selection for Support Vector Machines by Alignment with Ideal Kernel

Neus Català and Mario Martin

Universitat Politècnica de Catalunya,
Jordi Girona, 1-3. 08034 Barcelona, Spain
ncatala@talp.upc.edu
mmartin@lsi.upc.edu

Abstract. Feature selection has several potentially beneficial uses in machine learning. Some of them are to improve the performance of the learning method by removing noisy features, to reduce the feature set in data collection, and to better understand the data. In this paper we present how to use empirical alignment, a well known measure for the fitness of kernels to data labels, to perform feature selection for support vector machines. We show that this measure improves the results obtained with other widely used measures for feature selection (like information gain or correlation) in linearly separable problems. We also show how alignment can be successfully used to select relevant features in non-linearly separable problems when using support vector machines.

Key words: Feature selection, empirical alignment, kernel methods

1 Introduction

Feature selection has several potentially beneficial uses in machine learning. Some of them are to improve the learning performance by removing noisy or redundant features, to reduce the feature set in data collection or when using the learned classifier, and to better understand the data.

According to [1], three general methods are used to perform feature selection: *wrapper*, *filter* and *embedded* methods. *Wrapper* methods try to find a suitable subset of features by testing them after learning. The high computational cost of these methods (due to several runs of the learning algorithm) makes them inapplicable to large data sets. *Filter* methods consist in ranking the set of features by using a relevance measure and selecting the best ones. This process is done before learning and they are more efficient than wrapper methods. Unfortunately, most filter methods only are able to detect linear correlations of features with data labels and are unable to find a suitable set of features for non linearly separable problems. Finally, *embedded* methods are methods that take profit of some features of the learning algorithm in order to simultaneously determine features and classifier during the training process.

In this paper we focus on the use of feature selection for support vector machines (SVMs) which are popular and state of the art methods for learning classifiers, but instead of modifying the SVM algorithm as some embedded methods propose, we will use the *empirical kernel alignment* [2] measure as a filter measure to help the selection of relevant features. We will present three procedures that use the alignment measure to perform feature selection. The first one uses alignment to rank features and select the top ones. The second procedure incrementally adds features that produce an increase in alignment. These two procedures are described in section 3. In section 4 we show how these procedures are successfully applied to linearly separable problems. The third procedure, described in section 5, implements a decremental approach that removes features in order to increase the alignment. We will show in section 5 that this procedure is able to perform feature selection for nonlinear problems.

2 Previous work

One way to do feature selection for SVMs is selecting features by using filter measures. Some empirical studies [3] [4], specially in natural language applications, show that among the best filter measures are *Information Gain* and *Bi-Normal Separation*. However, these studies are limited to SVMs using linear kernels because these measures are only able to capture linear relations with labels.

On the other hand, several embedded methods have been recently proposed for feature selection in the SVM framework. For instance, Weston *et al.* propose in [5] to find via gradient descent those features which minimize the leave-one-out error bounds that depend on the radius which includes all vectors and the margin of the learned SVM. Guyon *et al.* [6] present a method named *Recursive Feature Elimination* (RFE) that from the whole set of features and the learned SVM, removes features that do not decrease the margin of the learned SVM. Finally, other methods [7] and [8] propose different approximations to SVMs using the l_0 norm. Minimizing this norm implies to minimize the number of weights different from zero (i.e. minimize the number of features used while learning).

3 Feature Selection by using Kernel Alignment

In the following, assume that we have a sample $(x_i, \dots x_n)$ of data with labels $(y_1 \dots y_n)'$. Each label $y_i \in \{+1, -1\}$. SVMs find the maximum margin hyperplane that separates data with different labels. In order to find the desired hyperplane, SVMs only need the dot product of data in the original space or in another space (by implicitly projecting the data in a new space and computing the dot product there). The function used to calculate the dot products is named the kernel function, while the squared $n \times n$ matrix with the dot product of each pair of objects in the data set is named the matrix kernel.

The ease to separate the data depends on the kernel used. Our goal will be to find a set features to build the kernel that eases the separation of data.

3.1 Kernel alignment for subsets of features

A way to measure how a kernel helps to separate the data is by comparing it with the ideal kernel $T = yy'$. The ideal kernel is a $n \times n$ matrix that by definition presents $T_{i,j} = +1$ when x_i and x_j have the same label, and -1 otherwise. To compare matrices we use the Frobenius product between squared matrices $\langle M, N \rangle_F$, defined as $\langle M, N \rangle_F = \sum_{i,j}^{n} M_{ij} N_{ij}$. Thus, empirical kernel alignment that allows us to measure the fitness of a kernel K to the set of training labels is defined as in [2]:

$$A(K, T) = \frac{\langle K, T \rangle_F}{\sqrt{\langle K, K \rangle_F \langle T, T \rangle_F}} \tag{1}$$

Alignment has a number of convenient features. It can be calculated before learning takes place, it is sharply concentrated around its expected value (and hence stable to different splits of the training data), and it is known to be related with good generalization performance.

Alignment can be used to choose the best kernel of a set of kernels [2]. We will use this 'a priori' model selection procedure to find a suitable set of features that eases the learning process. We define kernel $K_{\mathcal{F}}$ from the set of features $\mathcal{F}$ as the kernel matrix obtained by applying the selected kernel function on the data represented only with features in set $\mathcal{F}$.

The obvious way to use alignment for feature selection is to rank features by their alignment. In this way, for each feature i we define $\mathcal{F} = \{i\}$ and calculate the alignment $A(K_{\mathcal{F}}, T)$. The final set of features is built with the r top ranked features (where r is a parameter set by the user or by model selection using a validation set). We name this method *one-shot Feature Selection by Alignment* (*osFSA*). However, this method does not consider relations with other features, but only with labels. One consequence of this fact is that *osFSA* could end with a set of redundant features because the addition of one feature to the set does not consider what is already explained by other features. Another consequence is that *osFSA* is not able to build sets of features for problems where non-linear kernels are necessary (where interactions between features are crucial). In the following section we show how to overcome the first problem. In section 5 we show how to deal with non-linear kernels.

3.2 Incremental Alignment Feature Selection

A common way to do feature selection is starting from an initial set of features and adding and/or removing features following a local search procedure. An efficient (but not optimal) way to do that is by using a greedy procedure that begins with the empty set of features and iteratively adds the feature which increases the most a given target measure. We propose this method for feature selection, where the target measure is the alignment. We name this method *incremental Feature Selection by Alignment* (*iFSA*), and it can be used in conjunction with the linear kernel. It has the advantage over *osFSA* that the final set of features does not contain redundant features. A new feature is added when it increases

the alignment obtained from the current set of features (that is, it 'explains' something new not explained by already selected features).

However, this method presents a drawback. The cost of computing the kernel alignment is $O(n^2k)$, where n is the number of examples in the data set (n^2 for calculating the Frobenius product), and k is the cost of calculating the kernel value for a pair of examples (in the case of the linear kernel, this cost is the number of features m). Though this is inexpensive compared with the cost of the learning algorithm, we have to consider that alignment has to be calculated for each set of features to be tested.

Fortunately, the *iFSA* procedure we propose measures how alignment is improved when adding a new feature to the set of selected features. There exists an efficient way to compute the new kernel from the previous kernel and reducing in this way the cost k. For the linear kernel (K^1), the following equation shows how to calculate the new kernel when adding a new feature i.

$$K^1_{\mathcal{F}\cup\{i\}}(x,y) = \sum_{j\in\mathcal{F}\cup\{i\}} x_j y_j \;=\; \sum_{j\in\mathcal{F}} x_j y_j + x_i y_i \;\;=\;\; K^1_{\mathcal{F}}(x,y) + x_i y_i \qquad (2)$$

In section 5, we will explain the similar trick for non linear kernels. In this way, for the most common kernels, we can reduce to $O(n^2)$ the cost of computing the alignment $A(K_{\mathcal{F}\cup\{i\}},T)$ from $K_{\mathcal{F}}$. For large data sets this could still be a problem. Fortunately, alignment is a sharply concentrated measure [2] and could be accurately estimated from a sample of the whole training set. With these considerations in mind, the method of *incremental Feature Selection by Alignment* algorithm is detailed as follows:

Algorithm 1 Incremental Feature Selection by Alignment (*iFSA*)

1. $\mathcal{C}$ = Complete set of features
2. $\mathcal{F}= \emptyset$
3. $K^1_{\mathcal{F}} = 0$ for all x,y
4. *CurrentAlign* $= 0$
5. **repeat**
6. **for each** i **in** $\mathcal{C}$ **do**
7. *IncrAlign(i)* $= A(K^1_{\mathcal{F}\cup\{i\}},T) -$ *CurrentAlign*
8. **endfor**
9. *add* = Feature i with higher *IncrAlign(i)*
10. **if** *IncrAlign(add)* > 0 **then**
11. Calculate incrementally $K^1_{\mathcal{F}\cup\{add\}}$ from $K^1_{\mathcal{F}}$
12. $\mathcal{F}= \mathcal{F} \cup \{add\}$
13. $\mathcal{C}= \mathcal{C}\backslash\{add\}$
14. *CurrentAlign*$= A(K^1_{\mathcal{F}}, T)$
15. [Optional] Remove from $\mathcal{C}$ features i with *IncrAlign(i)* < 0
16. **endif**
17. **until** *IncrAlign(add)* ≤ 0

The algorithm looks always for the feature which increases the most the alignment. This feature is added to $\mathcal{F}$ and the procedure is repeated until no

more features remain in the set of candidate features $\mathcal{C}$ that increase the alignment. Note that after selecting one feature, it is removed from set $\mathcal{C}$ along with all features that decrease the alignment. Note also that the algorithm can be modified to add the best k features at each iteration of the outer loop, or to stop the feature selection when the best increment in alignment is below a user selected threshold.

In order to test the methods proposed, the next section details the performance of *osFSA* and *iFSA* in two problems that are usually solved by using linear kernels.

4 Evaluation on linearly separable data

For the first experiments, we selected two data sets from text categorization, where the number of features is very large. Text Categorization is the task of assigning textual documents to one or more predefined categories based on their content. Usually, each document is represented as a vector of words where each word corresponds to a feature. The value or weight of a feature symbolize how it is representative of the document content.

The first data set is the `TechTC-100`[1] data collection. The second data set is the well known benchmark on text categorization `20 Newsgroups`[2].

4.1 TechTC-100 data set

This data set consists in 100 subsets of 150 documents each one. Each subset contains examples of two different categories which have to be separated by the learning mechanism. We have selected this collection because previous work [4] shows that feature selection is useful in this data set.

The set of documents was preprocessed in the following way: a) stop words were removed from texts, b) Porter stemmer was used to reduce the number of words, c) words appearing in less than 3 documents were removed, and d) each document was represented by the *tf-idf* weighting procedure (which is the standard in text categorization).

Forman [3] has compared different metrics in text categorization and has found Information Gain (IG) and Bi-normal Separation (BNS) to be the best metrics for filter feature selection. [4] confirms the same results in the `TechTC-100` data set, finding that there are no significative differences between the two cited methods. Thus, in order to test the methods we propose, we will compare their results with the ones obtained by using IG for ranking features.

Each experiment was repeated 5 times for each set of documents, and the results were averaged. At each experiment, the set of documents was randomly partitioned in 5 subsets of the same length. One subset (validation set) was reserved for tuning the learning parameters. The other 4 subsets were used in a 4-fold cross-validation scheme: 3 subsets for training and 1 for testing the SVM learned with those parameters that give best results on the validation set.

[1] Available at `http://techtc.cs.technion.ac.il/techtc100/techtc100.html`
[2] `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20`

	Accuracy	*N.Features*
no FS	76.01	-
IG	80.05	65.43
iFSA	82.87	49.71
osFSA	80.74	64.02

Table 1. Averaged accuracy on the test set and number of features selected for the 100 problems of TechTC-100.

	no FS	*IG*	*iFSA*	*osFSA*
no FS	0	30	23	29
IG	70	0	6	33
iFSA	77	93	0	91
osFSA	71	66	7	0

Table 2. Number of problems of TechTC-100 on which the method of the row obtained a better accuracy on the test set than the method of the column.

The parameters to tune are the number of features to keep (r) and the C parameter of the SVM (the trade-off between training error and margin). Note that the first features selected by *iFSA* are those which increase the most the alignment, while the last features selected scarcely increase the alignment. Note also that a low increment in alignment can be too low to be significative (and adding those features could drop performance on the test set). Thus, the true relevant features could be only the first r features selected by *iFSA* and not all the ones selected. Since the number of possible r values is relatively high, we tested only 16 possible values. The r values tested were selected by running the *iFSA* procedure while storing the increment in alignment each feature produces, the iteration at which each feature was selected, and the final alignment obtained. Then, we chose the first features that accumulate the 85% of the final alignment (the size of this set is r_1). Later we chose the first features that 'explain' a 86% of the final alignment (the size of this set is r_2), and so on until we finally selected the set of features that explain the 100% of the final alignment (size r_16). We finally chose the C and r parameters in a standard grid model selection way on the validation set. The C values tested were $\{10^1, 10^2 \dots 10^7\}$. In the case of IG and *osFSA*, the number of features and the C value chosen were also those yielding a better accuracy on the validation set. The C and r values tested for these methods are the same ones that were tested in *iFSA*.

Results averaged for the 100 sets of documents are reported in tables 1 and 2 (where *no FS* stands for 'no feature selection performed'). These tables show that the *iFSA* procedure helps the most in increasing the accuracy for the TechTC-100 data set, returning in addition the smaller feature set. In particular, better results over *osFSA* are explained because *iFSA* does not select redundant features. Tables also show that *osFSA* is also a slightly better procedure than Information Gain for feature selection.

4.2 20 Newsgroups

The experiments described in this section show the performance of feature selection by alignment in data sets with an unbalanced proportion of positive and negative examples.

The data set selected for this experiment consists in a collection of documents from 20 usenet news groups. For each group there were collected 1000 documents.

	400 training				1000 training			
Data set	*no FS*	*IG*	*iFSA*	*osFSA*	*no FS*	*IG*	*FSA*	*osFSA*
alt.atheism	4.78	4.94	4.92	5.01	4.25	4.68	4.44	4.66
comp.graphics	4.58	5.00	4.76	4.80	4.18	4.60	4.42	4.50
comp.os.ms-windows.misc	4.33	4.58	4.54	4.57	4.04	4.59	4.40	4.49
comp.sys.ibm.pc.hardware	4.66	4.91	4.84	4.92	4.34	4.85	4.70	4.82
comp.sys.mac.hardware	5.00	4.92	4.45	4.79	3.76	4.07	3.78	3.91
comp.windows.x	5.01	4.76	4.37	4.72	3.86	4.13	3.81	4.15
misc.forsale	4.98	4.64	4.32	4.59	4.07	4.25	4.09	4.15
rec.autos	5.00	4.26	3.93	4.00	3.31	3.61	3.54	3.59
rec.motorcycles	5.01	2.45	2.14	2.30	2.73	2.30	2.21	2.28
rec.sport.baseball	5.01	4.17	3.82	4.09	2.87	3.36	3.24	3.38
rec.sport.hockey	5.01	3.46	3.11	3.47	2.04	2.49	2.16	2.53
sci.crypt	5.00	3.22	2.90	3.11	2.41	2.51	2.14	2.36
sci.electronics	5.01	5.07	4.98	5.09	4.43	4.69	4.56	4.84
sci.med	5.01	4.47	4.30	4.45	3.61	3.74	3.78	3.72
sci.space	5.02	4.23	3.98	4.31	2.93	3.40	3.20	3.48
soc.religion.christian	5.00	4.68	4.25	4.77	3.39	4.22	3.97	4.28
talk.politics.guns	5.01	4.80	4.57	4.65	3.61	4.06	3.95	4.14
talk.politics.mideast	5.02	3.56	3.24	3.57	2.25	2.39	2.25	2.35
talk.politics.misc	5.01	4.93	4.89	4.93	4.29	4.75	4.60	4.69
talk.religion.misc	5.02	5.20	5.01	5.06	4.88	5.14	5.01	5.10
Average	**4.92**	**4.41**	**4.17**	**4.36**	**3.56**	**3.89**	**3.71**	**3.87**

Table 3. Error in test set for the 20 Newsgroups benchmark data set.

Some references [9] [3] [4] show that feature selection does not work in this data set in terms of increasing the accuracy.

The experiment consisted in learning 20 support vector machines, one for each newsgroup, that separates the newsgroup from all the others. Note that in this case, for each positive example we have 19 negative examples. In order to deal with unbalanced data sets, we used a modified version of alignment [10] in which labels to calculate the ideal kernel are changed from $+1$ to $+1/m$ and from -1 to $-1/n$, where m and n are the number of positive and negative examples respectively. The learning of the SVM is done with the original set of labels.

We performed experiments on two different sizes of the training set: 400 examples (20 positive and 380 negative), and 1000 examples (50 positive and 950 negative). For each experiment we randomly selected 8 disjoint sets of examples as training sets. Selection of the parameters for the SVM were obtained by following the procedure described for the Technion data set. The averaged errors on the test set are shown in table 3. As in the previous section, we also report for comparison the performance of IG on the same data set.

Results show again that *iFSA* widely outperforms IG for the two sizes of training sets. Results also show that *osFSA* has a slightly better performance than IG. Finally, experiments show that FS helps to improve accuracy in the smaller training set.

5 Evaluation on Non Linearly separable data

The previous section shows the ability of feature selection by alignment to deal with linearly separable data. Most of the measures for ranking features (f.i. correlation and information gain), only consider linear relations between features. One of the main advantages of feature selection by alignment is that it is able to deal also with problems where features have a non linear relation with labels.

However, the incremental algorithms we have described in section 3 are not able to learn such set of features. Assume a typical nonlinear interaction between features, for instance, that feature i is relevant for the problem only when feature j is also present. In the case that neither i nor j were in the initial set of features, the algorithm will never add them. It will not add feature j because it only increases the alignment when i is present (and it is not the case). It will not add feature i for the symmetrical reason.

Note that feature selection in nonlinear cases could be solved by explicitly projecting the data in the feature space induced by the kernel. However, the dimension of this space does not grow linearly with the number of input features. Moreover, for some kernels (like RBF kernels) the explicit travel to the feature space is not possible.

Fortunately, there is an efficient way of using alignment for non-linear kernels to select relevant features. Instead of starting from the empty set of features and adding new features while alignment is increased, we will start from the whole set of features and iteratively remove features when that increases the alignment. Note that features with beneficial nonlinear relations with other features will be preserved. In the case described above, j will not be removed, because i is present and it would decrease the alignment. Feature i will not be removed for the symmetrical reason. But features with no relevant relation (linear or not linear) with labels will be removed. We name this procedure *decremental Feature Selection by Alignment* (*dFSA*) and it is described in detail in the following:

Algorithm 2 Decremental Feature Selection by Alignment (*dFSA*)

1. $\mathcal{F}$= Complete set of features
2. Calculate $K^1_{\mathcal{F}}$ for all x,y
3. *CurrentAlign* = $A(K_{\mathcal{F}},T)$
4. **repeat**
5. **for each** i **in** $\mathcal{F}$ **do**
6. *DecrAlign(i)* = *CurrentAlign* - $A(K_{\mathcal{F}\setminus\{i\}},T)$
7. **endfor**
8. *out* = Feature i with lower *DecrAlign(i)*
9. **if** *DecrAlign(out)* ≤ 0 **then**
10. Calculate $K^1_{\mathcal{F}\setminus\{out\}}$ from $K^1_{\mathcal{F}}$
11. $\mathcal{F}$= $\mathcal{F}\setminus\{out\}$
12. *CurrentAlign*= $A(K_{\mathcal{F}},T)$
13. [Optional] Remove from $\mathcal{F}$ features i with *DecrAlign(i)* > 0
14. **endif**
15. **until** *DecrAlign(out)* > 0

Training	*Recall*	*N. of feats*	*Exact Sol.*
50	99.40	6.38	38.60
100	100.00	2.40	95.90
150	100.00	2.00	100.00

Table 4. First experiment on non linear data. See details in text.

As in the case of the incremental algorithm, we can speed up the computation of the kernel. The linear kernel when one feature is removed is calculated as follows:

$$K^1_{\mathcal{F}\setminus\{i\}}(x,y) = K^1_{\mathcal{F}}(x,y) - x_i y_i \tag{3}$$

This efficient computation is useful for calculating polynomial kernels of degree d when removing one feature.

$$K^d_{\mathcal{F}\setminus\{i\}}(x,y) = (1 + K^1_{\mathcal{F}}(x,y) - x_i y_i)^d \tag{4}$$

Or in the case of using RBF kernels, storing matrix d:

$$d_{\mathcal{F}}(x,y) = \sum_{i\in\mathcal{F}} (x_i - y_i)^2 \tag{5}$$

and thus,

$$K_{\mathcal{F}\setminus\{i\}}(x,y) = e^{-\gamma(d_{\mathcal{F}}(x,y) - (x_i - y_i)^2)} \tag{6}$$

In order to show the ability of the proposed method to deal with nonlinear problems, we selected two artificial data sets described in [5] and [8].

5.1 Weston *et al.* dataset

In the first data set, 52 variables are available and only the first two are relevant. A precise description of these synthetic data can be found in [5]. The set has the same number of positive and negative examples. If the label of the example is -1, then $\{x_1, x_2\}$ are drawn with equal probability from $N(\{-\frac{3}{4}, -3\}, I)$ or $N(\{\frac{3}{4}, 3\}, I)$. If the label is +1, $\{x_1, x_2\}$ are drawn with equal probability from $N(\{3, -3\}, I)$ or $N(\{-3, 3\}, I)$. Features x_3 to x_{52} are noise $N(0, 20)$. The data set contains 10,000 examples generated in this way. This problem can be solved by using a quadratic polynomial kernel, but not by a linear kernel.

The *dFSA* algorithm has been tested on different sizes of the training set which has been normalized to get zero mean and unit standard deviation. The examples not used for training are used for testing and are normalized according to the same normalization parameters. The algorithm stops automatically when no further features can be removed and the resulting set of features is examined. No validation set was required. We collected the number of times that the relevant features were selected (Recall), the number of features in the final set, and the percentage of cases in which the outcome was exactly the set of relevant features $\{x_1, x_2\}$. We performed 500 experiments for each size of the training set. The averaged the results are shown in table 4.

	$f_1(1)$		$f_2(3)$		Sp 0.97 (10)		Sp 0.95 (14)		Sp 0.00 (20)	
Train.	*Recall*	*N.feats*	*Recall*	*N.feats*	*Recall*	*N.feats*	*Recall*	*N.feats*	*Recall*	*N.feats*
25	100.00	4.46	92.67	9.29	90.87	17.42	91.87	17.93	91.05	18.21
50	100.00	1.45	98.00	7.16	92.51	16.13	93.17	17.87	98.90	19.78
75	100.00	1.12	99.07	4.02	95.10	15.62	94.64	17.55	98.15	19.63
100	100.00	1.03	100.00	3.26	95.56	14.50	94.84	16.99	98.45	19.69
200	100.00	1.03	100.00	3.03	98.49	11.54	95.37	15.82	99.85	19.97
250	100.00	1.00	100.00	3.01	99.28	10.30	97.19	15.02	99.85	19.97
300	100.00	1.00	100.00	3.00	100.00	10.06	97.88	15.07	99.80	19.96

Table 5. Recall and number of features collected for different kinds of polynomials. f_1 is polynomial x_1, f_2 is polynomial $x_1x_2 + x_3$. The other columns refer to polynomials with different degree of sparseness in coefficients. The numbers between parenthesis are the average number of relevant features for each kind of polynomial.

The table shows that the algorithm is able to remove noisy features even with small training sets. With only 50 examples, *dFSA* removes 44 from 50 noisy features. With 100 examples the procedure almost always finds the exact solution. In [5] similar results were obtained.

5.2 Sparse polynomial functions

We will use sparse quadratic polynomial functions to show the ability of *dFSA* for selecting features when using non-linear kernels. A polynomial with some zero coefficients (a sparse polynomial) defines a non-linear problem where only those features that are present in terms of the polynomial with non-zero coefficients are relevant. This experiment has been previously described in [8]. However, we make the problem somehow more difficult by extending the initial set of features to 20 instead of 10. Note that polynomial kernels of degree 2 with an input space of 20 dimension induce a feature space with 230 dimensions. We considered polynomials $y = sign(x_1)$, $y = sign(x_1x_2 + x_3)$, and polynomials with sparseness 0.99, 0.97, 0.95 and 0. A given percentage of sparseness implies that that percentage of terms have zero coefficients while the other terms have a random coefficient from $\{+1, -1\}$. In all these cases, the algorithm should select those features of the input space that appear in terms with non zero coefficients.

For each case, we randomly built 100 polynomials. For each polynomial we created 1.000 training points with 20 features with values randomly chosen in $N(0, 1)$. The label for each point was assigned by the sign of the evaluation of the polynomial function for the point.

Instead of selecting the best r number of features to keep as we done in the linear experiments, we measured at each iteration of *dFSA* if the removal of features decrease the accuracy in a validation set, and stop in that case. In our experiments, we used a validation set of 100 examples.

The results of our experiments in terms of the features selected are shown in table 5. Figure 1 shows the error on 500 test examples for the SVMs trained with the selected features for different sizes of the training set. Results on polynomials with very sparse coefficients show that the *dFSA* is able to largely improve

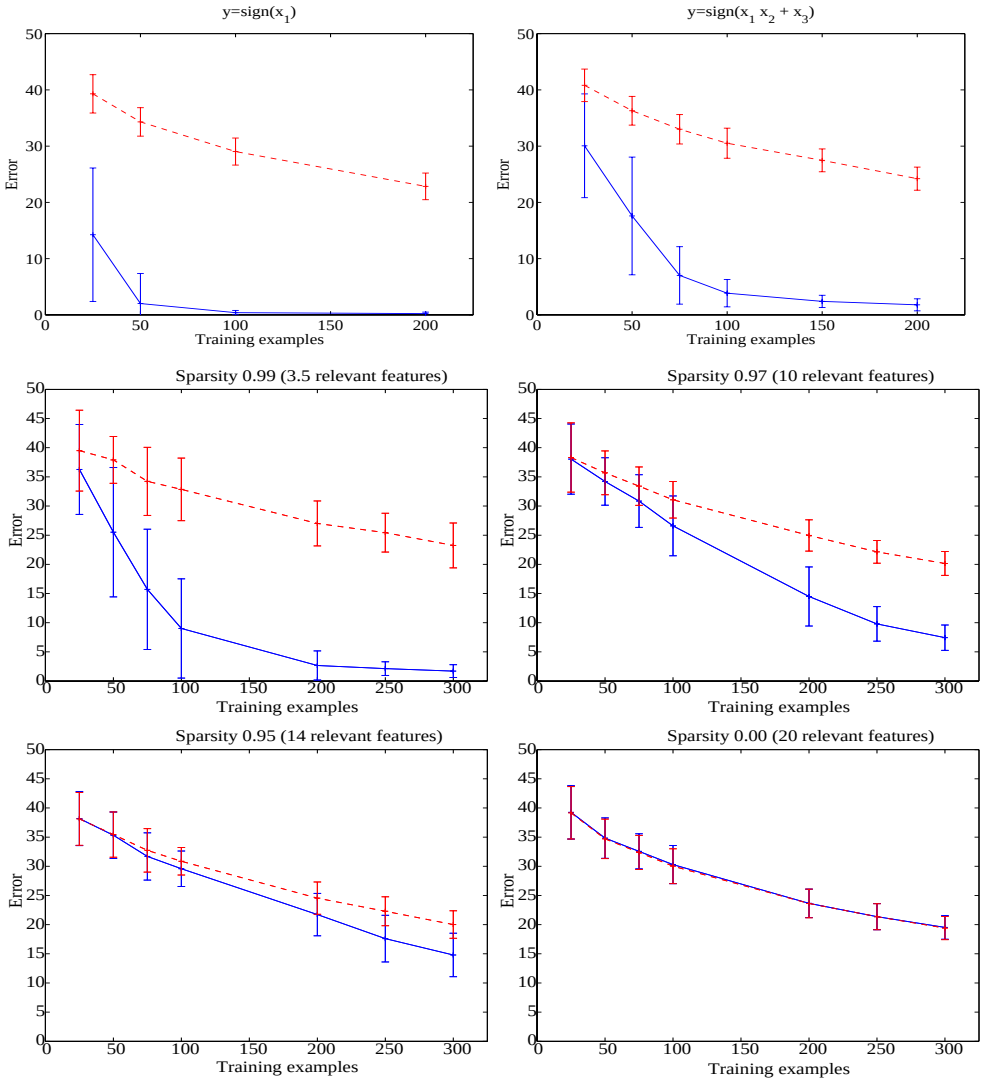

Fig. 1. Error in test set for different kinds of polynomials and different training sets. Each figure shows results for one kind of polynomial. The dashed red lines show the error of SVMs without feature selection. The blue solid lines show the error with *dFSA*.

accuracy of SVMs when there is a large amount of noise. In addition, in cases where there are no noisy features, the algorithm does not show a noticeable worse performance when compared with SVMs using all features. When using 10 features, similar results that those reported in [8] were obtained.

6 Conclusions

In this paper we have presented three procedures using alignment as a filter measure to perform feature selection for SVMs. On one hand, these procedures

have the advantages of filter methods over wrapped methods like performing the selection before learning takes place (and thus, saving computational resources). On the other hand, these methods allow for using standard implementations of SVMs instead of using modified versions of them as some embedded methods propose. The methods presented show these advantages while overcoming the usual drawback of filter measures of being unable to deal with nonlinear problems. Finally, we have shown that even in linear problems, these methods show better results than state of the art filter measures for text categorization tasks.

However, these methods present two main problems. The first one is that though theoretically, feature selection stops when no increase in alignment is achieved by adding a new feature, we have found in the experiments that stopping earlier returns better results. This fact is explained because the last features to be included present too low increment in alignment to be statistically significative. The second problem is that for very large data sets, the cost of calculating the alignment $O(n^2)$ can be too high. We plan to solve these problems by studying bounds on the convergence of alignment that [2] shows to be sharply concentrate. These bounds could help in both, to determine when increment in alignment is statistically significative, and to determine how many examples of the training set are needed to accurately estimate the alignment (instead of using the whole training set and thus reducing the cost of computing the alignment).

References

1. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature Extraction: Foundations and Applications. Springer-Verlag (2006)
2. Cristianini, N., Shawe-Taylor, J., Elisseeff, A., Kandola, J.: On Kernel-Target Alignment. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems 14, MIT Press (2002)
3. Forman, G.: An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. **3** (2003) 1289–1305
4. Gabrilovich, E., Markovitch, S.: Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5. In: Procs. of the 21st international conference on Machine learning, ACM Press (2004)
5. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., Vapnik, V.: Feature selection for SVMs. In: NIPS. (2000) 668–674
6. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning **46**(1-3) (2002) 389–422
7. Bradley, P.S., Mangasarian, O.L.: Feature Selection via Concave Minimization and Support Vector Machines. In: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann. (1998) 82–90
8. Weston, J., Elisseeff, A., Schölkopf, B., Tipping, M.: Use of the zero norm with linear models and kernel methods. J. Mach. Learn. Res. **3** (2003) 1439–1461
9. Joachims, T.: Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In: ECML '98: Proceedings of the 10th European Conference on Machine Learning, Springer-Verlag (1998) 137–142
10. Kandola, J., Shawe-Taylor, J., Cristianini, N.: On the Extensions of Kernel Alignment. Technical Report NC-TR-02-120, NeuroCOLT (2002)