Signal Processing 89 (2009) 2066–2071

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Fast Communication

Computational load reduction in decision functions using support vector machines

J. Acevedo-Rodríguez *, S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Siegmann, F. López-Ferreras

Department of Signal Theory and Communications, Escuela Politécnica Superior, Universidad de Alcalá, 28871 Alcalá de Henares, Madrid, Spain

ARTICLE INFO

Article history:
Received 7 October 2008
Received in revised form
23 January 2009
Accepted 26 March 2009
Available online 7 April 2009

Keywords:
Support vector machines (SVMs)
Kernel methods
Image recognition

ABSTRACT

A new method of reducing the computational load in decision functions provided by a support vector classification machine is studied. The method exploits the geometrical relations when the kernels used are based on distances to obtain bounds of the remaining decision function and avoids to continue calculating kernel operations when there is no chance to change the decision. The method proposed achieves savings in operations of 25–90% whilst keeping the same accuracy. Although the method is explained for support vector machines, it can be applied to any kernel binary classifier that provides a similar evaluation function.

1. Introduction

Classifiers based on kernel machines have been successfully applied in a variety of fields including engineering, information retrieval and bioinformatics [1]. These kind of classifiers use the so called kernel trick, that computes a inner product in a Hilbert space without and explicit transformation of the data. Kernel functions provide an easy way to export linear classification algorithms to non-linear ones [2]. Among these learning machines, the most popular one is the support vector machine (SVM) [3] but also it is quite extended the use of least-squares support vector machines [4] and the relevance vector machine (RVM) [5].

There are several approximations to speed-up the training phase (i.e. [6,7] or [8]). However, in real-time classification systems (RTCS), while the training of the model can be done off-line, what is really important is the computational load of the test or decision function. SVMs and RVMs have the nice property of sparseness [9], providing decision functions with a few number of base functions, what is specially useful for RTCS. Nevertheless, each kernel evaluation requires a number of operations that grow exponentially with the dimension of the patterns to be classified. Thus, in image classification problems, where the number of features is usually high, it is still necessary to reduce the computational load of the decision function even when the learning machine provides sparse decision functions. This need is even more clear in multiclass problems, where a one against all strategy is usually done and the set of binary classifiers requires a sensible increase in kernel evaluations than in the two-class case.

Several works (i.e. [10,11]) try to minimize the number of support vectors using a reduced set of them but once the evaluation function is defined these approaches do not try to save more kernel evaluations to test new samples. The proposal of this work is to reduce the number of kernel evaluations once that the evaluation function has been calculated. The method requires the kernel function used to be monotonously increasing with the distance between vectors. First, in the case of Gaussian kernel, a new method of reducing the number of operations in the test phase is proposed whilst keeping the same accuracy.

* Corresponding author.
E-mail address: javier.acevedo@uah.es (J. Acevedo-Rodríguez).

0165-1684/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2009.03.032

Then a modification of the Gaussian kernel is studied, providing tighter bounds than in the first case. Although the paper is focused on SVM, the proposed methods can be applied to any kernel classifier that provides a similar evaluation function.

2. Proposed method with Gaussian kernel

From the training phase, SVM classifiers give a decision function where new samples are classified according to

$$f(\boldsymbol{x}) = sgn\left(\sum_{i=1}^{Nsv} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{sv}_i) + b\right) \tag{1}$$

where $\boldsymbol{sv}_i$ are samples from the training set named support vectors, $y_i \in \{1, -1\}$ is the binary-class associated with every support vector, Nsv is the number of support vectors, $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_{Nsv}\}$ are the Lagrange coefficients obtained in the training phase such as $0 < \alpha_i < C$, and C is a regularization constant fixed a priori . The b parameter is the bias of the decision function and is also obtained in the training phase [12]. Finally, the function $K(\boldsymbol{x}, \boldsymbol{sv}_i)$ is the kernel function that calculates the inner product in a higher space without an explicit transformation. The most extended kernel is the Gaussian one, defined by

$$K(\boldsymbol{x}, \boldsymbol{sv}_i) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{sv}_i\|^2}{2\sigma^2}\right) \tag{2}$$

Assuming a particular support vector $\boldsymbol{sv}_1$ and a list containing all the support vectors arranged in order of increasing distance from $\boldsymbol{sv}_1$

$$L = \{sv_1, sv_2, \ldots sv_{Nsv}\}$$
$$d(\boldsymbol{sv}_q, \boldsymbol{sv}_1) \leqslant d(\boldsymbol{sv}_{q+1}, \boldsymbol{sv}_1), \quad q = 2, \ldots, Nsv - 1 \tag{3}$$

for a new pattern to be classified $\boldsymbol{x}$, the distance from the first support vector is named $d(\boldsymbol{x}, \boldsymbol{sv}_1)$. Using Cauchy–Schwarz inequality the following condition is always fulfilled:

$$d(\boldsymbol{sv}_1, \boldsymbol{sv}_q) \leqslant d(\boldsymbol{sv}_1, \boldsymbol{x}) + d(\boldsymbol{x}, \boldsymbol{sv}_q) \tag{4}$$

Thus, it is possible to obtain a lower bound on $d(\boldsymbol{x}, \boldsymbol{sv}_q)$ by means of

$$d(\boldsymbol{x}, \boldsymbol{sv}_q) \geqslant d(\boldsymbol{sv}_1, \boldsymbol{sv}_q) - d(\boldsymbol{sv}_1, \boldsymbol{x}) \tag{5}$$

Moreover, since $d(\boldsymbol{sv}_q, \boldsymbol{sv}_1) \leqslant d(\boldsymbol{sv}_{q+1}, \boldsymbol{sv}_1)$, Eq. (5) is in fact a lower bound for all the support vectors arranged after q:

$$d_n^{low}(\boldsymbol{x}, \boldsymbol{sv}_n) \geqslant d(\boldsymbol{sv}_1, \boldsymbol{sv}_q) - d(\boldsymbol{sv}_1, \boldsymbol{x}) \quad \forall n \geqslant q \tag{6}$$

On the other hand, using Cauchy–Schwarz inequality the maximum distance from the input $\boldsymbol{x}$ to each support vector satisfies

$$d_n^{upp}(\boldsymbol{x}, \boldsymbol{sv}_n) \leqslant d(\boldsymbol{sv}_1, \boldsymbol{sv}_{Nsv}) + d(\boldsymbol{sv}_1, \boldsymbol{x}) \quad \forall n \leqslant Nsv \tag{7}$$

that is, an upper bound on that distance for every support vector.

Eq. (1) can be rewritten as

$$f(\boldsymbol{x}) = f_n(\boldsymbol{x}) + f_n^+(\boldsymbol{x}) - f_n^-(\boldsymbol{x})$$
$$f_n^+(\boldsymbol{x}) = \sum_{i=1}^{Nsv_n^+} \alpha_i^{n+} \exp(-\|\boldsymbol{x} - \boldsymbol{sv}_i^{n+}\|^2/2\sigma^2)$$
$$f_n^-(\boldsymbol{x}) = \sum_{i=1}^{Nsv_n^-} \alpha_i^{n-} \exp(-\|\boldsymbol{x} - \boldsymbol{sv}_i^{n-}\|^2/2\sigma^2) \tag{8}$$

where $f_n(\boldsymbol{x})$ is the decision function evaluated as far as the n support vector of the sorted list, but also including the bias parameter b, $f_n^+(\boldsymbol{x})$ is the contribution of the remaining positive support vectors, where sv^{n+} is the set of remaining support vectors whose class is $y_i = 1$, with α_i^{n+} as multipliers associated with those remaining positive support vectors, whereas $f_n^-(\boldsymbol{x})$ is the contribution of the remaining negative support vectors. If a bound B of the remaining set of support vectors can be found, in such a way that the following condition is fulfilled:

$$\|f_n(\boldsymbol{x})\| > B > \|f_n^-(\boldsymbol{x}) - f_n^+(\boldsymbol{x})\| \tag{9}$$

there is no chance to change the sign of $f_n(\boldsymbol{x})$ and so, the test of the sample $\boldsymbol{x}$ does not need more kernel evaluations. If support vectors are arranged in a list, as has been described, and taking into account the bounds described in Eqs. (6) and (7), it can be said that if the following conditions are given:

$$f_n(\boldsymbol{x}) \geqslant 0$$
$$f_n(\boldsymbol{x}) \geqslant \exp\left(-\frac{d_n^{low^2}}{2\sigma^2}\right) \sum_{i=1}^{Nsv^{n-}} \alpha_i^{n-} - \exp\left(-\frac{d_n^{upp^2}}{2\sigma^2}\right) \sum_{i=1}^{Nsv^{n+}} \alpha_i^{n+} \tag{10}$$

the sample $\boldsymbol{x}$ is classified as positive class, so it is not necessary to continue evaluating the remaining $Nsv - n$ support vector, with a consequent saving in the number of kernel evaluations. On the other hand, if these other conditions are given

$$f_n(\boldsymbol{x}) \leqslant 0$$

$$f_n(\boldsymbol{x}) \leqslant -\exp\left(-\frac{d_n^{low^2}}{2\sigma^2}\right) \sum_{i=1}^{Nsv^{n+}} \alpha_i^{n+} + \exp\left(-\frac{d_n^{upp^2}}{2\sigma^2}\right) \sum_{i=1}^{Nsv^{n-}} \alpha_i^{n-} \tag{11}$$

the sample $\boldsymbol{x}$ is classified as negative, and as in the previous case, it is not necessary to calculate the rest of the kernel functions. Note that the summations in the second part of Eqs. (10) and (11) can be precalculated and stored in memory in the training phase. Thus, the operations needed to check the mentioned conditions are an addition to obtain the lower bound and two exponentials. These extra operations represent no cost compared to a kernel evaluation.

In Table 1 it is shown a real example with the real remaining contribution and the maximum contribution calculated from the distance bounds. In this example, once that all the support vectors have been evaluated the decision function has a value $f(\boldsymbol{x}) = 2.27$ and so, it classifies the sample as positive. It can be appreciated how in the bias evaluation and in the three first support

Table 1
Proposed Gaussian method evolution for each support vector ($\sigma^2 = 0.5$ and $f(\boldsymbol{x}) = 2.27$ positive).

Support vector number	$\alpha_i y_i$	$d(\boldsymbol{x}, \boldsymbol{sv}_i)$	$d(\boldsymbol{sv}_1, \boldsymbol{sv}_i)$	$f_n(\boldsymbol{x})$	d_n^{low}	d_n^{upp}	Remain. contrib.	Positive contrib. bound	Negative contrib. bound
0				−1.39 (Bias)			3.54	9.82	
1	5.23	0.171	0.00	3.54	0.00	0.84	−1.26		−16.67
2	−9.91	0.427	0.28	−3.34	0.00	0.84	5.62	11.25	
3	13.41	0.387	0.35	6.59	0.18	0.84	−4.32		−9.21
4	−2.45	0.422	0.37	**4.87**	0.19	0.84	−2.59		**−4.79**
5	−7.93	0.66	0.63	1.55	0.46	0.84	0.72		0.39
6	1.31	0.62	0.67	2.15	0.50	0.84	0.121		0.08
7	0.342	0.72	0.89	**2.27**	0.72	0.84			

Bold values in table show when the proposed and traditional algorithm will stop in the example.

vectors the remaining bounds calculated can change the sign of the evaluation function. However, once that the fourth support vector has been evaluated the bound of the negative contribution is lower than the value of the evaluation function and so, there is no chance to change the sign and the sample is classified as positive. The proposed algorithm will stop at this point avoiding to calculate the rest of support vectors.

The proposed algorithm can be divided into two parts. In the first stage, after training the kernel classifier, the following steps are executed:

(1) Find K centroids over the set of support vectors. These centroids can be calculated using the k-means algorithm.
(2) Select the K support vectors closer to those centroids found in the previous step. For each of the selected support vectors create a list with support vector indexes ranked in ascending order of distance from the selected vector.
(3) Besides having the index of the support vector, each element of the list K_i must store the distance from the first vector of the list, the summation of the remaining positive α_i^{n+} and the summation of the remaining α_i^{n-}.

In the test phase, the following steps are executed:

(1) Select a list L_h in such a way that

$$h = \min_h \quad d(\boldsymbol{x}, \boldsymbol{sv}_1^h) \tag{12}$$

where $d(\boldsymbol{x}, \boldsymbol{sv}_1^h)$ is the distance from the first support vector of the list h.
(2) Evaluate $f_0(x) = b$ and set $n = 1$.
(3) For each support vector in the list, the function $f_n(x)$ is upgraded and the fulfilment of one of the conditions described in Eqs. (10) and (11) is verified. If such conditions are not given the value of n is increased and the step is repeated until the conditions are fulfilled or all the support vectors are evaluated.

The number of centroids K is a free parameter set by default to the number of classes. The optimum number depends on the problem under study, especially in those classes that are distributed in several clusters. Increasing this number ensures that the test sample evaluated is located near a centroid but on the other hand it increases the number of distances to be calculated and the memory requested to storage all the lists. In the extreme case, where the number of centroids is equal to the number of support vectors all the distances must be calculated and so there is no saving in operations.

3. Proposed method with exponential kernel

The bounds found in Eqs. (6) and (7) allow us to calculate the maximum and minimum contribution of the remaining functions $f_n^-(\boldsymbol{x})$ and $f_n^+(\boldsymbol{x})$. However, Eqs. (10) and (11) take into account the same upper and lower distance bounds for all the remaining support vectors. The proposal of this part of the work is to find tighter bounds for the maximum and minimum contribution of the remaining functions.

Instead of the Gaussian kernel described in (2), the exponential kernel is considered

$$K(\boldsymbol{x}, \boldsymbol{sv}_i) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{sv}_i\|}{2\sigma^2}\right) = \exp(-\gamma d(\boldsymbol{x}, \boldsymbol{sv}_i)) \tag{13}$$

On the other hand, each support vector has the following bounds on its distance to the pattern under study $\boldsymbol{x}$:

$$\begin{aligned} d(\boldsymbol{x}, \boldsymbol{sv}_i) &\leqslant d(\boldsymbol{sv}_1, \boldsymbol{sv}_i) + d(\boldsymbol{x}, \boldsymbol{sv}_1) \\ d(\boldsymbol{x}, \boldsymbol{sv}_i) &\geqslant d(\boldsymbol{sv}_1, \boldsymbol{sv}_i) - d(\boldsymbol{x}, \boldsymbol{sv}_1) \end{aligned} \tag{14}$$

And so, the contribution of each support vector is bounded by

$$\begin{aligned} \alpha_i \exp(-\gamma d(\boldsymbol{sv}_i, \boldsymbol{x})) &\geqslant \alpha_i \exp(-\gamma d(\boldsymbol{sv}_1, \boldsymbol{sv}_i)) \exp(-\gamma d(\boldsymbol{sv}_1, \boldsymbol{x})) \\ \alpha_i \exp(-\gamma d(\boldsymbol{sv}_i, \boldsymbol{x})) &\leqslant \alpha_i \exp(-\gamma d(\boldsymbol{sv}_1, \boldsymbol{sv}_i)) \exp(+\gamma d(\boldsymbol{sv}_1, \boldsymbol{x})) \end{aligned} \tag{15}$$

Thus, the stopping conditions with the exponential kernel can be rewritten as

$$f_n(\boldsymbol{x}) \geqslant 0$$

$$\begin{aligned} f_n(\boldsymbol{x}) \geqslant\ & \exp(+\gamma d(\boldsymbol{x}, \boldsymbol{sv}_1)) \sum_{i=1}^{Nsv^{n-}} \alpha_i^{n-} \exp(-\gamma d(\boldsymbol{sv_i}, \boldsymbol{sv}_1)) \\ & - \exp(-\gamma d(\boldsymbol{x}, \boldsymbol{sv}_1)) \sum_{i=1}^{Nsv^{n+}} \alpha_i^{n+} \exp(-\gamma d(\boldsymbol{sv_i}, \boldsymbol{sv}_1)) \end{aligned} \tag{16}$$

and

$$f_n(\boldsymbol{x}) \leqslant 0$$

$$f_n(\boldsymbol{x}) \leqslant -\exp(+\gamma d(\boldsymbol{x},\boldsymbol{sv}_1)) \sum_{i=1}^{Nsv^{n+}} \alpha_i^{n+} \exp(-\gamma d(\boldsymbol{sv}_i,\boldsymbol{sv}_1)) + \exp(-\gamma d(\boldsymbol{x},\boldsymbol{sv}_1)) \sum_{i=1}^{Nsv^{n-}} \alpha_i^{n-} \exp(-\gamma d(\boldsymbol{sv}_i,\boldsymbol{sv}_1)) \quad (17)$$

The steps to execute the algorithm with this second kernel are the same than those described with the Gaussian kernel but the stopping conditions are the one described in Eqs. (17) and (16). Again, the summations included in the stopping conditions can be precalculated during the training phase and the distance to the first support vector is calculated to select the list. Note that, with the common Gaussian kernel, an additional exponential term appears due to the square of the distance and so, it would not be possible to precalculate the aforementioned summations. In Fig. 1 it is shown the comparison between the new bounds and the ones described in the previous section. It can be appreciated the normalized difference between the bounds and the real remaining value for a real case and how these new bounds are quite near to the zero line.

4. Results

The proposed methods were tested in different classification problems with emphasis in those problems with a high number of features. Several problems were selected from the University of California Irvine (UCI) repository [13]. In addition, to test the methods in a RTCS, these were also applied to the traffic sign recognition problem described in [14], using four different datasets associated with different traffic sign groups. Results for the Gaussian kernel case are shown in Table 2 where, besides the description of the features and number of test samples, the number of kernel evaluations, using both the classical method and with the proposed one, is described. Moreover, the percentage of kernel evaluations saved is shown, enabling an appreciation of how the proposed method achieves a high saving in terms of the number of operations, making the system easier to work in real time. Accuracy results were tested to verify that results are the same in all cases.

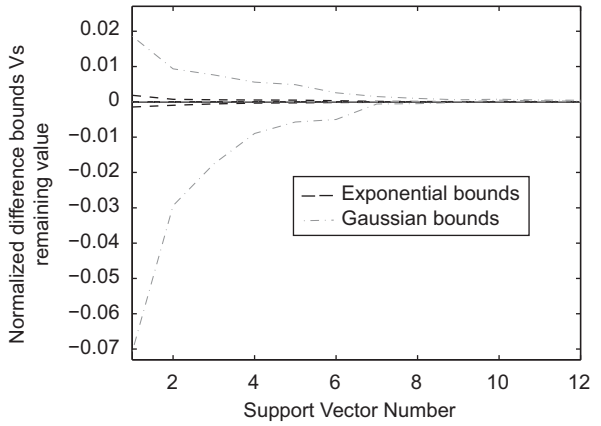

Fig. 1. Comparison between normalized bounds.

Results obtained with the exponential kernel method, in which the stopping conditions are the ones described in Eqs. (16) and (17), are shown in Table 3. As in the previous case, it is also represented the percentage of kernel operations saved, but also compared to the kernel evaluations that were done with the proposed method using the Gaussian kernel. It should be said that, with this second method, savings compared with the classical method are even greater than the ones described in Table 2 but the number of support vectors to achieve similar accuracies is sometimes greater with the exponential kernel. This situation is shown in Figs. 2 and 3 where it can be appreciated the number of kernel evaluations, using the classical method and the proposed algorithm, for a reduced set of test samples when the dataset named as optical letter is considered. Samples were ordered by the number of kernel evaluations needed with the proposed algorithm and it is also shown the mean of the kernel evaluations. The gap between both methods is smaller in the case of the Gaussian kernel but the mean number of kernel evaluations needed, once that the proposed algorithms have been applied is slightly smaller in the case of the exponential kernel.

As it was mentioned, the number of centroids selected in the *k*-means algorithm is a free parameter that has been set to the number of classes in previous examples. In Fig. 4 is represented the percentage of saved kernel evaluations for different number of centroids for the red triangular

Table 2
Results using the proposed method and the Gaussian kernel.

Dataset		Number of features	Test samples	Kernel evaluations		
				Classical method	Proposed method	Saved (%)
UCI	Letter	16	4000	96.1×10^6	41.56×10^6	56.74
	Optical letter	64	1797	1.09×10^6	7.32×10^5	33.27
	Arrythmia	279	200	105×10^3	81.39×10^3	22.48
	Image segmentation	16	2100	3.043×10^6	4.16×10^5	86.32
Traffic sign	Red circular	709	5913	1.28×10^6	9.96×10^5	22.52
	Blue rectangular	961	1091	2.95×10^6	1.21×10^6	58.80
	Blue circular	709	867	1.11×10^6	2.67×10^5	76.06
	Red triangular	511	4067	2.58×10^6	2.014×10^6	22.99

Table 3
Results using the proposed method and the exponential kernel.

Dataset		Kernel evaluations			
		Classical method	Proposed method	Saved (%)	Saved vs. RBF (%)
UCI	Letter	90.65×10^6	22.35×10^6	75.34	46.22
	Optical letter	1.92×10^6	7.12×10^5	62.91	2.73
	Arrythmia	185.2×10^3	21.5×10^3	88.39	73.58
	Image segmentation	2.95×10^6	3.56×10^5	87.93	14.47
Traffic sign	Red circular	2.05×10^6	1.02×10^6	50.24	−2.41
	Blue rectangular	3.52×10^6	1.11×10^6	68.47	8.26
	Blue circular	1.46×10^6	1.89×10^5	87.05	29.21
	Red triangular	2.26×10^6	1.90×10^6	15.92	5.47

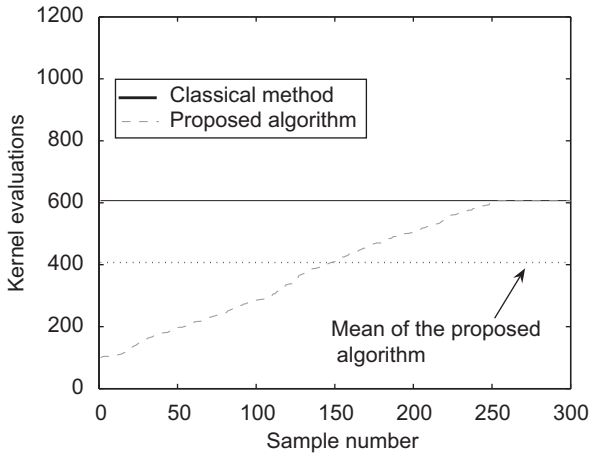

Fig. 2. Kernel evaluations needed for a reduced set of optical letter test samples with Gaussian kernel.

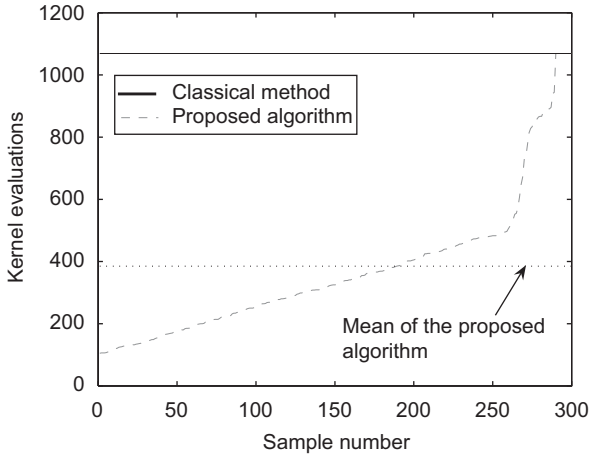

Fig. 3. Kernel evaluations needed for a reduced set of optical letter test samples with exponential kernel.

traffic sign dataset. The conclusion from this figure is also valid for the rest of datasets evaluated and it indicates that increasing the number of lists too much gives worse percentage of saved kernel evaluations. In the example

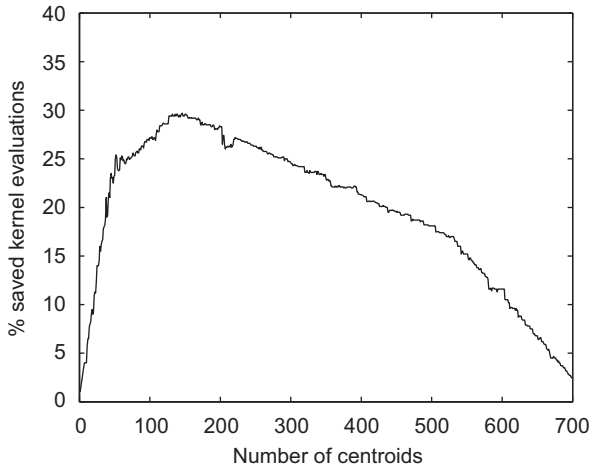

Fig. 4. Percentage of saved kernel evaluations against the number of centroids.

shown in the graphic it can also be seen that the percentage increase fast until the number of classes, 23 in this case, is reached.

5. Conclusion

Two methods to reduce computational load in the test phase have been proposed. The first one can be applied when the kernel used is the popular Gaussian one, with savings in kernel evaluations of 25–85%. The exponential kernel allow to know the remaining values of the positive and negative support vectors with a small error margin and so, savings in operations are greater than in the previous case. Although the use of this second kernel implies to obtain more support vectors, result shows that the total amount of operations saved is usually greater than with the Gaussian kernel.

Acknowledgments

This work was supported by the Comunidad of Madrid under Project CAM-UAH CCG07-UAH/TIC-1740, by the

Ministerio de Educación y Ciencia de España under Project TEC2004/03511/TCM and by the Ministerio de Ciencia e Innovación under Project TEC2008-0277/TEC.

References

[1] B. Scholkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, Cambridge, MA, USA, 2001.
[2] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, USA, 2004.
[3] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[4] J.A. Suykens, T. VanGestel, J. DeBrabanter, B. DeMoor, J. Vandewalle, Least Squares Support Vector Machines: Support Vector Machines, World Scientific, Singapore, 2002.
[5] M. Tipping, Sparse Bayesian learning and the relevance vector machine, Journal of Machine Learning Research 1 (2001) 211–244.
[6] S. Fine, K. Scheinberg, Efficient SVM training using low-rank kernel representations, Journal of Machine Learning Research 2 (2001) 243–264.
[7] C.K. Williams, M. Seeger, Using the Nystrom method to speed up kernel machines, in: Advances in Neural Information Processing Systems, vol. 13, MIT Press, Cambridge, MA, 2001, pp. 682–688.
[8] I.W. Tsang, J.T. Kwok, P.M. Cheung, Core vector machines: fast SVM training on very large data sets, Journal of Machine Learning Research 6 (2005) 363–392.
[9] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, USA, 2006.
[10] J.C. Burges, B. Scholkopf, Improving the accuracy and speed of support vector machines, in: Advances in Neural Information Processing Systems, vol. 9, MIT Press, Singapore, 1997, pp. 375–381.
[11] B. Scholkopf, P. Knirsch, A. Smola, C. Burges, Fast Approximation of Support Vector Kernel Expansions, and an Interpretation of Clustering as Approximation in Feature Spaces, Springer, Berlin, 1998, pp. 124–132.
[12] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.
[13] D.N.A. Asuncion, UCI machine learning repository, 2007 ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.
[14] S. Maldonado, S. Lafuente, P. Gil, H. Gomez, F. Lopez, Road-sign detection and recognition based on support vector machines, IEEE Transactions on Intelligent Transportation Systems 8 (2) (2007) 264–278.