

Fast Stopping in Support Vector Machine Classifiers

Neil Muller

Department of Applied Mathematics
University of Stellenbosch
Private Bag X1, Matieland Stellenbosch 7602 South Africa
neil@dip.sun.ac.za

Abstract

Support Vector Machines have received a lot of attention as a non-linear classifier of late. For realistic datasets, however, the number of support vectors becomes unmanageably large. Most of the proposed approaches to solve this involve approximating the decision boundary. In this article, we propose a simple approach for minimising the number of computations needed to classify a new pattern. For several classes of problems, this can dramatically reduce the time taken to classify elements far from the decision boundary.

1. Introduction

Support Vector Machines (see for example [1, 2]) have received a great deal of attention as a classifier for pattern recognition in the last few years. For instance, in face recognition, support vector machines have been used for pose estimation (see [3]), face detection in complex scenes (see [4]) and feature detection with a face (see [5] or [6]). A recent survey of applications is listed in [7].

One of the major attractions of the support vector machine approach is the easy extension to non-linear problems by means of the so-called “kernel trick” (see for example [8]). This flexibility comes with a price: the non-linear support vector machine is much more expensive to evaluate.

Most of the effort on support vector machines has focused on the computational cost of training the support vector machine (see for example [9]). Comparatively little attention has been focused on the use of the support vector machines, and most of the effort has been spent on approximating the final decision boundary (see [10] or [11]).

In many pattern recognition problems, many of the test cases will be far from the decision boundary. By detecting those cases early, we can stop the evaluation of the support vector machine early and this significantly improve the computational cost of evaluating the support vector machine.

2. Support Vector Machines

2.1. The Linear case

We briefly describe the linear separable case. For a more complete description, see for example [12].

Consider the labelled training set (of size N) (\mathbf{x}_i, y_i) with $y_i = \{-1, +1\} \forall i$. We assume it is linearly separable. Thus there exists some \mathbf{w} and b so that

$$y_i (\mathbf{x}_i^T \mathbf{w} + b) - 1 \geq 0.$$

We look for the separating hyper-plane which maximises the margin. It can be shown that this reduces to maximising

$$\frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \alpha_i y_i (\mathbf{x}_i^T \mathbf{w} + b) + \sum_{i=1}^N \alpha_i$$

subject to

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i \quad (1)$$

and

$$\sum_{i=1}^N \alpha_i y_i = 0.$$

This is a constrained quadratic programming problem and can be solved using standard optimisation techniques.

Once the solution has been obtained, \mathbf{w} is easily calculated from 1 and then a new sample \mathbf{z} can be classified by evaluating

$$\text{sign}(\mathbf{w}^T \mathbf{z} + b).$$

2.2. The Kernel trick

Support Vector Machines can easily be extended to non-linear problems by means of the so-called kernel trick. We note that the vectors in the linear case occur only in dot-products. Thus, if we have some non-linear mapping $\phi(\mathbf{x})$, then we only need to be able to calculate $\phi(\mathbf{x})^T \phi(\mathbf{y})$.

Fortunately, it can be shown that there exists a large class of functions $K(\mathbf{x}, \mathbf{y})$ so that $K(\mathbf{x}, \mathbf{y}) =$

$\phi(\mathbf{x})^T \phi(\mathbf{y})$ for some $\phi(\mathbf{x})$. Thus we can replace all occurrences of $\mathbf{x}^T \mathbf{y}$ with $K(\mathbf{x}, \mathbf{y})$ and calculate the SVM in the new vector space $\phi(\mathbf{x})$. In this case, evaluating the SVM for a new sample \mathbf{z} reduces to

$$\text{sign} \left(\left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}) \right) + b \right) \quad (2)$$

The function $K(\mathbf{x}, \mathbf{y})$ is called the kernel.

It is well known that any $K(\mathbf{x}, \mathbf{y})$ satisfying Mercer's condition can be used as a kernel for SVM's (see for example [8] or [12]).

3. Fast stopping in SVM's

For non-linear support vector machines, we need to express the decision boundary as a combination of the training set elements. For large training sets, the number of evaluations needed to classify a new sample then becomes intractable.

Several approaches have been proposed to reduce the computational cost of classifying new samples, such as that in [11]. These approaches involve approximating the support vector decision boundary, however.

We note, however, that in a large class of problems (face detection, for instance), most of the new samples will be far from the decision boundary. Thus, if we stop evaluation as soon as it is clear that further support vectors will not change the classification, we should significantly improve the classification performance.

Noting that the decision boundary is expressed as 2 and noting that, for the kernel trick to work, we implicitly assume that there exists a $\phi(\mathbf{x})$ so that

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}).$$

We use the standard linear algebra result that

$$K(\mathbf{x}, \mathbf{y}) \leq \sqrt{K(\mathbf{x}, \mathbf{x}) K(\mathbf{y}, \mathbf{y})}. \quad (3)$$

Using 3 and 2, we note that, given

$$P = \sum_{i=1}^Q \alpha_i y_i K(\mathbf{x}_i, \mathbf{z}), \quad (4)$$

the decision will be unchanged if

$$\sum_{i=Q+1}^N |\alpha_i y_i| \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)} \sqrt{K(\mathbf{z}, \mathbf{z})} \leq |P + b|. \quad (5)$$

Since $|\alpha_i y_i| \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}$ can be precalculated for all the support vectors, evaluating this remainder term involves N multiplications and additions only. Furthermore, since this is a cumulative sum, we can easily evaluate the remainder starting from any Q .

We sort the support vectors by $\|\alpha_i y_i\| \sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}$, so that the LHS of 5 decreases as rapidly as possible.

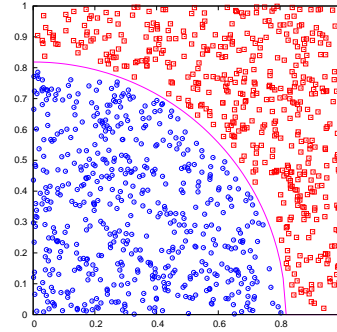


Figure 1: Synthetic Training Data.

Since the $\sqrt{K(\mathbf{x}_i, \mathbf{x}_i)}$ terms can be calculated as soon as the support vector as been trained, the cost of their calculation is not part of the evaluation cost. Thus, to construct the cumulative sum, we need to evaluate $\sqrt{K(\mathbf{z}, \mathbf{z})}$ and then use N multiplications and additions. Therefore, the additional computational cost when we don't stop the support vector machine is $O(N)$. The cost of evaluating the kernel is some function of the dimensionality D of the data, so without early stopping the cost of testing an item is $Nf(D)$, whereas in our case its $cf(D) + O(N)$ where, in most cases, $c \ll N$.

4. Examples

Test runs were done in octave (see [13]), using a support vector machine toolkit available from [14].

4.1. Synthetic Data

We generate a synthetic example in 2D of 1000 training points and 100 test points that is separable by a quadratic boundary. We train a simple quadratic SVM to separate the classes. The training data is shown in figure 1.

Training the support vector machines results in a system with 145 support vectors.

We evaluate the classification performance using both the full SVM approach and our modified example. Unsurprisingly, both methods return a 100% classification success rate.

The full classifier takes 86 seconds to classify the data on a 800MHz Pentium III, while the modified algorithm takes 80 seconds. The small gain in performance is due to the low dimensionality of the data, so the kernel evaluation cost is quite low.

4.2. MNist database

To demonstrate the technique on real world data, we use the MNIST database provided by [15]. This database consists of a normalised and labelled subset of the NIST Special Databases 1 & 3 of handwritten digits. Each element of the database is a 28x28 pixel images, giving

features vectors with 784 elements.

Since we are not interested in accuracy, we have not tried to optimise the support vector parameters to maximise accuracy. We used a simple polynomial kernel of degree 5. We trained two support vector machines, one for class one against all other classes, and another for testing class two against all other classes. In each case, we used a subset of 2000 of the labelled training data, giving support vector machines with 268 and 422 support vectors respectively. We tested with 1000 elements from the test set.

The support vector machines achieve accuracy of 60 % for the 1 versus the rest case and 65 % for the 2 versus the rest case. The running time was 1765 seconds and 2806 seconds respectively for the full case, and 982 seconds and 1223 seconds respectively for the modified case.

5. Conclusions

We demonstrate how, for problems where many of the tested samples will be far from the decision boundary, evaluating an approximation of whether the remaining support vectors can change the decision allows us to abort computation early.

As this evaluation can be done cheaply, with many elements being recalculated, for a large class of real-world problems, this significantly improves the performance of the classifier. As we evaluate the full SVM on decisions close to the boundary, there is no loss of accuracy as well. The cost of checking the approximation is small in comparison to evaluating the full SVM, so the penalty paid is not a significant factor in this case.

This approach can also be combined with many of the existing approximation approaches, with the corresponding loss of accuracy.

6. References

- [1] C. Cortes and V. N. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 1–25, 1995.
- [2] Edgar E. Osuna, Robert Freund and Federico Girosi, "Support vector machines: Training and applications," Tech. Rep. A. I. Memo 1602, MIT Artificial Intelligence Laboratory, Mar. 1997.
- [3] Jeffery Ng and Shaogang Gong, "Multi-view face detection and pose estimation using a composite support vector machine across the view sphere," in *IEEE International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, 1999.
- [4] Edgar Osuna, Robert Freund and Federico Girosi, "Training support vector machines: an application to face detection," in *CVPR' 97*, 1997.
- [5] Jeffrey Huang, David Li, Xuhui Shao and Harry Wechsler, "Pose discrimination and eye detection using support vector machines (SVM)," pp. 528–536. Springer-Verlag, New York, 1998.
- [6] F. Smeraldi, N. Capdeviele and Bigün, "Facial features detection by saccadic exploration of the Gabor decomposition and support vector machines," in *Proceedings of the 11th Scandinavian Conference on Image Analysis*, 1999.
- [7] Hyeran Byun and Seong-Whan Lee, "Applications of support vector machines for pattern recognition: A survey," in *Pattern Recognition with Support Vector machines*, Seong-Whan Lee and Alessandro Verri, Eds. IEEE, Aug. 2002, IEEE Computer Society.
- [8] Bernhard Schölkopf, "Statistical learning and kernel methods," Tech. Rep. MSR-TR-2000-23, Microsoft Research, Microsoft Corporation, Feb. 2000.
- [9] Edgar Osuna, Robert Freund and Federico Girosi, "An improved training algorithm for support vector machines," in *Proceedings of the IEEE International Workshop on Neural Networks for Signal Processing (NNSP97)*, Sept. 1997.
- [10] Christopher J. C. Burges, "Simplified support vector decision rules," in *International Conference on Machine Learning*, 1996, pp. 71–77.
- [11] B. Schölkopf, P. Knirsch, A. Smola and C.J.C. Burges, "Fast approximation of support vector kernel expansions and an interpretation of clustering as approximation in feature spaces," in *DAGM Symposium Mustererkennung*, R. J. Ahler, P. Levi, M. Schanz and F. May, Eds., Berlin, Germany, 1998, pp. 124–132, Springer-Verlag.
- [12] Alex J. Smola and Bernhard Schölkopf, "A tutorial on support vector regression," Tech. Rep. NC2-TR-1998-030, ESPRIT Working Group in Neural and Computational Learning II, Oct. 1998.
- [13] , GNU Octave Homepage, <http://www.octave.org/>.
- [14] Anton Schwaighofer, , Matlab SVM toolkit, www.cis.tugraz.at/igi/aschwaig/software.html.
- [15] , The MNIST database of handwritten digits, <http://yann.lecun.com/exdb/mnist/>.