ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

A Multiplicative Updating Algorithm for Training Support Vector Machines

Nello Cristianini, Colin Campbell
Department of Engineering Mathematics,
University of Bristol, UK
{nello.cristianini,c.campbell}@bris.ac.uk
John Shawe-Taylor
Department of Computer Science, Royal Holloway College, UK
jst@dcs.rhbnc.ac.uk

Abstract. Support Vector Machines find maximal margin hyperplanes in a high dimensional feature space, represented as a sparse linear combination of training points. Theoretical results exist which guarantee a high generalization performance when the margin is large or when the representation is very sparse. Multiplicative-Updating algorithms are a new tool for perceptron learning which are guaranteed to converge rapidly when the target concept is sparse. In this paper we present a Multiplicative-Updating algorithm for training Support Vector Machines which combines the generalization power provided by VC theory with the convergence properties of multiplicative algorithms.

1. Introduction

Multiplicative-updating algorithms are a relatively new technique for training perceptrons. They were first introduced by Littlestone (with his work on Winnow [6]) and Warmuth, and further studied and developed by others [4]. In this approach, weights are modified by a multiplicative factor rather than an additive correction. This is equivalent to performing standard gradient descent in a different space, obtained by taking the logarithm of the weight vector. This strategy is known to perform particularly well in the case of *sparse* target concepts, i.e. when many of the weights of the target perceptron are set to zero [4, 6].

In this paper we propose a multiplicative updating rule which finds a large margin perceptron in the feature space of Support Vector Machines (SVMs). SVMs can perform complex classification tasks by using a nonlinear function ϕ to map training points, x_i, to a high-dimensional space (called the *Feature Space*) where the dataset is linearly separable. By finding the maximal separating hyperplane in Feature Space good generalisation is ensured [8]. Moreover the representation of the solution is frequently *sparse* in the space where the optimization is performed. We briefly describe Support Vector Machines in the

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

next section and a multiplicative rule for training them in Section 3. Experimental results on real and artificial datasets are presented in section 4.

2. Support Vector Learning Algorithms

Hyperplanes can be represented in Feature Space by means of kernel functions which represent the dot products between mapped pairs of input points:

$$K(x', x) = \sum_i \phi_i(x')\phi_i(x)$$

Examples of particular kernel functions are Gaussians and polynomial kernels:

$$K(x, x') = e^{-||x-x'||^2/2\sigma^2} \qquad K(x, x') = (\langle x, x' \rangle + 1)^d$$

Given p input points x_i and corresponding targets y_i (where $y_i = \pm 1$), the learning task amounts to finding the α_i which maximise the Lagrangian:

$$\mathcal{L}(\alpha) = \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{p} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{1}$$

subject to constraints $\alpha_i \geq 0$ and $\sum_{i=1}^{p} \alpha_i y_i = 0$. Only those points which lie closest to the hyperplane have $\alpha_i > 0$ (the rest have $\alpha_i = 0$) and these are the *support vectors* which are therefore the most informative patterns in the data. The resulting decision function can then be written:

$$f(x) = \text{sign}\left(\sum_{i \in \mathtt{SV}} \alpha_i^o y_i K(x, x_i) + \theta \right) \tag{2}$$

where α_i^o are the α_i found after maximising $\mathcal{L}(\alpha)$ and $\mathtt{SV}$ represent the indexes of the support vectors only (θ is the bias).

Suppose inputs are drawn independently according to a distribution on a domain X, and let c be any concept in X. Then, the probability that a consistent hypothesis generated by d support vectors has error larger than ϵ is upper bounded by**:**

$$\sum_{i=1}^{d} \binom{p}{i} (1-\epsilon)^{p-i}$$

which implies the following upper bound (with confidence $1 - \delta$) for the test error of a consistent hypothesis consisting of d support vectors [5]:

$$\epsilon \leq \frac{1}{p-d} \left(d \log_2 \left(\frac{ep}{d} \right) + \log_2 \left(\frac{p}{\delta} \right) \right)$$

Thus concepts which are sparse with few support vectors in the feature space are associated with good generalisation.

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

3. A Multiplicative Learning Algorithm for Training Support Vector Machines.

Experimental and theoretical results indicate that multiplicative updating rules converge faster than the traditional additive corrections of gradient descent techniques when the perceptron to be learned has many weights set to zero and is therefore sparse [4]. For general target functions, however, additive and multiplicative updatings have been proved to be incomparable: depending on the unknown target function, each can be faster or slower than the other.

In general a multiplicative learning algorithm has a weight vector w_t, and its predictions are $y_t = w_t \cdot x_t$. The update rule is:

$$w_{t+1,i} = w_{t,i} e^{-\eta \frac{\partial L(w_t)}{\partial w_{t,i}}}$$

where $L(w_t)$ is the chosen loss function to be minimized.

For Support Vector Machines we wish to *maximise* the function: $\mathcal{W}(\alpha) = \mathcal{L}(\alpha) - \lambda \sum_i \alpha_i y_i$, where $\mathcal{L}(\alpha)$ is the Lagrangian given in equation (1) and the second term implements the constraint $\sum_{i=1}^{p} \alpha_i y_i = 0$. Furthermore maximisation of $\mathcal{W}(\alpha)$ must observe the constraints: $\alpha_i \geq 0$. For many datasets the vector α will be sparse with only a small fraction of training points positioned close to the separating hyperplane and therefore constituting support vectors. This suggests that a multiplicative algorithm should be efficient for many datasets.

The partial derivative of $\mathcal{W}$ with respect to α is

$$\frac{\partial \mathcal{W}(\alpha)}{\partial \alpha_k} = 1 - \gamma_k - \lambda y_k \qquad \text{with} \qquad \gamma_k = y_k \sum_{i=1}^{p} \alpha_i y_i K(x_i, x_k).$$

The updating rule for the α_k is hence:

$$\alpha_k \leftarrow \alpha_k e^{\eta(1-\gamma_k - \lambda y_k)}$$

The value of λ can be obtained by imposing satisfaction of the constraint $\sum_i \alpha_i y_i = 0$ throughout. The final value of λ can be identified with the bias, θ, in (2). This condition becomes $\sum_k (\alpha_k e^{\eta(1-\gamma_k - \lambda y_k)}) y_k = 0$ and yields:

$$e^{\eta\lambda} = \sqrt{\frac{\sum_{k \in \{\texttt{positive}\}} \alpha_k e^{\eta(1-\gamma_k)}}{\sum_{k \in \{\texttt{negative}\}} \alpha_k e^{\eta(1-\gamma_k)}}}$$

A convenient initialization can be obtained by choosing $\alpha_i > 0$ such that:

$$\sum_i \alpha_i y_i = 0$$

is satisfied. For example if $N_p = |\{y_i > 0\}|$ is the number of positively labelled data points and $N_n = |\{y_i < 0\}|$ the number of negatively labelled points then:

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

$$\begin{aligned} \alpha_i &= 1/N_p \quad \text{for} \quad y_i{>}0 \\ \alpha_i &= 1/N_n \quad \text{for} \quad y_i \leq 0 \end{aligned}$$

The multiplicative updating algorithm is therefore:

- Initialize α
- For epoch = 1 to T
  - For i = 1 to p calculate: $\gamma_i = y_i \sum_k \alpha_k y_k K(x_i, x_k)$ endfor
  - Calculate: $\lambda = \frac{1}{2\eta}\left(\log \sum_{k \in \texttt{pos}} \alpha_k e^{\eta(1-\gamma_k)} - \log \sum_{k \in \texttt{neg}} \alpha_k e^{\eta(1-\gamma_k)}\right)$
  - For i = 1 to p: $\alpha_i \leftarrow \alpha_i \exp(\eta(1-\gamma_i - \lambda y_i))$ endfor
- endfor

3.1. Analysis of Convergence

A theoretical analysis shows that the same convergence properties of Multiplicative-Updating algorithms also apply to multiplicative SV machines (see [1] for the proof): when there are few support vectors they outperform classical gradient descent. In the bounds given in Corollaries 3.3 and 3.4 of [1], after a certain sequence S of updates the value of the objective function falls within a difference ε from the optimal. Both the bound for gradient descent and the one for multiplicative updatings involve a $\sqrt{|S|}$ in the denominator - hence we get the same rate of convergence in both cases - the differences lying in the constants involved. The comparison of the two algorithms then involves comparing the form of the two ε:

$$\varepsilon_{GD} = \frac{p}{\sqrt{|S|}}|\alpha^* - \alpha^0|\left(1 + \frac{R^2}{\gamma^2}\right) \qquad \varepsilon_{MU} = \frac{2}{\gamma\sqrt{|S|}}\sqrt{\left(1 + \frac{R^2}{\gamma^2}\right)\sum_{i=1}^{p}\alpha_i \ln\left(\frac{\alpha_i^*}{\alpha_i^0}\right)}$$

where α^* is the optimal solution and α^0 is the initial point. The MU factor is significantly smaller than the GD when we have a sparse α^* (i.e. few support vectors). Thus for an equivalent number of epochs, the multiplicative algorithm goes much closer to the optimal point than an additive algorithm if the number of support vectors is small. See [1] for futher details.

4. Experimental Results

We have evaluated the performance of the algorithm on four classification datasets. For classification with binary-valued inputs we have used the majority rule and the mirror symmetry problem as examples. For classification

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

datasets with analogue inputs we have used real-life datasets, namely, the sonar classification task of Gorman and Sejnowski [3] and the Wisconsin breast cancer dataset [7].

4.1. Majority Rule. The majority rule is a straightforward binary classification task in which the inputs have binary-valued components ± 1 and the target is $+1$ if the majority of bits in the input string are $+1$ with the output is a -1 otherwise. For bit strings of length 40 and 200 training instances the figures below display the margin (Fig. 1(left)) and generalisation error (Fig. 1(right): 500 test examples were used) versus the number of epochs (for $\sigma = 5.0$). The training error reached 0 in the second epoch.

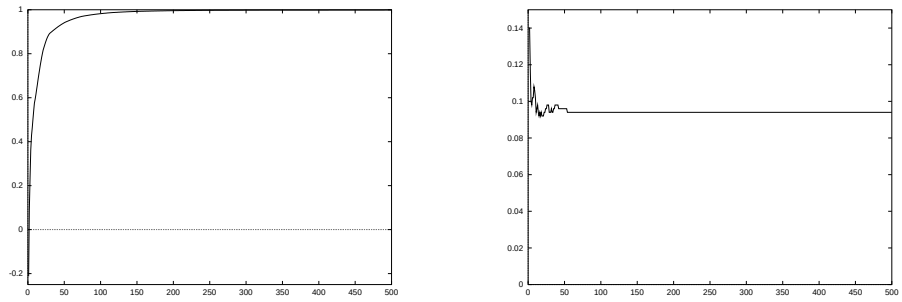

Figure 1: Margin evolution (left) and generalisation error vs number of epochs (right) for the Majority Rule

4.2. Mirror Symmetry Problem. For the mirror symmetry problem the output is a 1 if the input pattern is exactly symmetrical about its centre, otherwise the output is a -1 (the input patterns have components ± 1). For $\sigma = 5.0$, 200 training instances and inputs with 30 component values, the margin evolution (against number of epochs) is illustrated in Fig. 2(left). The final generalisation error was 0.0455 (or 95.5% generalisation) on a test set of 10,000 (allowing repetitions).

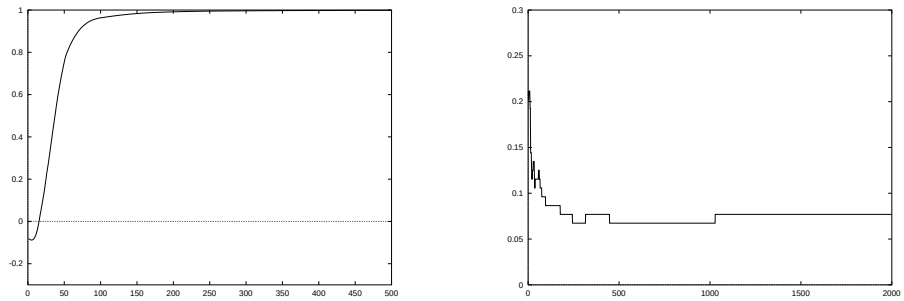

Figure 2: Margin evolution: mirror symmetry problem (left). Generalisation error vs number of epochs: sonar classification experiment(right)

4.3. Sonar classification experiment. For the aspect-angle dependent

ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 189-194

sonar classification problem of Gorman and Sejnowski [3] the 208 instances are equally divided into a training and test set. For $\sigma = 1.0$ the final generalisation performance was 92.31% (Fig. 2(right)) which exceeded the maximum performance of 90.4% reported by Gorman and Sejnowski for a multi-layered neural network trained with the Back-Propagation algorithm and using a variable number of hidden nodes.

4.4. Wisconsin Breast Cancer Dataset. The Wisconsin breast cancer dataset[7] contains 699 patterns with 9 attributes. There are 16 instances with missing values which we discarded from the training and test sets. With $\sigma = 1.0$, a training set of 550 and test set of 133 instances we obtained a generalisation of 95.56% which compares favourably with results presented elsewhere [7]. The training error fell to 0 at the end of the first epoch.

We have verified that the solution obtained is identical to that for a Support Vector Machine trained using a Quadratic Programming (QP) algorithm. Furthermore, the algorithm presented here can be readily extended to handle regression. However, the choice of the learning rate η remains unclear: if η is too large, the algorithm does not converge, too small and it is slow to converge. Currently the algorithm is slower than conventional QP algorithms for this reason but this issue may be resolved with further research.

References

[1] Cristianini N., Campbell C,. Shawe-Taylor J., Multiplicative Updatings for Support Vector Machines; *Neurocolt Technical Report;* `www.neurocolt.com`

[2] Friess T., Cristianini N., Campbell C., The Kernel-Adatron: a Fast and Simple Learning Procedure for Support Vector Machines. In Shavlik, J. (ed), *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, p. 188-196.

[3] Gorman, R.P. and Sejnowski, T.J., *Neural Networks*, 1(1988) p. 75-89.

[4] Kivinen, J. and Warmuth, M., *Journal of Information and Computation*, vol. 132, no. 1, pp. 1-64, 1997

[5] Littlestone, N. and Warmuth, M., Relating Data Compression and Learnability, unpublished manuscript, University of California Santa Cruz, 1986.

[6] Littlestone, N., *Machine learning*, 2:285-318, 1994

[7] Ster, B. and Dobnikar, in A. Bulsari et al. (ed.), *Proceedings of the International Conference EANN'96*, 1996, p. 427-430.

[8] V.Vapnik, T*he Nature of Statistical Learning Theory*, Springer Verlag 1995