Pruning Support Vectors in the SVM Framework and Its Application to Face Detection

Pei-Yi Hao

Department of Information Management, National Kaohsiung University of Applied Science, Kaohsiung, Taiwan, 80778, R.O.C.

Abstract

This paper presents the pruning algorithms to the support vector machine for sample classification and function regression. When constructing support vector machine network we occasionally obtain redundant support vectors which do not significantly affect the final classification and function approximation results. The pruning algorithms primarily based on the sensitivity measure and the penalty term. The kernel function parameters and the position of each support vector are updated in order to have minimal increase in error, and this makes the structure of SVM network more flexible. We illustrate this approach with synthetic data simulation and face detection problem in order to demonstrate the pruning effectiveness.

Keywords: support vector machine, network pruning, model selection, kernel-based learning, face detection.

1. Introduction

In this paper we address the problem of redundant support vectors in implementing support vector machine (SVM), a new pattern classification and function approximation technique recently developed by Vapnik [6; 17]. Traditional techniques for pattern classification are based on the minimization of empirical risk - that is, on the attempt to optimize the performance on the training set. In contrast, SVMs minimize the structural risk - that is, the probability of misclassifying yet-to-be-seen patterns for a fixed but unknown probability distribution of the data. This new induction principle relies on the theory of uniform convergence in probability, and it is equivalent to minimizing an upper bound on the generalization error.

When solving the QP problem in SVM, we occasionally obtain support vectors corresponding to weights close to zero. Those support vectors (SVs) do not significantly affect the final sample classification and function approximation results and thus are redundant in the SVM decision procedure. It is an important issue to reduce the number of support vectors. In this paper we focus on developing an appropriate methodology to eliminate unnecessary support vectors. The related works includes eliminating the redundant SV in a post-processing step of support vector training [4; 5; 12], some other methods use another training algorithm with a modified criterion instead of SVM. The benefits of pruning redundant support vectors include: increasing the speed of classification and regression, reducing hardware or storage requirements, and in some cases enabling meaningful support vector extraction. Furthermore, removing unimportant weights in a trained neural network may improve its generalization ability [14]. This hypothesis could also be supported by the "bound of risk" introduced by [17]. Since the Vapnik-Chervonenkis (VC) dimension of the learning machine is small, the bound of difference between true risk and empirical risk is tight. Intuitively, VC dimension can be adopted as a measure of the capacity of a learning machine, and a learning machine with few weights may have low VC dimension [3]. Therefore, eliminating redundant support vectors reduces the VC dimension and so improves the generalization ability.

The organization of this paper is as follows. We first give an outline of the SVM classification and regression in Section 2, and then describe the redundant support vectors problem. In Section 3, we provide both the penalty term and the sensitivity measure-based pruning algorithms. Experiments are then discussed.

2. Support Vector Machine Frameworks

2.1 SVM Classification:

Let $\{(x_1, y_1),...,(x_l, y_l)\} \subset \aleph \times \{-1, 1\}$ be a given training data set, where $\aleph$ denotes the space of input points. The SVM classification is to map data points x_i into a high dimensional feature space via a nonlinear transform Φ and to classify them by a hyperplane $w \cdot \Phi(x) + b$ in feature space. Its dual quadratic programming classification problem is

$$\begin{aligned} \underset{\lambda_i, \lambda_i^*}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j=1}^{l}\lambda_i\lambda_j\left\langle\Phi(x_i)\cdot\Phi(x_j)\right\rangle + \sum_{i=1}^{l}\lambda_i \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\lambda_i = 0 \\ & \lambda_i \in [0, C] \qquad \forall i \end{aligned} \tag{1}$$

where λ_i are the Lagrange multipliers. The functional form of the mapping $\phi(x_i)$ does not need to be known since it is implicitly defined by the choice of kernel

$$K(x_i, x_j) = \left\langle \phi(x_i) \cdot \phi(x_j) \right\rangle \quad (2)$$

The vector w has the form: $w = \sum_{i=1}^{l} y_i \lambda_i \Phi(x_i)$ and therefore

$$f(x) = \sum_{i=1}^{l} y_i \lambda_i K(x_i, x) + b \quad (3)$$

The points with $\lambda_i \neq 0$ were identified as support vectors since only those points determine the final decision result among all training points.

2.2 SVM Regression

Let $\{(x_1, y_1), ..., (x_l, y_l)\} \subset \aleph \times R$ be a given training data, where $\aleph$ denotes the space of input points. The SVM regression is to map data points x_i into a high dimensional feature space via a nonlinear transform Φ and to regress them by a linear function $f(x) = w \cdot \Phi(x) + b$ in feature space. By introducing an ε -insensitive loss function $|\xi|_\varepsilon$

$$|\xi|_\varepsilon := \begin{cases} 0 & if\ |\xi| \le \varepsilon \\ |\xi| - \varepsilon & otherwise \end{cases}$$

Its dual quadratic programming-regression problem is

$$\underset{\lambda_i, \lambda_i^*}{\text{maximize}} \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{l} (\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*) \left\langle \Phi(x_i) \cdot \Phi(x_j) \right\rangle \\ -\varepsilon \sum_{i=1}^{l} (\lambda_i + \lambda_i^*) + \sum_{i=1}^{l} y_i (\lambda_i - \lambda_i^*) \end{cases} \quad (4)$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{l} (\lambda_i - \lambda_i^*) = 0 \\ \lambda_i, \lambda_i^* \quad \in [0, C] \end{cases}$$

where λ_i and λ_i^* are the Lagrange multipliers. The vector w has the form: $w = \sum_{i=1}^{l} (\lambda_i - \lambda_i^*) \Phi(x_i)$ and therefore

$$f(x) = \sum_{i=1}^{l} (\lambda_i - \lambda_i^*) K(x_i, x) + b \quad (5)$$

The points with $(\lambda_i - \lambda_i^*) \neq 0$ were identified as support vectors since only those points determine the final decision result among all training points.

2.3 Some notes on the SVM framework:

According to Eqs. (3) and (5) the final decision function of SVM has the following formulation

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, \bar{x}_i) + b \quad (6)$$

where $\bar{x}_i$ are support vectors chosen from the training points and $\alpha_i \in [-C, C]$ for i=1,…,n. The support vector machine can be represented as a network architecture, as shown in Fig. 1, where each hidden unit in the network architecture corresponds to one support vector in SVM.

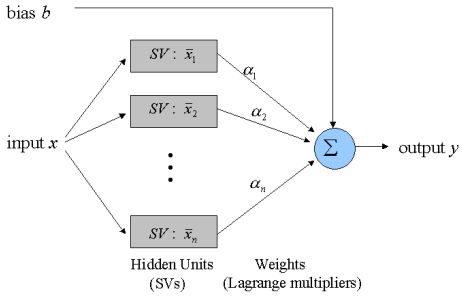


Fig 1: The SVM network architecture representation

The motivating idea behind the SVM network architecture is that decision function has the following two properties:

(1) According to Eq. (6) we observe that only those points corresponding to α_i that differ from zero will affect the final decision result. The value of α_i is obtained by solving the quadratic programming problem in Eqs. (1) and (4). Occasionally, we will obtain some points where α_i is close to zero. Clearly, those points will not significantly affect the final decision result, and we consider them as the redundant support vectors. However, we can not eliminate those points arbitrarily because this could lead to bad results. We propose a suitable mechanism to prune those redundant support vectors without significantly increasing the errors in Section 3.

(2) According to Eqs. (1) and (4) we observe that the learnable parameters during SVM learning procedure are α_i and b. Parameter $\bar{x}_i$ is the support vector coordinate, and it has to choose from training points. Parameter q, the Gaussian kernel width, is a predefined constant. If parameters $\bar{x}_i$ and q are learnable, i.e., the positions of support vectors can be located arbitrarily and the width of Gaussian kernel corresponding to each support vector can be different, then this leads to a flexible formulation in the SVM approximation function. In other words, we can represent the final decision function with fewer support vectors.

3. Support Vector Pruning Algorithm

In this work, we propose a two-stage learning algorithm to prune unimportant support vectors, as shown in Fig 2. The first stage is the original SVM learning, and in the second stage we attempt to prune redundant support vectors. At the same time, we will also update parameters in order to have minimal increase in error.

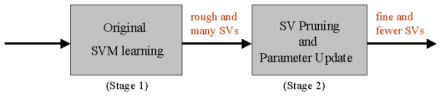


Fig 2: Two-stage procedure for redundant SV pruning

Here we propose a methodology to prune redundant support vectors in a well-trained SVM network obtained after stage 1. We remove redundant support vectors by setting their parameter α_i equal to zero. Meanwhile, we adjust other parameters to minimize the increase in error after pruning them. At this stage, the learnable parameters are α_i, b, $\bar{x}_i$, and q_i,

where q_i is the width of Gaussian kernel corresponding to ith support vector. Furthermore, we use the actual output obtained from the well trained SVM network in stage 1 as the desired training target in stage 2 in order to preserve the generalization ability of the original SVM learning.

3.1 Penalty Term Method

The first method modifies the error function so that the normal back-propagation algorithm effectively prunes the network by driving weights to zero during training. The weights may be removed when they decrease below a certain threshold. The modified error function provides an appropriate tradeoff between reliability of the training data and complexity of the model. This tradeoff can be realized by minimizing the total risk expressed as:

$$E = E_p + \lambda E_c \tag{7}$$

The first term E_p is the standard performance measure, which is typically defined as a mean-square error

$$E_p = \frac{1}{2}\left(f^{[k]} - d^{[k]}\right)^2$$

where $$f^{[k]} = \sum_{i=1}^{n} \alpha_i \exp(-q_i \left\| x^{[k]} - \bar{x}_i \right\|^2) + b$$

is the actual output, and $d^{[k]}$ is the desired output for current input $x^{[k]}$. E_c is the complexity penalty, which depends on the network (model) alone. Since a hidden unit (SV) with parameter α_i that is almost zero will not affect the final decision result, we can reasonably remove it. Here we adopt the complexity term proposed by [19], and apply it to our SVM pruning algorithm. The complexity penalty term is

$$E_c = \sum_{i=1}^{n} \frac{(\alpha_i/\alpha_0)^2}{1+(\alpha_i/\alpha_0)^2} \tag{8}$$

where α_0 is a predefined constant. For $|\alpha_i| >> \alpha_0$, the cost of a weight approaches λ. For $|\alpha_i| << \alpha_0$, the cost is nearly 0. The flowchart of the penalty term-based redundant support vector pruning method is shown in Fig. 3.

Now we use gradient descent method to update the parameters in the SVM. The basic updating rule for weight is

$$W(t+1) = W(t) + \eta\left(-\frac{\partial E}{\partial W}\right) \tag{9}$$

3.2. Sensitivity Calculation Method

The basic idea of this approach is to use information on second-order derivatives of the error surface in order to make a trade-off between network complexity and training performance [8; 10]. In a well trained SVM network, the functional Taylor series of the error with respect to the weights (or parameters) is

$$\delta E_p = \left(\frac{\partial E_p}{\partial w}\right)^T \cdot \delta w + \frac{1}{2}\delta w^T \cdot H \cdot \delta w + O\left(\|\delta w\|^3\right) \tag{10}$$

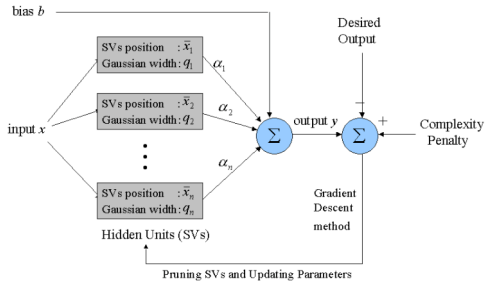


Fig 3: Flowchart of support vectors pruning using penalty term.

where $H \equiv \partial^2 E_p / \partial w^2$ is the Hessian matrix. For a network trained to minimum error, the first (linear) term vanishes.

We ignore the third and all higher order terms. The purpose of the sensitivity calculation method is to set to zero one of the weights that minimizes the increase in error given in Eq (10). Eliminating w_q is expressed as $e_q^T \cdot \delta w + w^T = 0$ where e_q is the unit vector in weight space corresponding to weight w_q. The above task is therefore to

$$\min_q \left\{ \min_{\delta w} \left(\frac{1}{2}\delta w^T \cdot H \cdot \delta w \,\middle|\, s.t.\ e_q^T \cdot \delta w + w_q = 0 \right) \right\} \tag{11}$$

The Lagrangian can be formulated as:

$$L = \frac{1}{2}\delta w^T \cdot H \cdot \delta w + \beta(e_q^T \cdot \delta w + w_q) \tag{12}$$

where β is a Lagrangian multiplier. Taking the derivative of L with respect to δw and setting to zero, the "optimal weight change" and the "resulting change in error" can be obtained as:

$$\delta w = -\frac{w_q}{\left[H^{-1}\right]_{qq}} H^{-1} \cdot e_q \tag{13}$$

$$L_q = \frac{1}{2}\frac{w_q^2}{\left[H^{-1}\right]_{qq}} \tag{14}$$

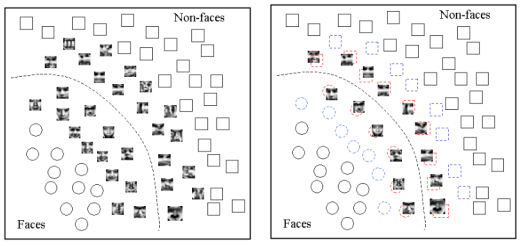
where L_q is the "saliency" of weight q – the increase in error when the weight q was pruned.

4. Experiments and Discussion

In this section, we apply proposed pruning algorithms to a real world application - face image detection. Speed is an important factor in face detection problem, and eliminating redundant SVs will definitely increase the speed of face detection. We utilize the MIT CBCL face database as training and test data (http://www.ai.mit.edu/projects/cbcl/). The training set was generated by [16] and the non-faces training set was generated by [9]. There are 2,429 faces and 4,548 non-faces in the final training set. The test set we used is a subset of the CMU Test Set 1 15], relevant information about how the subset was chosen can be found in [9]. There are 472 faces and 23,573 non-faces in the test set. This is the same data set that was used in [1]. Each image in the database is of size 19 by 19, which either contains a human face or not.

In original support vector learning we extract 1,464 support vectors from the training data where the parameters are set to be C=10 and q=0.02. The percentage ratio of SVs is about

20.9% among training data. The classification accuracy is 100% in training set and 97.812% in test set, respectively in original SVM approach. Using penalty term method we eliminated 709 redundant SVs while the classification accuracy is 100% in training set and 97.812% in test set, respectively. Using sensitivity calculation method we eliminated 676 redundant SVs while the classification accuracy is 100% in training set and 97.775% in test set, respectively. Since about 50% SVs were pruned in both pruning algorithms, the final face detection performs almost two times faster than the original one. Besides, the SVs in face detection have another geometrical interpretation as shown in Fig. 4 (a). Those SVs are images (contained both faces and non-faces) that selected from training set to support the final decision function, which are called the support faces [11]. After pruning redundant SVs, the image pixels in remaining support faces change to new values, and this makes image deviation from original location. We denoted the remaining SVs as extended support faces, and these extended support faces do not belong to the original training images, as shown in Fig. 4 (b). Fig. 5 shows some of face detection results in the CMU test set.



(a) (b)

Fig. 4: Geometrical interpretation of how the SVM model separates the face and non-face classes. (a) The support faces are obtained after SVM training. (b) After pruning, the square and circular symbols drawn with blue dot line denote the redundant support faces that have been eliminated. The remaining support faces deviate from the original support faces while the square and circular symbols drawn with red dot line denote the original support faces.

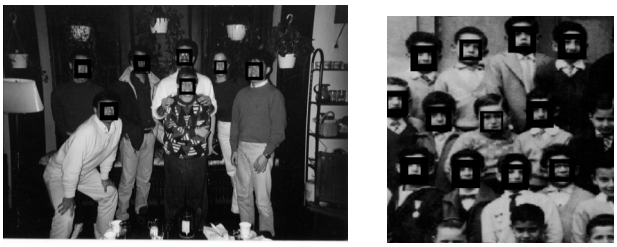


Fig. 5: Examples of face detection results in CMU test images

5. References

[1] Alvira, M. and R. Rifkin, "An Empirical Comparison of SNoW and SVMs for Face Detection," Center for Biological and Computational Learning, MIT, A.I. memo, Cambridge, MA, no. 2001-004, 2001.

[2] Blanz, V., B. Schölkopf, H. Bulthoff, C. J. C. Burges, V. N. Vapnik, and T. Vetter, "Comparison of view-based object recognition algorithms using realistic 3D models". In Proc of ICCANN'96. LNCS Vol. 1112, 251-256, 1996.

[3] Burges, C. J. C., "A tutorial on support vector machines for pattern recognition." Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 955-974, 1998

[4] Burges, C. J. C. and B. Scholkopf, "Improving the accuracy and speed of support vector learning machines," In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems* 9, pages 375-381, Cambridge, MA, 1997. MIT Press.

[5] Burges, C. J. C., "Simplified support vector decision rules," In *Proc. 13th Inter. Conf. on Machine Learning*, pp. 71-77, San Mateo, CA, 1996, Morgan Kaufmann.

[6] Cortes, C. and V. N. Vapnik, "Support Vector Network.", *Machine Learning*, vol. 20, pp. 1-25, 1995.

[7] Haykin, S., Neural Networks – a Comprehensive Foundation. Prentice Hall, New Jersey, 2nd edition, 1999.

[8] Hassibi, B., D. G. Stork, and G.. J. Wolff, "Optimal Brain surgeon and general network pruning." IEEE International Conference on Neural Networks, vol. 1, pp. 293-299, San Francisco, 1992.

[9] Heisele, B., T. Poggio, and M. Pontil. "Face detection in still gray images," AI Memo 1687, Massachusetts Institute of Technology, 2000.

[10] LeCun, Y., J. S. Denker, and S. A. Solla, "Optimal brain damage." Advances in Neural Information Processing Systems, vol. 2, pp. 598-605, San Mateo, CA: Morgan Kaufmann, 1990.

[11] Moghaddam, B., and Ming-Hsuan Yang, "Learning gender with support faces," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 707-711, May 2002.

[12] Osuna, E., and F. Girosi. "Reducing run-time complexity in SVMs." In *Proceedings of the 14th International Conf. on Pattern Recognition, Brisbane, Australia*, 1998.

[13] Weigend, A. S., D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with approach to forecasting." Advances in Neural Information Processing Systems, vol. 3, pp. 875-882, San Mateo, CA: Morgan Kaufmann, 1991.

[14] Reed, R., "Pruning algorithms—A survey," IEEE Trans. Neural Networks, vol. 4, pp. 740–747, Sept. 1993.

[15] Rowley, H. A., S. Baluja and T. Kanade, "Neural Network-Based Face Detection," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, 1998.

[16] Sung, K.-K., "Learning and Example Selection for Object and Pattern Recognition," MIT, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Cambridge, MA, 1996.

[17] Vapnik, V. N., The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.