Feature Selection for Support Vector Machines

Lothar Hermes and Joachim M. Buhmann
Dept. of Computer Science III, University of Bonn
Römerstr. 164, 53117 Bonn, Germany
{hermes, jb}@cs.uni-bonn.de

Abstract

In the context of support vector machines (SVM), high dimensional input vectors often reduce the computational efficiency and significantly slow down the classification process. In this paper, we propose a strategy to rank individual components according to their influence on the class assignments. This ranking is used to select an appropriate subset of the features. It replaces the original feature set without significant loss in classification accuracy. Often, the generalization ability of the classifier even increases due to the implicit regularization achieved by feature pruning.

1. Introduction

In a typical classification scenario, each data unit is represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ consisting of n individual components. When classifying customers, for example, typical feature components might code information about the age of the customer, his sex, his income, his marital status etc. In many applications however, it is not known a priori which of these components are relevant for a given classification problem. From an efficiency viewpoint, the classification should be based on as few vector components as possible to avoid any unnecessary computation. Moreover, limiting the number of features can sometimes be helpful because it cuts down the model capacity and thus reduces the risk of over-fitting. On the other hand, one usually wishes to achieve the best classification quality that is still practicable. Reducing the input dimension always bears the danger of reducing the expected classification performance by introducing a bias. As a consequence, the omission of data components is a very critical task and should – if possible – be restricted to those components which do not contain any information about the correct class assignments.

This paper describes a feature selection strategy which defines scores for the available features on the basis of a single training run. Being especially designed for support vector machines, this technique reorders the feature dimensions according to their relative importances for the classification decision and that way assigns priorities for retaining them in the final feature set. The applicability of this strategy is demonstrated for artificial and real-world data. A combination with more general strategies that iteratively add or drop feature subsets [6] is beyond the scope of this paper.

	$K(\mathbf{u}, \mathbf{v})$	$\nabla_{\mathbf{v}} K(\mathbf{u}, \mathbf{v})$
lin.	$\mathbf{u}^T\mathbf{v}$	$\mathbf{u}$
pol.	$\left(1 + \mathbf{u}^T\mathbf{v}\right)^\theta$	$\theta\left(1 + \mathbf{u}^T\mathbf{v}\right)^{\theta-1}$
RBF	$\exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{\theta}\right)$	$\frac{2(\mathbf{u}-\mathbf{v})}{\theta} \cdot \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{\theta}\right)$
NN	$\tanh\left(\theta_1 \mathbf{u}^T\mathbf{v} - \theta_2\right)$	$\frac{\theta_1}{\cosh^2(\mathbf{u}^T\mathbf{v}-\theta_2)} \cdot \mathbf{u}$

Table 1. Some common kernel functions.

2. Support Vector Machines

Consider a classification problem with two classes ω_+ and ω_- first. Given ℓ training examples $\mathbf{x}_i \in \mathbb{R}^d$ labeled by $y_i \in \{+1, -1\}$, support vector machines [5] separate ω_+ and ω_- by an *optimal hyper-plane* $\mathbf{w}^T\mathbf{x} + b = 0$, where the optimality is defined by maximizing its distance from the nearest training example. To enforce a unique parameterization of the hyper-plane, one demands

$$y_i \cdot \left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 \quad \forall i = 1, \ldots, \ell \ , \tag{1}$$

and minimizes $\frac{1}{2}\|\mathbf{w}\|^2$ under these constraints. This concept can also be extended to the non-separable case, i.e. when (1) has no solution. Introducing *slack variables* $\xi_i \geq 0, i = 1, \ldots, \ell$, (1) is then weakened to

$$y_i \cdot \left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i \quad \forall i = 1, \ldots, \ell \ , \tag{2}$$

while the objective function is supplemented to keep the constraint violation as small as possible:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{\ell} \xi_i \stackrel{!}{=} \min \ . \tag{3}$$

This problem is usually posed in its *dual form* in terms of Lagrange multipliers $\lambda_i \in [0, C]$, $i = 1, \ldots, \ell$. It can be

solved by standard numerical optimization packages or by specifically adapted quadratic optimization techniques. After optimization, the distance of a vector $\mathbf{x}$ from the optimal separating hyper-plane is proportional to

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + b \ , \tag{4}$$

where the bias b can easily be calculated from any *margin vector* $\mathbf{x}_i$ satisfying $0 < \lambda_i < C$. The decision function $f(\mathbf{x})$ that assigns a vector $\mathbf{x}$ to a class y is therefore given by $f(\mathbf{x}) = \operatorname{sgn}(g(\mathbf{x}))$. In a typical classification task, only a small subset of the Lagrange multipliers λ_i usually tend to be greater than zero. The respective training vectors are called *support vectors*, as $f(\mathbf{x})$ depends on them exclusively.

Noticing that the training vectors $\mathbf{x}_i$ are solely used in inner products, the SVM concept can also be extended to a non-linear classification strategy. The inner products are then replaced by a kernel function $K(\mathbf{u}, \mathbf{v})$ that obeys Mercer's condition [5]. This replacement can be interpreted as mapping the data vectors into a high-dimensional feature space preceding the hyper-plane classification. Depending on the selected kernel function, several well-known classification schemes like polynomial classifiers, RBF functions or two-layer neural nets are obtained (Tab. 1).

For a K-class problem, K individual support vector machines are trained to separate one class ω_k from all other classes. A given vector $\mathbf{x}$ is then assigned to the class with maximal distance $g_k(\mathbf{x})$ as given by (4).

3. Ranking Features

For our purposes, the most elementary situation is given whenever an individual data component, say x_1, does not carry any information on the class label y of data objects. In this case, one aims at identifying a *selection function* $s(\mathbf{x}) = (x_2, \ldots, x_n)$ that is independent of the first data component x_1 and still provides a *sufficient statistic* for y:

$$p(y \mid \mathbf{x}, s(\mathbf{x})) = p(y \mid s(\mathbf{x})) \ . \tag{5}$$

When dealing with practical problems however, this case is very unlikely to occur. Moreover, estimating the probability distributions in (5) is impracticable due to the restricted training data typically available.

Instead, we propose to estimate the importance of individual feature components from the discriminant function $g(\mathbf{x})$ provided by the SVM. The idea is to first train the support vector machine using all available data components. The training provides optimal values for the Lagrange multipliers λ_i which in turn constitute the support vectors and the decision boundary (Fig. 1 (a), circles and solid line, respectively). To rank the components of a given vector $\mathbf{x}$ according to their influence on the classification decision, we then compute the gradient of $g(\mathbf{x})$ at position $\mathbf{x}$. Fig. 1 (b) shows a typical contour plot of $g(\mathbf{x})$ for a RBF kernel, the small arrows indicating the local gradients $\nabla g(\mathbf{x})$. As $g(\mathbf{x})$ is a linear combination of kernel products, $\nabla g(\mathbf{x})$ can be written as the weighted sum of kernel derivatives:

$$\nabla g(\mathbf{x}) = \sum_{i \in SV} \lambda_i y_i \nabla_{\mathbf{x}} K(\mathbf{x}_i, \mathbf{x}) \ . \tag{6}$$

For all main kernels, $\nabla_{\mathbf{x}} K(\mathbf{x}_i, \mathbf{x})$ can easily be evaluated (table 1). After normalization, $\nabla g(\mathbf{x})$ is compared to the unit vectors $\mathbf{e}_j$, $j = 1, \ldots, n$, representing the indices of the individual features. If a feature x_j is not important at position $\mathbf{x}$, $\nabla g(\mathbf{x})$ should be roughly orthogonal to $\mathbf{e}_j$, i.e. $\mathbf{x}_j$ should not influence the distance to the decision hyper-plane. So we measure the angle α_j between $\nabla g(\mathbf{x})$ and $\mathbf{e}_j$:

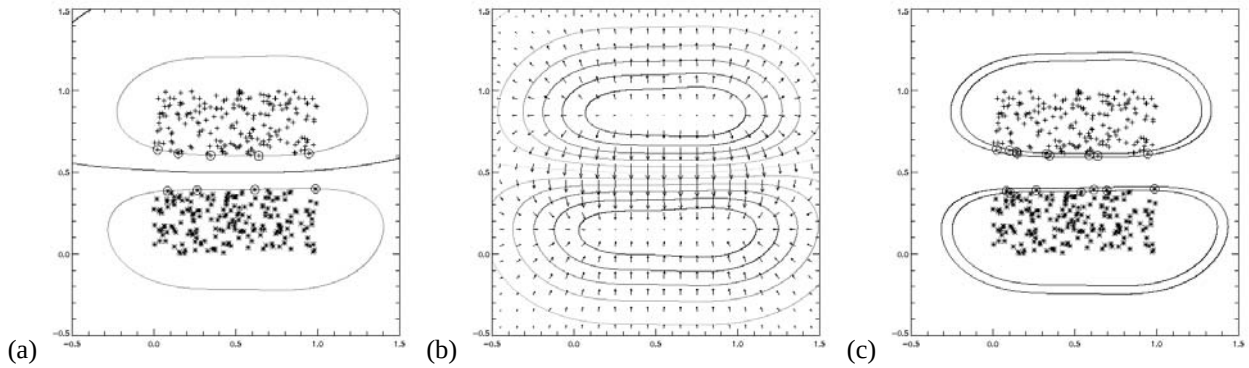
$$\alpha_j = \min_{\beta \in \{0,1\}} \left\{ \beta\pi + (-1)^{\beta} \arccos\left(\frac{(\nabla g(\mathbf{x}))^T \mathbf{e}_j}{\|\nabla g(\mathbf{x})\|} \right) \right\} \ . \tag{7}$$

Values $\alpha_j(\mathbf{x}) \approx \pi$ indicate that the respective vector component has only weak influence on the assignment of the vector $\mathbf{x}$. Small values indicate important features.

Where should this function be evaluated? Whenever a vector $\mathbf{x}$ is located far away from the decision boundary, its class assignment is insensitive to small displacements. Only $\mathbf{x}$ located near the decision hyper-plane could effectively influence its actual position. To get a non-local measure for the importance of individual features, α_j should thus be averaged over several positions within a restricted area of increased uncertainty just around the decision hyper-plane.

It is a key observation [3] that the optimization of the SVM problem always identifies one or more so-called *margin vectors* $\mathbf{x}_i$ for which $0 < \lambda_i < C$. Among the support vectors, these $\mathbf{x}_i$ are of special interest because of their property $y \cdot g(\mathbf{x}_i) = 1$, i.e. they are positioned adjacent to the decision hyper-plane, though are classified correctly. Obviously they are thus located in an area of increased uncertainty like the one we mentioned above. Therefore, it is a natural choice to evaluate the gradient angle at the margin vectors. As the number of margin vectors is typically quite small, especially in cases of good generalization, we also propose to include all vectors within a narrow ϵ-region around them (fig. 1 (c)). To formalize this, we demand that $|g(\mathbf{x}_i) - 1| \leq \epsilon$ must be valid for each training vector $\mathbf{x}_i$ which is considered for the feature evaluation. By I_ϵ summarize the indices of all training vectors which match this condition and abbreviate $\alpha_j(\mathbf{x}_i)$ by α_{ij} for notational convenience. We define $\tilde{\alpha}_j \in [0, 1]$ which averages the angles α_{ij} over $i \in I_\epsilon$:

$$\tilde{\alpha}_j = 1 - \frac{2}{\pi} \cdot \frac{\sum_{i \in I_\epsilon} \alpha_{ij}}{|I_\epsilon|} \ . \tag{8}$$



(a) (b) (c)

Figure 1. Two well separated classes in the plane, RBF-kernel, $\theta = 0.5$.

Given $\tilde{\alpha}_j$, $j = 1, \ldots, n$, we can now establish a ranking among the individual data components: Features with indices j are ranked first if $\tilde{\alpha}_j$ is large. The smaller $\tilde{\alpha}_j$, the smaller the estimated importance of a feature. If it is necessary to dispense with some components, we choose to keep the ones for which $\tilde{\alpha}_j$ is largest.

Although the proposed strategy is not based on explicit density estimations, it can still be related to the sufficient statistics approach mentioned earlier in this section. Assume that the a-posteriori probability $p(y = 1|\mathbf{x})$ can be written in the form

$$p(y = 1|\mathbf{x}) = (h \circ g)(\mathbf{x}) \quad , \tag{9}$$

where $h(t)$ is a continuous and smooth function increasing strictly monotonous with t. In fact, it has been shown in several experiments [4, 2] that estimating h can significantly enhance the classification performance of SVM, what might be interpreted as an empirical justification of (9). Again assume that the first data component x_1 is irrelevant, so that $s(\mathbf{x}) = (x_2, \ldots, x_n)$ is a sufficient statistic for y in the sense of (5). The a-posteriori probability $p(y = 1|\mathbf{x})$ should then be insensitive to any variation of x_1, i.e.

$$\frac{\partial}{\partial x_1} p(y = 1|\mathbf{x}) = h'(g(\mathbf{x})) \cdot \frac{\partial}{\partial x_1} g(\mathbf{x}) = 0 \ . \tag{10}$$

As $h'(g(\mathbf{x})) > 0$ according to our assumption, we have $\frac{\partial}{\partial x_1} g(\mathbf{x}) = 0$ for all $x_1 \in \mathbb{R}$ causing $\tilde{\alpha}_1$ to be zero. The irrelevant feature x_1 would thus be the first candidate to be discarded.

4. Results

In the first experiment, we sampled 500 vectors from two classes in a 20-dimensional data space, the class centers being located at $(1, \ldots, 1)^T$ and $(-1, \ldots, -1)^T$, respectively. Each component of the vectors was independently corrupted by white Gaussian noise with standard deviation dependent on the index i of the respective data component as $0.2 \cdot 1.2^{i-1}$. Fig. 2 (a) shows the feature importances that we estimated via (8) for a SVM with a polynomial kernel of degree 2 and $C = 10$. The decreasing values of $\tilde{\alpha}_j$ reflect the increasing noise in the respective data components x_j. In Fig. 2 (b), the change of the classification performance is plotted against the number m of retained features, the performance H^n of using all available features being taken as a reference value. The solid, dotted, and dashed lines denote the average performances (100 experiments) when using the most important, ($H^m_{\tilde{\alpha}+}$), randomly selected (H^m_r), and the least important features ($H^m_{\tilde{\alpha}-}$), respectively. We also implemented the strategy of ranking the features by their *average separation* [6]

$$\gamma_j \propto \sum_{y_k=+1} \sum_{y_l=-1} \|\mathbf{x}_{k,j} - \mathbf{x}_{l,j}\| \quad , \tag{11}$$

which performs bad in this case because it misinterprets the noise as structure (H^m_{as}). While our feature ranking strategy first discards the most unreliable components causing even an increase in the overall performance, the application of any of the other strategies instantly decreases the classification performance significantly.

Additional experiments were performed on several datasets from [1]. Fig. 3 (a) is based on the *wine recognition database* in which 13 chemical characteristics of Italian wines are used for an assignment to one of three cultivars. To handle this multi-class case, three binary linear SVM were trained. Their individual score values were averaged to obtain a common importance measure $\tilde{\alpha}_j = \sum_k \tilde{\alpha}^k_j / 3$. The graphic compares the performances of our score function $\tilde{\alpha}_j$ and the average separation γ_j as a function of the number m of retained features (the performance of random feature selection forming the baseline, the small graphic inside depicting the estimated feature importances). The results were averaged over 500 experiments, where a random $2/3$ part of the data vectors served as training set, the rest being used for testing. In this experiment, the advantage of our strategy over the average separation criterion is less significant because the class structures are well-

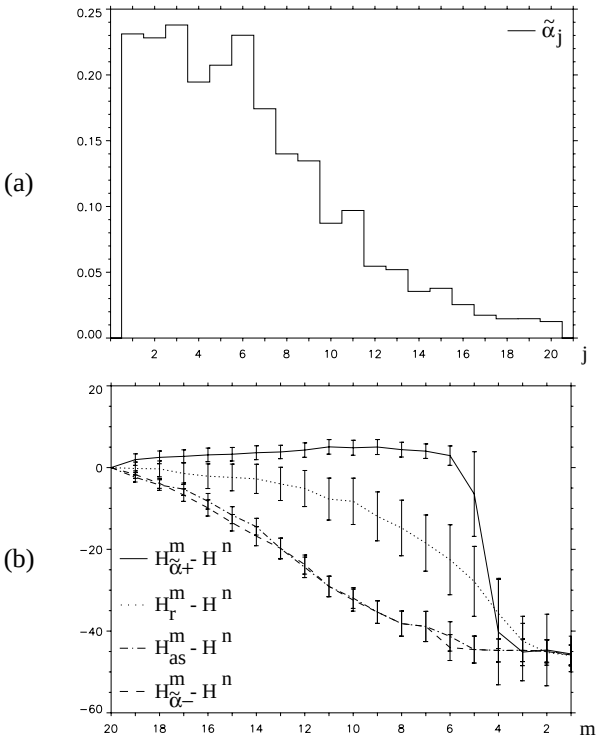


Figure 2. Results for an artificial dataset.

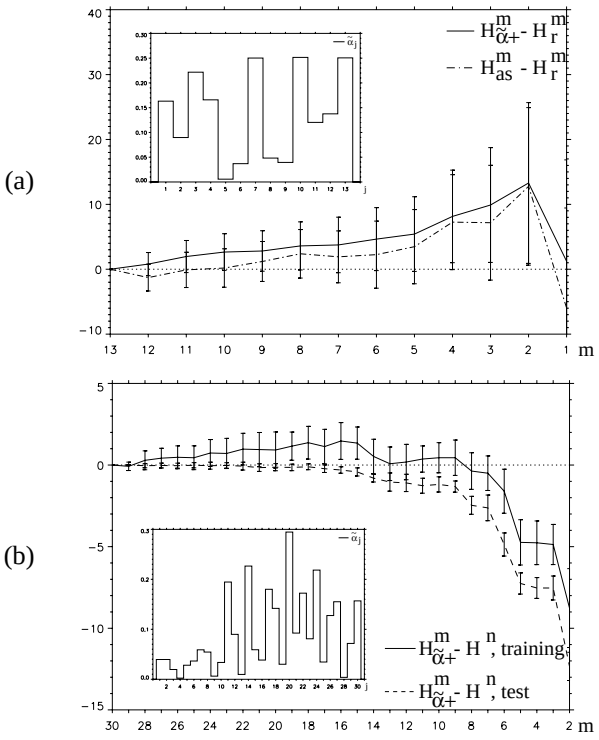


Figure 3. Results for real-world datasets.

behaved, preventing γ_j from suffering from its inherent sensitivity to noise. In Fig. 3 (b), the development of training and classification performances is plotted for the *Wisconsin Diagnostic Breast Cancer* dataset [1] (same parameter settings as before, results averaged over 20 random experiments). When features are discarded, the training performance (dashed line) monotonically decreases, whereas the test performance (solid line) slightly increases first. This can be explained by reduced over-fitting effects due to the smaller number of features. A drastic reduction of features, however, also leads to a decrease in the test performance. This behavior seems to be typical and has been observed in several other experiments as well.

5. Conclusion

In this contribution, we proposed a novel strategy to select individual features for SVM classification. It is based on the evaluation of local kernel gradients in the vicinity of the decision hyper-plane. At the first glance, this *pruning* of features might seem to be contradictory to the general SVM-strategy to operate in high-dimensional feature spaces and automatically select the right features there. However, decreasing the input dimension (as opposed to the dimension of the embedding space of the SVM) not only helps to reduce the influence of noise but yields a substantial acceleration of the SVM classifier. This is crucial in time critical applications or in case of large amounts of data. Our experiments validate that the proposed strategy produces satisfactory results both on artificial and on real-world data. Further improvements can probably be achieved by combining our score function with iterative feature selection schemes [6].

Acknowledgement

We thank Jan Puzicha and John Held for their valuable contributions. This study has been supported by the German Aerospace Agency (DLR) under grant 50EE9926.

References

[1] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, `http://www.ics.uci.edu/~mlearn`.

[2] L. Hermes, D. Frieauff, J. Puzicha, and J. M. Buhmann. Support vector machines for land usage classification in Landsat TM imagery. In *Proc. of the IEEE IGARSS'99*, 1999.

[3] E. E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. AI Memo 1602, AI Lab, Massachusetts Institute of Technology, March 1997.

[4] J. C. Platt. Probabilistic outputs for suppoort vector machines and comparisions to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1998.

[5] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[6] A. Webb. *Statistical Pattern Recognition*. Arnold, 1999.