On Mixtures of Linear SVMs for Nonlinear Classification

Zhouyu Fu[1] and Antonio Robles-Kelly[1,2]

[1] RSISE, Bldg. 115, Australian National University, Canberra ACT 0200, Australia
[2] National ICT Australia (NICTA)*, Locked Bag 8001, Canberra ACT 2601, Australia

Abstract. In this paper, we propose a new method for training mixtures of linear SVM classifiers for purposes of non-linear data classification. We do this by packaging linear SVMs into a probabilistic formulation and embedding them in the mixture of experts model. The weights of the mixture model are generated by the gating network dependent on the input data. The new mixture of linear SVMs can be then trained efficiently using the EM algorithm. Unlike previous SVM-based mixture of expert models, which use a divide-and-conquer strategy to reduce the burden of training for large scale data sets, the main purpose of our approach is to improve the efficiency for testing. Experimental results show that our proposed model can achieve the efficiency of linear classifiers in the prediction phase while still maintaining the classification performance of nonlinear classifiers.

1 Introduction

Support Vector Machines (SVMs) [1] have been widely used in different application areas including pattern recognition, computer vision and machine learning. SVMs seek to recover an optimal separating hyperplane in the feature space such that the margin between two classes, i.e. the minimum distance between data points with different labels, is maximised. By mapping the input data onto a higher-dimensional feature space in a nonlinear fashion and seeking an optimal separating hyperplane in the feature space, SVMs can generalise so as to handle linearly inseparable data sets. Moreover, the feature mapping can be done implicitly through the kernel trick.

Despite all their merits, nonlinear SVMs can carry a heavy computational burden for label prediction at testing time. To classify a single novel testing point, one needs to evaluate the nonlinear kernel function many times depending on the size of the set of support vectors extracted in the training phase. This can be very computationally intensive for large-scale prediction tasks. One such example is image classification, where feature vectors can be extracted at pixel

* NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 489–499, 2008.

level and an SVM applied to each of the pixels in the image. In contrast with their non-linear counterparts, linear SVMs can perform testing in linear time in the number of testing samples, whereas they are limited to those cases where the samples are linearly separable.

To combine the advantages of both, linear and nonlinear SVMs, and overcome their limitations, in this paper, we aim at developing a novel classification method with linear time complexity in the prediction stage and the classification performance comparable to the nonlinear classifiers. This is achieved by combining the mixture of experts model proposed by Jordan and Jacobs [2,3] with the use of linear SVM classifiers. Since the mixture of experts model is inherently a maximum likelihood probabilistic model, the trick here is to view the linear SVM from a probabilistic point so as to cast the problem into a maximum likelihood estimation (MLE) framework. We do this by viewing the classifiers as experts in a mixture model.

The work presented here differs from that elsewhere in the literature in a number of ways. Competing approaches [4,5] employ a divide-and-conquer strategy to partition the input space into disjoint regions. Once the space has been partitioned, a local non-linear SVM classifier is trained on samples falling in each region. Again, this may greatly reduce the training time but does not improve the efficiency for prediction due to the fact that non-linear SVMs are still used for the expert components. A straightforward remedy might be to replace the nonlinear SVMs with linear counterparts, but it is impossible, in general, to guarantee the training data in each region are linearly separable. This limits the use of linear classifiers in these settings. With our probabilistic formulation, we can sidestep this problem and formulate the problem in a more principled way. Moreover, the mixture of linear SVMs model can be trained using an EM algorithm by casting the problem into a MLE setting.

The remainder of the paper is organised as follows. Section 2 provides the background on the SVM classifier model. Section 3 details the proposed mixture of linear SVMs, including the model formulation and the EM algorithm employed for training. Experimental results are given in Section 4, followed by conclusions in Section 5.

2 The SVM Classifier Model

In this paper, we focus our attention in binary classification problems. Multiclass classification can be handled by converting to a number of binary classification problems via the pairwise coupling framework [7]. Given the labelled binary data set $(\mathbf{x}_i, y_i)$, where $i = \{1, \ldots, N\}$, and $y_i \in \{-1, 1\}$, the linear SVM classifier recovers an optimal separating hyperplane which maximises the margin of the classifier. This can be formulated as the following constrained quadratic optimisation problem

$$\begin{cases} \text{minimise} & \dfrac{||\mathbf{w}||^2}{2} + C\sum_i \epsilon_i \\ \text{subject to} & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \epsilon_i \qquad i = \{1, \ldots, N\} \end{cases} \tag{1}$$

where $\mathbf{w}$ and b are the slope vector and the intercept of the hyperplane $y = \mathbf{w}^T\mathbf{x} + b$ respectively. The first term on the RHS of the above cost function is related to the classifier margin, which is the distance between the marginal hyperplanes $\mathbf{w}^T\mathbf{x} + b = +1$ and $\mathbf{w}^T\mathbf{x} + b = -1$. The second term on the RHS is related to the classification error. The parameter C controls the relative importance of the regularisation term and the error term.

Equation 1 is the primal form of the SVM formulation. It can be translated into the following dual form

$$\begin{cases} \text{maximise} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} & \sum_{i=1}^{N} \alpha_i y_i = 1 \text{ and } 0 \leq \alpha_i \leq C \qquad i = \{1, \ldots, N\} \end{cases} \tag{2}$$

The transformation above is important since it permits the original problem of recovering the hyperplane parameters to be reduced to the solution of the Lagrange multipliers α_i's, which is a standard QP problem. The condition for the primal problem in Equation 1 also implies that $w = \sum_{\alpha_i} \alpha_i y_i \mathbf{x}_i$. Hence, the optimal separating hyperplane is represented by

$$f(\mathbf{x}) = \sum_{\alpha_i} w^T x + b = \sum_{\alpha_i} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b \tag{3}$$

Thus, testing points are classified according to the sign of the function above. The points $\mathbf{x}_i$ for which the associated $\alpha_i > 0$ are called support vectors. Geometrically, these are the points lying on or outside the marginal hyperplanes with $y_i(\mathbf{w}^T\mathbf{x_i} + b) \leq 1$.

The linear SVM can be generalised to the nonlinear one by exploiting a nonlinear mapping from the input space to the high dimensional feature space known as the kernel trick. This is done by using an implicit mapping, i.e. a mapping whose closed form $x \rightarrow \Phi(x)$ is usually unknown. This mapping can be characterised by the inner product between any two vectors in the nonlinear feature space, which can be computed efficiently via the kernel function $K(x_i, x_j) = \Phi_i(x)^T \Phi_j(x)$. Compared to the dual form of the linear SVM in Equation 2, the dual problem for the nonlinear SVM is defined by substituting the inner product $\mathbf{x}_i^T \mathbf{x}_j$ with $K(\mathbf{x}_i, \mathbf{x}_j)$ as follows

$$\begin{cases} \text{maximise} & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} & \sum_{i=1}^{N} \alpha_i y_i = 1 \text{ and } 0 \leq \alpha_i \leq C \qquad i = \{1, \ldots, N\} \end{cases} \tag{4}$$

The optimal separating hyperplane is then represented by

$$f(\mathbf{x}) = \sum_{\alpha_i} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{5}$$

A novel testing point is then assigned to the positive class if $f(\mathbf{x}) > 0$ and otherwise to the negative class.

There are a number of alternatives for kernels used in nonlinear SVMs. These define the form of nonlinear mapping. The most common ones include Radial

Basis Functions (RBFs), polynomial and the Sigmoidal kernels. In this paper, we adopt the Radial Basis Functions (RBFs) with only one free parameter to tune. The kernel function is hence given by $\mathbf{K}(\mathbf{x}_j, \mathbf{x}_i) = \exp\left(-\beta\|\mathbf{x}_j - \mathbf{x}_i\|^2\right)$ where β is a scale parameter.

It is worth noting that, compared to their non-linear counterparts, linear SVMs cannot handle data sets that are linearly inseparable. Despite this apparent disadvantage, there are two major reasons to be in favor of the use of linear SVMs. Firstly, they are much more efficient at training time for large data sets. This is because training of linear SVMs can be achieved by minimising the cost function directly in the primal form in Equation 1, while nonlinear SVMs must be trained by minimising the cost function in the dual form as given in Equation 4. Efficient algorithms exist for training linear SVMs in linear time, respective of the size of training data set [8]. Secondly and more importantly, in general, the prediction time of linear SVMs is much lower than that of nonlinear SVMs. The efficiency of a nonlinear SVM in testing depends much on the number of support vectors. For a novel testing point, the kernel function, which is highly non-linear for the RBF function, needs to be evaluated m times to calculate the value of $f(\mathbf{x})$ in Equation 5. This can be very time consuming if there are thousands or millions of data points to classify in the testing set. In the other hand, the linear SVM only involves one inner product operation and one summation at testing time. This is since the weight vector $\mathbf{w}$ in Equation 3 can be explicitly expressed as the linear combination of support vectors. Hence it is desirable to develop an SVM based model with both the efficiency of the linear SVM at the prediction stage and the classification accuracy of the nonlinear SVM classifier.

3 Mixture of Linear SVMs

3.1 Model Formulation

As mentioned earlier, here we present a mixture of linear SVMs for classification making use of a mixture of experts model [2,3]. The architecture of the model is illustrated in Figure 1. The model consists of two parts. The experts at the bottom level are the linear SVM classifiers. The gating network produces a data-dependent weight distribution to combine the output of the linear SVM classifiers. The larger the weight, the more the corresponding component classifiers contribute to the classification of data.

Compared to conventional probabilistic classifier models like Linear Discriminant Analysis (LDA) or the logistic regressor, linear SVMs have better generalisation performance due to their large margin nature and are much cheaper to train thanks to the availability of efficient training methods [8]. However, to be able to use linear SVMs within the mixture of experts framework, we must be able to interpret it from a probabilistic point of view. To this end, we view the constrained quadratic optimisation problem in Equation 1 as that corresponding to the negative log-likelihood of an energy functional whose first term on the RHS corresponds to the prior probability of the parameters $P(w, b)$ and the

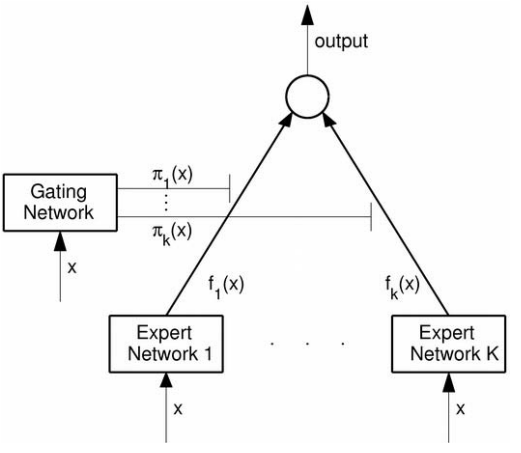

Fig. 1. Framework of Mixture of Experts

second term corresponds to the conditional probability $P(Y|X, w, b)$ related to classification errors. These are given by

$$P(w,b) = e^{-\gamma||w||^2} \tag{6}$$
$$P(Y|X,w,b) = \prod_i P(y_i|\mathbf{x}_i, w, b) = \prod_i e^{-\max(1-y_i(\mathbf{w}^T\mathbf{x}_i+b),0)}$$

where $\gamma = \log(\frac{1}{2C})$ and $\max(1, y_i(\mathbf{w}^T\mathbf{x}_i + b), 0)$ is the Hinge loss of the classification error. The loss is 0 for datapoint $\mathbf{x}_i$ if and only if it has a margin greater or equal to unity, i.e. $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$. Here we assume the classification errors are independent and identically distributed for the conditional probability term.

With the above probabilistic interpretation of linear SVMs, we now introduce the mixture model. Consider K linear SVMs whose parameters are denoted by $\theta_k = \{\mathbf{w}_k, b_k\}$. By combining these SVMs, we can then obtain the log-likelihood function of the linear SVM mixture model by

$$l(\theta) = \sum_{i=1}^{N} \log \sum_{j=1}^{K} \pi_j(\mathbf{x}_i) P(y_i|\mathbf{x}_i, \theta_j) + \sum_{j=1}^{K} P(\theta_j) \tag{7}$$

where $\pi_j(\mathbf{x}_i|\mathbf{v}_j)$ is the jth output of the gating network for datapoint $\mathbf{x}_i$ with parameter $\mathbf{v}_j$. Unlike the conditional mixture of Gaussian models, the mixing weights π here are data-dependent. This allows for more flexibility in the modeling. Following Jordan and Jacobs [3], we specify the function $\pi_j(\mathbf{x})$ with the following multinomial log-linear model given by

$$\pi_j(\mathbf{x}|\mathbf{v}_j) = \frac{e^{\beta \mathbf{v}_j^T \mathbf{x}}}{\sum_j e^{\beta \mathbf{v}_j^T \mathbf{x}}} \tag{8}$$

The reasons for the above formulation of the gating function are twofold. Firstly, the above function corresponds to a soft-max operator, where the soft nature is controlled by the parameter β. The smaller the value of β, the "softer" the assignment. In the extreme case when $\beta = 0$, the above gating function will return equal weights $\frac{1}{K}$ for all the components. On the other hand, when β approaches infinity, the gating function returns 1 for the component with maximum value of $\mathbf{v}_j^T\mathbf{x}$ and 0 otherwise. This is equivalent to dividing the whole space into several regions with piecewise linear boundaries, where each region is processed by the corresponding linear classifier.

3.2 The EM Algorithm

The mixture model of linear SVMs presented in the previous section can be trained efficiently using the EM algorithm. The E-step updates the posterior probabilities of assigning each datapoint to the experts. The M-step involves simultaneously updating the gate networks and SVM parameters so as to solve two independent optimisation problems. The parameters for the gate networks are updated by solving a convex optimisation problem. This can be done efficiently by a quasi-Netwon method. The expert components, which are comprised by the linear SVMs, are retrained with sample weights specified by the posterior probabilities computed in the E-step.

Suppose the parameter estimate at iteration t is given by $\theta^{(t)} = \{\mathbf{w}^{(t)}, b^{(t)}\}$. In the E-step, the posterior probabilities of assigning the datapoint $\mathbf{x}_i$ to the jth expert, i.e. the jth linear SVM classifier, is given by

$$q_{i,j}^{(t)} = \frac{\pi_j(\mathbf{x}_i)e^{-\max(1-y_i(\mathbf{w}^T\mathbf{x}_i+b),0)}}{\sum_j \pi_j(\mathbf{x}_i)e^{-\max(1-y_i(\mathbf{w}^T\mathbf{x}_i+b),0)}} \tag{9}$$

With the posterior probabilities, we are able to derive a lower bound on the log-likelihood in the following fashion

$$Q(\theta,\theta^{(t)}) = \sum_i\sum_j q_{i,j}^{(t)} \log \pi_j(\mathbf{x}_i)P(y_i|\mathbf{x}_i,\theta) - \sum_i\sum_j q_{i,j}^{(t)} \log q_{i,j}^{(t)} + \sum_{j=1}^{K} P(\theta_j) \tag{10}$$

where $\theta^{(t)}$ on the LHS of the equation indicates the dependency of the lower bound function Q on the current parameter estimate, which directly influences the posterior probability $q_{i,j}^{(t)}$.

It can be shown that the gap between the true log-likelihood function and its corresponding lower bound is given by the sum of Kullback-Leibler divergences between $q_i^{(t)}$ and q_i as follows

$$l(\theta) - Q(\theta,\theta^{(t)}) = \sum_i\sum_j q_{i,j}^{(t)} \log \frac{q_{i,j}^{(t)}}{q_{i,j}} \tag{11}$$

$$q_{i,j} = \frac{\pi_j(\mathbf{x}_i)P(y_i|\mathbf{x}_i,\theta)}{\sum_j \pi_j(\mathbf{x}_i)P(y_i|\mathbf{x}_i,\theta)}$$

where q_i gives the posterior probability distribution of point i estimated from the parameters θ. The gap is non-negative and varnishes if and only if $\theta = \theta^{(t)}$.

Moreover, the observation above implies that $l(\theta) \geq Q(\theta, \theta^{(t)}$ and $l(\theta^{(t)}) = Q(\theta^{(t)}, \theta^{(t)})$. In the statistical literature, Equation 10 is also referred to as the complete log-likelihood function by viewing the classifier membership as hidden variables whose distributions are given by the posterior probabilities in Equation 9. Minimising the log-likelihood in Equation 7 directly is a hard optimisation problem. It is usually more tractable to minimise the complete log-likelihood as we do in the M-step. We need to update both, the parameters of the gating function π_j and each linear SVM classifier. Moreover, since the gating unit and the linear SVM experts are decoupled in the complete log-likelihood cost function in Equation 10, we are able to solve them individually through two separate optimisation procedures.

For the update of the parameters v_j in the gating function, we only need to maximise the following function

$$\max \sum_i \sum_j q_{i,j}^{(t)} \log \pi_j(\mathbf{x}_i | v_j) \tag{12}$$

where π_j is the multinominal log-linear model as defined in Equation 8. This is a weighted logistic regression problem with concave log-likelihood function, which can be efficiently maximised via standard convex optimisation routines. Here, we employ a limited memory BFGS quasi-Newton method [9], which can be applied to large scale data sets.

To update the SVM classifiers, we need to retrain the linear SVMs with samples weighted by the corresponding posterior probability. Specifically, for the j^{th} linear classifier, we aim at maximising the following log-likelihood function

$$\begin{aligned} \max \sum_i q_{i,j}^{(t)} \quad & \log P(y_i | \mathbf{x}_i, \theta_j) + \log P(\theta_j) \\ = & \max - \sum_i q_{i,j}^{(t)} \max(1 - y_i(\mathbf{w}_j^T \mathbf{x}_i + b_j)) - \lambda \frac{1}{2} ||\mathbf{w}_j||^2 \end{aligned} \tag{13}$$

where $\lambda = e^{\gamma}$. This is equivalent to the following minimisation problem for training linear SVMs over weighted samples

$$\begin{aligned} & \min C \sum_i q_{i,j}^{(t)} \epsilon_i + \frac{1}{2} ||\mathbf{w}_j||^2 \qquad C = \frac{1}{2\lambda} \\ & \text{s.t. } y_i(\mathbf{w}_j^T \mathbf{x}_i + b_j) \geq 1 - \epsilon_i \end{aligned} \tag{14}$$

It is quite straightforward to observe that, at each EM iteration, the log-likelihood increases by the chain of relations $f(\theta^{(t)}) = Q(\theta^{(t)}, \theta^{(t)}) \leq Q(\theta^{(t+1)}, \theta^{(t)}) \leq f(\theta^{(t+1)})$. Hence, by repeating the EM steps we can obtain a convergent solution of the original maximum likelihood estimation problem.

4 Experimental Results

In this section, we demonstrate the performance of our algorithm with two experiments. The first experiment was performed on synthetic data. The second experiment was on pixel-level skin classification in hyperspectral images. For both experiments, we compared the proposed mixture of linear SVM classifiers (MixLSVM) with two alternative methods, the conventional linear SVM (LSVM) and nonlinear SVM classifiers with radial basis function (RBF) kernel. We set the number of linear SVMs to be 3 empirically for our mixture model in both experiments. The code was implemented in Matlab and the libsvm [10] package was used for training SVMs. The SVM parameters are selected via 5-fold cross validation.

First we focus on the synthetic data. A two-class data set was constructed comprising 200 points in each class uniformly drawn from two shifted sine signals in the two-dimensional space and perturbed with Gaussian noise in the vertical direction. We then applied the classification methods under study to the synthetic data. The resulting decision boundaries corresponding to kernel SVM, linear SVM and our approach were shown in Figure 2 respectively, superimposed on the generated synthetic data, where samples from different classes were plotted in different colors. We can see that our linear SVM mixture model is able to capture nonlinear information in the data for classification by locally combining linear classifiers. On the other hand, a globally linear classifier like the linear SVM fails to separate linearly inseparable data.

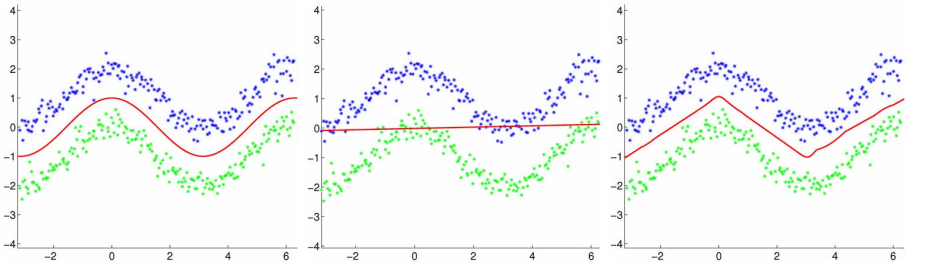

Fig. 2. From left to right, decision boundaries yielded by SVM, linear SVM and our approach superimposed on the synthetic data

Table 1. Testing result on the synthetic data

Data Size	Testing Error (%)			Testing Time		
	MixLSVM	SVM	LSVM	MixLSVM	SVM	LSVM
1K	**0.02**	0.01	9.28	0.011	0.022	0.006
10K	**0.02**	0.00	9.24	0.058	0.162	0.030
100K	**0.01**	0.00	9.26	0.562	1.687	0.323
1M	**0.01**	0.00	9.25	6.234	20.103	3.969

Table 2. Average cross validation error and testing time for skin classification

MixLSVM		SVM		LSVM	
Error (%)	Time (s)	Error (%)	Time (s)	Error (%)	Time (s)
4.95	**5.35**	4.54	76.25	8.66	3.65

Next, we generated testing data sets of different sizes from the same distribution and applied the trained classifiers for label prediction. The testing results are reported in Table 1. It can be seen that our algorithm achieves a desirable trade-off between classification performance and efficiency. It is more accurate than linear SVM with a much lower error rate and more efficient than kernel SVM, especially for large data sets containing millions of samples. Furthermore, the efficiency of kernel SVM for prediction replies highly on the number of support vectors. Since we only used a training set of 400 instances and about 1/3 of them are support vectors, it could even be worse if the kernel SVM is trained on a large training set. Both our algorithm and the linear SVM does not have such problem and the prediction efficiency only depends on the size of testing set.

We now turn our attention to skin detection in hyperspectral images. A set of hyperspectral images of human subjects was captured against a cluttered background by using a hyperspectral camera system comprised by a broad band monochrome camera and a Varispec liquid crystal tunable filter (LCTF). The spectral resolution of the LCTF is of 10nm between the 400nm to 720nm wavelengths, i.e. 33 bands covering the visible spectrum. The image spatial resolution is 696 by 520 pixels. The pseudo-colour images of some subjects are shown in the left-most column of Figure 3. For training, we have selected 2229 pixels in

Fig. 3. Example skin classification results. Left-most column: pseudo-color images of sample human subjects; Second, third and fourth columns: classification results delivered by our algorithm, kernel SVM and linear SVM

the skin region and 6808 pixels in the background region from the images. In order to obtain the feature vectors used at input for our mixture of experts, we have applied the photometric invariant preprocessing method in [11].

The performance of the three algorithms under study are reported in Table 2, including the average cross validation error on the training set as well as the testing time in seconds for each image. From the table, we again see that our algorithm achieves a similar performance as compared to kernel SVM while being much more efficient. The linear SVM, on the other hand, is quite efficient for testing, yet it is unable to separate nonlinear data in the feature space and has a larger classification error than that yielded by non-linear SVMs and our approach.

Some example skin detection results are shown in the last three columns of Figure 3. These correspond, from left-to-right to the results delivered by our approach, the kernel and linear SVMs. We can see that both, our approach and the kernel SVM yield skin maps with similar visual quality. These compare favourably to the results yielded by the linear SVM, which recovers more false positives than the other two methods.

5 Conclusions

In this paper we have proposed a novel method to assemble linear SVM classifiers. Here, we have used the mixture of experts framework and re-interpreted linear SVMs in a probabilistic framework. We have adopted the EM algorithm to learn the parameters of the classifiers. The mixture of classifiers can handle nonlinear data in an efficient manner. This is evidenced by the experiments on skin classification in hyperspectral imaging. As compared to non-linear and linear SVMs, our method provides a means to the computational efficiency of linear SVMs and the performance of non-linear classifiers.

References

1. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: ACM Conf. on Computational Learning Theory (1992)
2. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3, 79–87 (1991)
3. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the em algorithm. Neural Computation 6, 181–214 (1994)
4. Kwok, J.T.: Support vector mixture for classification and regression problems. In: Intl. Conf. on Pattern Recognition (1998)
5. Collobert, R., Bengio, S., Bengio, Y.: A parallel mixture of svms for very large scale problems. In: Advances in Neural Information Processing Systems (2002)
6. Kruger, S., Schaffoner, M., Katz, M., Andelic, E., Wendemuth, A.: Mixture of support vector machine for hmm based speech recognition. In: Intl. Conf. on Pattern Recognition (2006)
7. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: Advances in Neural Information Processing Systems, vol. 10 (1998)

8. Joachims, T.: Training linear svms in linear time. In: SIGKDD (2006)
9. Nocedal, J., Wright, S.: Numerical Optimization. Springer, Heidelberg (2000)
10. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using the second order information for training svm. Journal of Machine Learning Research 6, 1889–1918 (2005)
11. Fu, Z., Robles-Kelly, A., Tan, R., Caelli, T.: Invariant object material identification via discriminant learning on absorption features. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshop on Object Tracking and Classification Beyond the Visible Spectrum, pp. 140–147 (2006)