Mixing Linear SVMs for Nonlinear Classification

Zhouyu Fu, Antonio Robles-Kelly Senior Member, IEEE, Jun Zhou Member, IEEE

Abstract

In this paper, we address the problem of combining linear Support Vector Machines (SVMs) for classification of large-scale non-linear data sets. The motivation is to exploit both the efficiency of linear SVMs in learning and prediction and the power of non-linear SVMs in classification. To this end, we develop a linear SVM mixture model that exploits a divide-and-conquer strategy by partitioning the feature space into sub-regions of linearly separable data-points and learning a linear SVM for each of these regions. We do this implicitly by deriving a generative model over the joint data and label distributions. Consequently, we can impose priors on the mixing coefficients and do implicit model selection in a top-down manner during the parameter estimation process. This guarantees the sparsity of the learned model. Experimental results show that the proposed method can achieve the efficiency of linear SVMs in the prediction phase while still providing a classification performance comparable to non-linear SVMs.

Index Terms

EM Algorithm, Classification, Mixture of Experts, Model Selection, Support Vector Machines

Z. Fu is with the Gippsland School of Information Technology, Monash University, Churchill, Victoria 3842, Australia
A. Robles-Kelly, and J. Zhou are with the National ICT Australia (NICTA), Canberra, ACT 2601, Australia.
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.
Mixing Linear SVMs for Nonlinear Classification

I. INTRODUCTION

Classification is a fundamental problem in pattern analysis. It involves learning a function that separates data points from different classes. The support vector machine (SVM) classifier, which aims at recovering a maximal margin separating hyperplane in the feature space, is a powerful tool for classification and has demonstrated state-of-the-art performance in many problems [1]. SVMs can operate either explicitly in the input space leading to the linear SVM or implicitly in the feature space via the kernel mapping giving rise to the kernel SVM. Linear SVMs are simple to train and use, as they only involve inner product operations with the input data. However, they can be quite restricted in discriminative power and can not be applied to non-linear data. This limits their application to many real-world problems where the data distributions are inherently non-linear. The non-linear SVM, on the other hand, can handle linearly inseparable data but is not as efficient as the linear classifier. Its complexity is multiplied with the number of support vectors. This is unfavourable for prediction tasks on large-scale data sets.

Hence, it is desirable to have a classifier model with both the efficiency of linear SVMs and the power of non-linear SVMs. Here, we develop a mixture model so as to combine linear SVMs for the classification of non-linear data. The proposed mixture of linear SVMs (MLSVM) is based on a divide-and-conquer strategy that partitions the input space into hyper-spherical regions. Data in each region are linearly separable and can be handled by a linear SVM. We do this implicitly by deriving a probabilistic model over the joint data and label distributions. The joint distribution model enables us to explicitly incorporate mixture coefficients into the likelihood function. Consequently, we can perform structure learning and parameter learning simultaneously by imposing a proper prior on the mixture coefficients based on the minimum message length (MML) criterion [2]. Parameter update is then achieved by the EM algorithm [3]. For model selection, we start with an over-complete model and automatically prune the nodes corresponding to the vanishing mixture coefficients during the parameter optimisation process. Hence, structure learning is implicitly incorporated into the optimisation process and performed in a top-down manner. This is the major advantage of the proposed model as compared to alternative probabilistic models for classification, like the mixture of experts model by Jacobs and Jordan [4]. Moreover, unlike the mixture of experts, which uses linear gating networks with soft-max activation functions, we have adopted gating networks with non-linear radial basis functions (RBF). This provides more flexibility in the partitioning of the feature space. It leads to a smooth partition over the feature space that prevents abrupt change of local regions where different linear classifiers are applied.

The proposed method is quite general and can be used in combination with other choices of linear classifiers with minor modification in the M-step of the EM algorithm as long as they have proper probabilistic interpretations. Note that, in this paper, we focus on the case of large sample size with moderate feature dimension, since we are mainly interested in the problem of efficiency in prediction for non-linear data. High dimensional features are more
likely to be linearly separable and, therefore, linear classifiers are usually sufficient for them.

The remainder of the paper is organised as follows. Section II provides the background on the SVM classifier model. Section III describes the details of the proposed classifier model, including the model formulation, parameter estimation and model selection. Experimental results are presented in Section V, followed by conclusions in Section VI.

II. THE SVM CLASSIFIER MODEL

In this paper, we focus our attention on binary classification problems. Multiclass classification can be handled by converting the problem at hand into a number of binary classification ones through classifier fusion methods [5]. Given the labelled binary data set \((X, Y) = \{(x_i, y_i) | i = 1, \ldots, N, y_i \in \{-1, 1\}\}\), the linear SVM classifier recovers an optimal separating hyperplane \(w^T x + b = 0\) which maximises the margin of the classifier. This can be formulated as the following constrained optimisation problem [1], [6]

\[
\min_w \frac{||w||^2}{2} + C \sum_i \ell(w; x_i, y_i) \tag{1}
\]

The first term on the right-hand-side is the regularisation term on the classifier weights, which is related to the classifier margin through the inverse distance between the marginal hyperplanes \(w^T x = 1\) and \(w^T x = -1\). Without loss of generality, we have subsumed the bias term \(b\) in the above formulation by appending each instance with an additional dimension \(x_i^T = [x_i^T, 1]\) and \(w^T = [w^T, b]\). The second term on the right-hand-side is related to the classification error, where \(\ell(w; x_i, y_i) = \max (1 - y_i w^T x_i, 0)\) is the Hinge loss that upper bounds on the empirical error of the classifier. The parameter \(C\) controls the relative importance of the regularisation term and the error term.

SVMs can also be trained by solving the Lagrangian dual of Equation 1, which results:

\[
\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i \alpha_i x_i^T x_j y_j \alpha_j \tag{2}
\]

\[
s.t. \quad 0 \leq \alpha \leq C \quad \text{and} \quad \sum_i y_i \alpha_i = 0 \quad \forall i
\]

The classifier for linear SVM is then represented by

\[
f(x) = w^T x + b \tag{3}
\]

\[
= \sum_{\alpha_i > 0} \alpha_i y_i \langle x_i, x \rangle + b \tag{4}
\]

where \(w\) is the classifier weight vector defined by the dual variables. The weight vector can be computed explicitly by \(w = \sum_{\alpha_i > 0} \alpha_i y_i x_i\) and used for prediction.

The advantage of solving the dual form is that only inner products between data-points are needed. Consequently, we can derive the non-linear SVM by implicitly mapping the input data \(x\) into the feature space and training the SVM for the mapped features \(\phi(x)\). This is achieved by the kernel trick, where the implicit feature space is induced by the kernel function governed by the inner product for feature space maps \(K(x_i, x_j) = \phi(x_i)^T \phi(x_j)\). Non-linear
SVMs can then be trained by replacing the inner products in Equation 2 with the corresponding kernel $K(x_i, x_j)$. The resulting classifier for the non-linear SVM is then represented by

$$f(x) = \sum_{\alpha_i > 0} \alpha_i y_i K(x_i, x) + b$$

(5)

where the $\alpha$’s are the Lagrangian multipliers. Conceptually, the only difference between the non-linear SVM and their linear counterparts is in the use of kernel function instead of the inner product in Equation 4. Computationally, linear SVMs can be directly evaluated by using Equation 3, which is much more efficient than non-linear SVMs for purposes of prediction.

Note that only those instances with positive value of $\alpha_i$, called support vectors, will contribute to classification. Geometrically, these correspond to the points lying on or outside the marginal hyperplanes which incur nonzero hinge losses, i.e. $y_i f(x_i) < 1$ with $f(x)$. Different kernels can be used for non-linear SVMs. The most common ones include Radial Basis Functions (RBF), polynomial and the sigmoidal kernels. In this paper, we focus on the RBF kernel given below, which, in practice, is the most widely used for non-linear SVMs.

$$K(x_i, x_j) = \exp (-\tau \|x_i - x_j\|^2)$$

where $\tau$ is the scale parameter that controls the decay rate of the distance.

Despite the effectiveness in classification of non-linear data, the non-linear SVM has much higher computational complexity for prediction than its linear counterpart. According to Equation 5, it requires $|V|d$ summations and $|V|$ exponential operations to compute the kernel function values for evaluating the classifier output at a single testing instance, where $d$ is the data dimension and $|V|$ is the cardinality of the set of support vectors. Depending on the training data set size and the degree of non-linearity, prediction can be very time consuming. To cope with a highly non-linear decision boundary, a great number of support vectors may be generated in the training process, which leads to higher computational complexity in testing time. Also, a larger training data set will result in greater number of support vectors in training than a smaller data set does. This also adds up to the complexity of prediction. Hence non-linear SVM is inefficient for large-scale prediction tasks where there are thousands or millions of data points to classify in the testing set.

Linear SVMs, on the other hand, do not have such problem. They can perform prediction with $d$ summations via Equation 3 and the testing time is independent of the number of support vectors. This renders them a much more efficient alternative than non-linear SVMs. Moreover, there are very efficient algorithms for training linear SVMs which can run in linear time [7], [8]. This is impossible for non-linear SVMs, as the evaluation of the full kernel matrix is already quadratic in time complexity. Nevertheless, the performance of linear SVMs is usually suboptimal as compared to non-linear ones since they can not handle linearly inseparable data. Hence it is desirable to develop an SVM model with both the efficiency of the linear SVM at the prediction stage and the classification accuracy of the non-linear SVM classifier.
III. MIXTURE OF LINEAR SVM MODEL

A. Model Formulation

In this section, we present our Mixture of Linear SVMs model (MLSVM). Without loss of generality, we illustrate our model with a two-layer mixture model as shown in Figure 1(a). Generalisation to hierarchical mixtures is straightforward and will be discussed in Section III-E. The model in Figure 1(a) consists of two parts. The hidden layer is composed of the gating network that produces a soft-partition of the input space by generating a data-dependent weight distribution. Each node in the hidden layer is connected to a linear SVM classifier in the input layer, which is responsible for the classification of data points residing in the corresponding partition.

The proposed mixture model captures the following posterior probabilistic distributions of the labels $Y$ given input data $X$

$$
P(Y|X, \Theta) = \prod_i P(y_i|x_i, \Theta) = \prod_i \sum_{z_i} \xi_{z_i} P(z_i|x_i, \beta) P(y_i|z_i, x_i, \gamma)$$

(6)

where $i$ indexes correspond to the data samples and $\Theta = \{\xi, \beta, \gamma\}$ are the parameters of the underlying model.

In the expression above, we have assumed that the training instances/labels are independent and identically distributed and $z_i$ is the hidden variable introduced for the $i$th instance that takes values in $\{1, \ldots, K\}$, where $K$ is the total number of mixtures. The weight of the $i$th instance is determined by both $\xi_{z_i}$ and $P(z_i|x_i, \beta)$. The data independent term $\xi_{z_i}$ specifies the prior probability of the mixture component indexed by $z_i$, whereas the data dependent term $P(z_i|x_i, \beta)$ specifies the probability model for the gating function and is related to the importance of the $i$th instance with respect to the $z_i$th mixture component. $P(y_i|z_i, x_i, \gamma)$ is governed by the classification process for the $z_i$th component and specifies the posterior probability of the $i$th instance output by the corresponding linear SVM model. The parameters for the gating functions are given by $\beta$ and $\gamma$, respectively. We elaborate further on their specific parametric forms later in the paper.

Our MLSVM model can also be visualised by a graphical model as shown in Figure 1(b). In the figure, $x$ and $y$ are the target variables over which the probability model is defined, and $z$ is the hidden variable. The box containing $x$, $y$ and $z$ indicates i.i.d. copies of variables, where $N$ is the number of copies. The arrows indicate dependencies between the variables, where the variable pointed by an arrow is conditionally dependent on the variable where the arrow originates from.

The mixture model is generated from a multinomial distribution with parameter $\xi = \{\xi_1, \ldots, \xi_K\}$ for $K$-mixture. The posterior probability $P(z|x, \beta)$ is specified by the following equation

$$
P(z_i = j|x, \beta) = \frac{\exp \left(-\tau \|x - v_j\|^2 \right)}{\sum_j \exp \left(-\tau \|x - v_j\|^2 \right)}$$

(7)

The parameter $\beta = \{v_1, \ldots, v_K\}$ specifies the centroid of each component. The probability for each centroid is determined by the Euclidean distance between the instance and the centroid location. The closer the distance, the larger the probability. The assignment of instances to the components is probabilistic in nature, where $\tau$ is a scale parameter that controls the softness of the assignment. The smaller the value of $\tau$, the “softer” the assignment.
In the extreme case when $\tau = 0$, the gating function produces equal weights for all components. On the other hand, when $\tau$ approaches infinity, the gating function returns 1 for the closest component and 0 for the others. The probability $P(y|z, x, \gamma)$ is governed by the classifier model with parameter $\gamma$ conditional on $x$ and $z$, where $\gamma = \{w_1, \ldots, w_K\}$ and $w_j$ is the parameter for the $j$th linear SVM. We can clearly see the correspondence between the graphical model and the probability distribution in Equation 6.

It is worth noting that the proposed MLSVM model bears some resemblance with the classical mixture of expert model proposed by Jacobs and Jordan [4]. Nonetheless, there are two main differences. Firstly, our model is a mixture of weighted experts and more general in nature. This is since the mixture of experts model can be treated as a special case of ours with the mixing coefficients $\xi_j$ set to $\frac{1}{K}$. The explicit use of mixing coefficients enables us to perform implicit model selection via enforcing a sparseness prior over $\xi$ as we will discuss in the following section. For the mixture of expert model, however, there is no way to control the model structure by enforcing constraints on the parameters and we have to resort to heuristics for explicit model selection [9], [10]. Secondly, we have used a non-linear RBF gating function in Equation 7, whereas the mixture of expert model adopted a linear function for the gating network. Hence our model is more flexible in generating the partitions over the data space than the mixture of expert model. Note that an alternative mixture of expert model was proposed in [11] which also makes use of the RBF gating function by modelling the joint distributions of the data and labels in a generative manner. Hence, it is not directly comparable with discriminative models for classification such as the one proposed in this paper.

Following the maximum likelihood estimation (MLE) principle, the parameter estimation of our classification
model can be achieved by maximising the following log-likelihood function

\[ \mathcal{L}(\Theta) = \sum_i \log P(y_i|x_i, \Theta) + \sum_j \Omega(w_j) \]

\[ = \sum_i \log \sum_j \xi_j P(z_i = j|x_i, v_j) \cdot P(y_i|x_i, w_j) + \sum_j \Omega(w_j) \]

where \( \Omega(w_j) = \log \left( P(w_j) \right) \) specifies the log-prior term over the SVM parameters for regularisation purposes and \( P(z_i = j|x_i, v_j) \) is specified in Equation 7.

In order to incorporate linear SVMs into the above MLE framework, we have to reformulate it from a probabilistic viewpoint. To this end, we view the constrained quadratic optimisation problem in Equation 1 as that corresponding to the negative log-likelihood of the energy function whose first term on the right-hand side is related to the prior \( \Omega(w) \) and the second term corresponds to the conditional probability \( P(y|x, w) \) related to classification error. These are given by

\[ \Omega(w) = -\zeta ||w||^2 \]

\[ P(y_i|x_i, w) = e^{-\ell(w^T x_i + b)} \]

Here we have omitted the normalisation factor for the conditional probability \( P(y_i|x_i, w) \), which leads to an approximation of the probability measure. This is mainly for the convenience of numerical optimisation which enables us to use the existing fast linear SVM solvers [8] during parameter estimation. This simplification is particularly valid in the large margin case where the probability of misclassification is usually very small. More importantly, from an optimisation point of view, we can still apply the EM algorithm to optimise the likelihood function in Equation 8, which is guaranteed to increase in each EM iteration regardless of whether or not \( P(y_i|x_i, w_j) \) is a proper probability measure over \( y_i \) as we will discuss in the next section. The posterior probability is unity , i.e. zero loss, for instance \( x_i \) if and only if it has a margin greater or equal to unity, i.e. \( y_i(w^T x_i + b) \geq 1 \).

B. The EM Algorithm

Now, we turn our attention to the use of the EM algorithm for solving the mixture of linear SVM model presented in the previous section. The E-step updates the posterior probabilities of assigning each instance to the component classifiers. Let \( \Theta^{(t)} = \{\theta_j^{(t)}\}_{j=1,\ldots,K} = \{\xi_j^{(t)}, v_j^{(t)}, w_j^{(t)}\}_{j=1,\ldots,K} \) be the parameters at the current iteration, the probability of assigning the \( i \)th instance to the \( j \)th classifier at iteration \( t \) is given by

\[ q_{i,j}^{(t)} = \frac{\xi_j^{(t)} P(z_i = j|x_i, v_j^{(t)}) \cdot P(y_i|x_i, w_j^{(t)})}{\sum_j \xi_j^{(t)} P(z_i = j|x_i, v_j^{(t)}) \cdot P(y_i|x_i, w_j^{(t)})} \]

where \( P(z_i = j|x_i, v_j^{(t)}) \) and \( P(y_i|x_i, w_j^{(t)}) \) are posterior probabilities associated with the gating networks and local experts as given by Equations 7 and 10, respectively.

The M-step involves simultaneously updating the parameters for the gating functions and SVM classifiers. This involves solving two independent optimisation problems. For the gating functions, the values of the mixing
coefficients $\xi_j$ at iteration $t+1$ can be obtained in closed form

$$\xi_j^{(t+1)} = \frac{\sum_i q_{i,j}^{(t)}}{N}$$  \hfill (12)

Estimation of the centroid parameters for the gating functions is equivalent to unconstrained optimisation. Specifically, for the $j$th mixture component, the centroid $v_j$ is updated by solving the following optimisation problem

$$v_j^{(t)} = \arg\min_{v_j} f(v_j^{(t)}) = \arg\min_{v_j} \left\{ -\sum_{i,j} q_{i,j}^{(t)} \log P(z_i = j|x_i, v_j^{(t)}) \right\}$$  \hfill (13)

where $P(z_i = j|x_i, v_j)$ is defined in Equation 7. The above problem is very similar to parameter estimation for logistic regression and can be tackled by widely available numerical routines for unconstrained optimisation [12]. Here, we adopt a limited memory BFGS (L-BFGS) method suited for large scale optimisation [13] to minimise the cost function in Equation 13. The L-BFGS method only requires the gradient information of the cost function for the past few iterations, which can be calculated by the equation below

$$\nabla v_j f(v_j^{(t)}) = 2\beta \sum_i (q_{i,j}^{(t)} - P(z_i = j|x_i, v_j^{(t)}))(v_j^{(t)} - x_i)$$  \hfill (14)

Furthermore, it is worth noting that we do not have to solve the optimisation problem associated with the parameters $v_j^{(t)}$ in Equation 12 for each iteration. We can initialise the L-BFGS method for each iteration with the values $v_j^{(t-1)}$, since the parameters change slowly across iterations. This can greatly reduce the time for parameter estimation and speed up the training process.

Parameter estimation for the linear SVMs reduces to updating the classifiers for reweighted samples where the weights are specified by the posterior probabilities computed in the E-step. Specifically, for the $j$th linear classifier, we solve the following classification problem

$$\max \left\{ \sum_i q_{i,j}^{(t)} \log P(y_i|x_i, \theta_j^{(t)}) + \log P(\theta_j^{(t)}) \right\} = \max \left\{ -\sum_i q_{i,j}^{(t)} \ell(x_i, y_i) - \frac{1}{2} ||w_j^{(t)}||^2 \right\}$$  \hfill (15)

This is exactly the same problem as the training of linear SVMs in Equation 1 with sample weights given by $q_{i,j}^{(t)}$ and $C = \frac{1}{2\xi}$. Again, for every iteration, we can train the weighted linear SVMs in incremental manner via updating the results from the previous iteration. Since we have used a dual coordinate descent method for solving linear SVMs, we have retained the dual variables as well as status variables for each linear SVM in each iteration for their use by the corresponding linear SVM for the next iteration. This can be done via trivial adaption of the original training algorithm and can greatly reduce the time spent for linear SVM training to a great extent.

Each EM iteration increases the log-likelihood given by Equation 8. This argument can be easily established by making use of the auxiliary function parameterised with respect to $\theta^{(t)}$ given by

$$Q(\Theta^{(t)}; \Theta^{(t-1)}) = \sum_{i,j} q_{i,j}^{(t)} \log \xi_j P(z_i = j|x_i, v_j^{(t)}) P(y_i|x_i, w_j^{(t)}) - \sum_{i,j} q_{i,j}^{(t-1)} \log q_{i,j}^{(t-1)} + \sum_j \Omega(w_j^{(t-1)})$$  \hfill (16)

which is the lower bound of $L(\theta^{(t)})$ since

$$L(\theta^{(t)}) - Q(\Theta^{(t)}; \Theta^{(t-1)}) = q_{i,j}^{(t-1)} \log \frac{q_{i,j}^{(t-1)}}{q_{i,j}^{(t)}}$$  \hfill (17)
The gap is non-negative and varnishes if and only if $\theta(t) = \theta(t-1)$. Hence the log-likelihood increases with the following relation

$$g(\theta(t)) \geq Q(\Theta(t) ; \Theta^{(t-1)}) \geq Q(\Theta^{(t-1)} , \Theta^{(t-1)}) = g(\theta(t-1))$$

The second inequality is true due to the maximisation step. Therefore, by repeating the EM steps in an interleaved manner, we can obtain a convergent solution of the original maximum likelihood estimation problem.

C. Model Selection

Model selection is a challenging problem for learning mixture models, which involves estimating the number of mixture components. The dominant approach for doing so relies on optimising certain criteria via a search over a range of number of components $K \in [k_{\text{min}}, k_{\text{max}}]$ such that

$$\hat{k} = \arg \min_K \left\{ -\log P(X | \Theta_K) + R(K) \right\}$$

where we have written $\Theta_K$ so as to denote the estimate for the candidate value $\Theta$ when the number of components is set to $K$ and $R(K)$ is a function that penalises large values of $K$. The first term on the right-hand side is related to MLE, whereas the second term is related to model complexity. Examples of criteria for complexity that have been used for mixtures include Bayesian inference criterion (BIC), minimum description length (MDL), minimum message length (MML) and so on [2].

One major disadvantage of the above deterministic approach is that parameter estimation and model evaluation are performed independently, and one may have to do an exhaustive search over a large interval of $k$ and run separate parameter estimation process for each of them to obtain an optimal estimate. To sidestep this difficulty, we adopt an alternative approach that couples parameter estimation and model selection in a unified framework. We start with an over-complete model with a large number of components and implicitly select the components in a top-down manner. This is achieved by enforcing priors on the mixing coefficients $\xi_j$, where the components with vanishing coefficients are annihilated automatically in the training process. Following [14], we adopt the Dirichlet-type prior for $\xi_j$ which is given by

$$P(\xi) \propto \exp \left(-\nu \sum_{j=1}^{K} \log \xi_j \right) \quad (18)$$

where $\nu$ is a trade-off parameter for the balance between the data and the complexity terms. Incorporating it into the log-likelihood function in Equation 8 leads to the following overall log-likelihood

$$\Gamma(\Theta) = \mathcal{L}(\Theta) + \log P(\xi) = \mathcal{L}(\Theta) - \nu \sum_j \log \xi_j \quad (19)$$

The advantages of using the above Dirichlet-type prior are two-fold. Firstly, it enforces sparseness over $\xi$ and hence penalises complex models with many components. Secondly, it is the conjugate prior of multinomial random variables. Therefore $\xi$ can be computed in the following closed form

$$\xi_j = \frac{\max (0, \sum_i q_{i,j}^{(t)} - \nu)}{\sum_j \max (0, \sum_i q_{i,j}^{(t)} - \nu)} \quad (20)$$
Input: Labelled sample \( \{x_i, y_i | i = 1, \ldots, N \} \) and \( k_{\text{max}} \)

Output: Parameter \( \Theta = \{\xi_j, v_j, w_j | j = 1, \ldots, k \} \)

- Initialise \( \Theta \) for \( k = k_{\text{max}} \).
- E-step: calculate \( q_{i,j} \) via Equation 11.
- M-step:
  - update \( \xi_j \)'s via Equation 20.
  - update \( v_j \) via Equation 13 for all \( j \) such that \( \xi_j \neq 0 \).
  - update classifiers \( w_j \)'s for all \( j \) such that \( \xi_j \neq 0 \) by re-training linear SVM in Equation 15.
- Repeat the above two steps until convergence.

Fig. 2. Algorithm for our Mixture of Linear SVMs

It is worth noting that here the only difference from the EM algorithm developed in the previous section is in the estimation of \( \xi_j \in \xi \).

D. Algorithm and Implementation

We now present the step-sequence of the MLSVM algorithm in Figure 2. We start with a relatively large \( k_{\text{max}} \) and apply component elimination in the parameter estimation process. K-means clustering is used to initialise EM. Specifically, we divide all training instances into \( K \) clusters and apply a linear SVM to instances from each cluster. The cluster centroids are initial values of \( v \)'s for the gating function in Equation 7 and initial SVM parameters yielded by the linear SVM applied to each cluster. In the case that any cluster only contains instances from a single class, one can initialise the corresponding SVM weight to \( e_{d+1} \) for clusters containing only positive instances and \(-e_{d+1}\) for those clusters comprised of negative instances, where \( e_{d+1} \) is a \( d + 1 \)-dimensional vector whose entry indexed \( d + 1 \) is unity and 0 for all others. In each M step, the SVM classifiers are updated from the weights from the previous iteration. This is much faster than training from random initialisation since, in practice, the classifiers do not change abruptly between subsequent iterations.

There are two free parameters to be tuned in our model, the regularisation parameter \( C \) for the linear SVM, and the regularisation parameter \( \nu \) for the prior term over \( \xi \). These two parameters control the complexity of the linear SVM as well as the overall mixture model and can be chosen via cross validation.

In the testing stage, a novel testing data-point is classified by the weighted average over the classifier mixture such that

\[
\text{sign}(C(x)) = \text{sign} \left[ \sum_{\xi_j \neq 0} \xi_j P(z_i = j | x, v_j) \left( e_{\text{max}(0,1-w_j^T x)} - e_{\text{max}(0,1+w_j^T x)} \right) \right]
\]  

(21)
Despite the complicated form of the discriminant function, it is straightforward to see that it has a low computational complexity which is linear in the number of mixture components with non-zero coefficients.

E. Extension to a Hierarchical Mixture

Similar to the extension from mixture of experts to the hierarchical mixture model (HME) [15], it is natural to extend MLSVM to the hierarchical mixture model (HMLSVM). For a three-layer network with two hidden layers of gating functions, we have the following probabilistic model,

\[
P(Y|X, \Theta) = \prod_n \sum_i \sum_j \xi_i \varrho_j P(r_n = j| x_n, u_i) P(z_n = i| x_n, v_{i,j}) P(y_n| x_n, w_{i,j})
\]  

(22)

where \( r_n \) and \( z_n \) denote the hidden variables for the selection of the gating units at the first and second hidden layers. In the equation above, \( \xi_i \) and \( \varrho_j \) are the data independent weights for the \( i \)th gating unit in the first layer and the \( j \)th gating unit in the second layer, which is connected to the \( i \)th unit in the first layer. The probabilities \( P(r_n = j| x_n, u_i) \) and \( P(z_n = i| x_n, v_{i,j}) \) specify the data dependent posteriors for the corresponding gating units.

Note that, for the hierarchical mixture above, we can still apply the EM algorithm to estimate the model parameters with slight modifications in the estimation of mixing coefficients. Model selection can also be achieved by imposing the same prior on the mixing coefficients, where a component is annihilated if its corresponding \( \xi_i \) or \( \varrho_j \) are negligible. In practice, the single-layer mixture model, which acts like a RBF kernel SVM, is capable of handling most classification problems arising in practical applications. On the other hand, the hierarchical model, despite its elegant theoretical underpinnings, is prone to over fitting because it is difficult to regularise the model. Hence, in this paper, we focus on the use of a single-layer MLSVM.

IV. DISCUSSION

The motivation of the work presented here is quite different from that elsewhere in the literature. Competing approaches [16], [17], [18] employ a divide-and-conquer strategy similar in spirit to the mixture of expert framework [4] in the partitioning of 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 create the efficiency for prediction due to the usage of non-linear SVMs as local experts. A straightforward remedy might be to replace the non-linear SVMs with their linear counterparts.

Note that, in general there is no guarantee that the training data in each region are linearly separable. Moreover, as mentioned in Section III-C, model selection is a hard problem for the HME. Previous methods for model selection applied to the mixture of experts [4] and its hierarchical extension [15] has been addressed by several authors [9], [19], [20], [10]. However, all of these approaches are focused on the use of heuristic bottom-up procedures that start with a small network and grow it adaptively for learning the topology of the network. This process can hardly be automated and, thus, in practice, a fixed number of components is usually used in mixture of expert learning.

The work presented in this paper is most similar in nature to the linear SVM mixture model proposed in [21],
which is directly based on the mixture of experts model [4] for linear SVMs. To the best of our knowledge, the problem of locally linear learning for fast prediction with automatic model selection has not yet been investigated.

Along a different line of research, approximation based methods have been widely used for non-linear SVMs to accelerate the process of training them for large-scale data. Such methods are based on approximating the kernel matrix with the outer product of linear vectors spanned by a set of basis functions. The bases can be obtained by using low-rank decomposition methods like incomplete Cholesky factorisation [22], [23] or the Nyström approximation [24]. Alternatively, they can also be chosen randomly following certain principles [25], [26]. Despite effective in speeding up the training process, kernel approximation methods come at an expense for the efficiency in testing. This is due to the fact that the linear features are obtained on top of the kernel matrix. Hence, the full kernel matrix needs to be computed in the testing phase, which breaks the sparsity of support vectors and further slows down the prediction process.

Another natural choice for combining linear classifiers for non-linear classification is ensemble methods such as bagging [27], boosting [28], and random forests [29]. These methods are quite effective in learning combinations of weak classifiers for classification of data with complicated distributions, yet they do not enforce sparseness on the hypothesis space. Therefore, when linear classifiers are used as the base learner, ensemble methods tend to use a greater number of them for difficult cases compared to our approach and thus are likely to increase the time spent on prediction. Moreover, ensemble methods are best combined with unstable base learners, such as decision trees, to fully exploit different random natures of the weak learners for combination. For stable classifier models like SVMs and artificial neural networks, the training process is quite robust to data perturbation. Thus, the classifiers obtained at different stages are likely to be highly correlated. This would not only increase the number of iterations required for convergence, but more importantly, would negatively affect the performance of the combination scheme with such classifiers used as weak learners.

V. Experiments

In this section, we demonstrate the performance of the proposed algorithm on both synthetic and real-world data. Two real-world applications are considered, namely optical character recognition (OCR) and hyperspectral image classification. We compare our MLSVM algorithm with linear SVMs (LSVM), non-linear SVMs with the RBF kernel, and boosted linear SVMs. The boosting algorithm used here is AdaBoost [28]. Unless otherwise stated, in our framework we have set the initial number of linear SVMs to 10. The final number of linear models, however, might be less than 10 due to the implicit model selection ability of the proposed MLSVM algorithm in Figure 2. We empirically set the parameter $\tau$ in Equation 7 to unity. Note that $\tau$ may also be treated as an independent parameter whose value can be chosen via cross validation. In practice, we have experimented with different settings of $\tau$ and found that its value does not have much impact on the classification performance. We have chosen the other parameters for MLSVM and the alternatives via 5-fold cross validation. LibSVM [30] and LibLinear [8] were used for training kernel and the linear SVMs, respectively. For boosted linear SVMs, we have used the optimal
regularisation parameter obtained via cross validation and fixed the number of boosting iterations to 100.

Figure 3 shows the toy binary data-set used in our synthetic data experiment, where 200 positive instances are shown in red circles which occupy the circular region in the middle surrounded by 200 negative instances in green circles. The instances are chosen by subsampling a large sample of instances generated by uniform distribution within the square region. Clearly, the instances from these two classes are not linearly separable. The top row, from left to right, shows the decision boundary recovered by kernel SVMs, test errors of boosted SVMs over sample iterations, and the log-likelihood values of MLSVM as given by Equation 8. We have fixed the number of components to 4 for the MLSVM. For the non-linear SVM, the decision boundary was plotted in black. The decision boundaries with margin values of 1 corresponding to the two classes. The support vectors are shown in solid circles. Even for this simple example, the SVM with RBF kernel yielded 71 support vectors, which is not particularly efficient for large-scale prediction tasks. For the boosted SVMs, we can see that the training error reduces slowly and becomes stable only after a large number of iterations. This has confirmed our previous assertion that boosting requires many iterations to converge. For MLSVM, on the other hand, we can see that the log-likelihood value increases with every iteration and requires only a few iterations for convergence.

This is further justified by the plots in Figure 3. In the middle row of the figure, from left to right, we show the results of MLSVM at the initial, 5th and final iterations. In each plot, the digit in each circle indicates the index of the mixture component for the training instances. The shaded regions indicate negative classes, whereas positive classes are in white. It can be seen that the classification model learned by the 5th iteration is already in good accordance to that yielded by the final iteration. This is in contrast with the boosted linear SVM model, which is constantly evolving. This can be seen from the results recovered by the boosting model at the first, tenth, and final iterations as shown in the bottom row of the figure. Since boosting is a greedy strategy for classifier combination, it uses a much larger number of weak classifiers, most of which aim at correcting errors incurred in previous iterations. This has the counterproductive effect of cancelling the effects of one another. More importantly, unlike the MLSVM, each boosting iteration introduces a new classifier into the ensemble model. Hence, boosting has to combine a lot more linear classifiers than the proposed algorithm to yield comparable performance. Even for this simple data-set, boosting employs approximately 100 linear SVMs to achieve a comparable performance to that of MLSVM, which only used 4 classifiers.

To further demonstrate the power of MLSVM for the recovery of complex decision boundaries, we performed classification on a second synthetic data set comprised by two spirals intertwined with each other, as shown in Figure 4. Each spiral has five turns and consists of 500 instances. These are generated by subsampling a larger sample of instances uniformly distributed around the spiral curve given by the polar function $r = 1.3\theta(-1.3\theta)$ whose independent variable is $\theta \in [0.5\pi, 5.5\pi]$. This is a hard classification problem due to the high non-linearity of the data set. We have trained our MLSVM classifier with 20 linear classifiers. The top row, from left to right, shows the results obtained by MLSVM in the initial, tenth and final iterations. In each plot, the digit in each circle indicates the index of the mixture component for the training instances. The shaded regions indicate negative
Fig. 3. MLSVM vs. boosting on synthetic two-class data. Top row, from left to right: decision boundary of non-linear SVM, test error of boosted SVMs, and log-likelihood of our algorithm as a function of iteration number; Middle row: results of our algorithm at the initial, fifth and final iterations; Bottom row: results for the boosted SVMs at the first, tenth and final iterations.

classes, whereas positive classes are in white. It can be seen that MLSVM can model rather complicated decision boundaries through mixing the linear classifiers.

Furthermore, the decision boundary of MLSVM is refined over iterations, from initially correctly classifying 95.7% of the data to 99.7% in the final iteration. This is further justified by the increased log-likelihood over iterations as shown by the rightmost plot in the bottom row of the figure. The first two plots in the bottom row show the decision boundaries produced by RBF kernel based non-linear and linear SVMs respectively. The linear decision boundary returned by the linear SVM only achieves a random classification rate of 50%. This also invalidates the use of the boosted SVM model, as boosting requires at least better than random classification performance at each iteration to proceed. The non-linear SVM achieves 100% accuracy with a smoother decision boundary than that of MLSVM. However, this is at the cost of generating over 750 support vectors in the training process so as to approximate such complicated decision boundary. This accounts for over three quarters of the training set and about 20 times of the complexity of our MLSVM, which only uses 20 linear classifiers.\(^1\) For the two-dimensional spiral data set, it takes 70 milliseconds for the non-linear SVM prediction and 3.5 milliseconds for our MLSVM. The experiments were effected on an Intel 3.0GHz Core 2 Duo desktop with 4GB RAM. Training

\(^1\)Note that the complexity of MLSVM scales with twice the number of linear classifiers, including gate function and classifier evaluations.
MLSVM is more expensive than SVM, which takes 5 seconds, as compared to only 0.5 seconds for the linear SVM with a fully optimised LibSVM routine. However, training can be done offline for most pattern classification problems whereas prediction must be performed online. Hence online prediction efficiency is often more important especially for large-scale problems, which is the main focus of our MLSVM method.

Now we turn our attention to real-world data. We first demonstrate the performance of our algorithm for Optical Character Recognition (OCR) on the MNIST database [31]. The database contains images of handwritten digits from 0 to 9. We have divided them into two classes, one for the odd numbers and another one for the even digits. For each of these classes, the training set contains 6000 images and the test set contains 1000 binary images, i.e. 60000 images for training and 10000 for testing as a whole. The resolution of each image is $28 \times 28$, which leads to a 784-dimensional feature vector by concatenating the pixel values column-wise. We have trained the algorithms under comparison over subsets of different sizes drawn randomly from the training set and applied them to the testing set. We have repeated this procedure 10 times and recovered quantitative results regarding classification performance.

Table I shows the classification rates on the test set in terms of accuracy for MLSVM and the alternatives. It can be seen clearly that the non-linear SVM outperforms the other methods. It delivers the highest classification accuracy followed by our MLSVM model. The MLSVM is comparable in performance with non-linear SVM on small sample sizes and is better than linear SVMs regardless of the size of training data set. The performance of boosted linear SVMs is, however, unsatisfactory for the OCR task. For small training sample sizes, it achieves
The same performance as the linear SVM. This may be due to the fact that the randomly selected subset of the training data is most likely linearly separable, which leads to the early stopping of the boosting procedure in the first iterations. This is confirmed by the number of iterations used for boosting reported in Table II. For larger training sizes, the performance of boosting slightly decreases with increasing number of iterations, as reported in the last four rows of Table I. This suggests that boosted SVMs are not the model of choice for bridging between the complexity of non-linear SVMs and the performance of linear classifiers. On the contrary, boosting appears to have a higher complexity for increasing number of iterations and lower performance as compared to linear SVMs. Our MLSVM model, on the other hand, clearly outperforms linear SVMs for OCR and is much more efficient than the non-linear SVM. It is worth noting that although the training data are likely to be linearly separable for small sample sizes, their distributions are inherently non-linear. Hence, non-linear SVMs still achieve higher classification accuracy than their linear counterparts. Our MLSVM can implicitly exploit the non-linear nature of the data set and deliver better performance than the linear SVM for small sample sizes.

We also compare the complexity of the methods under study. Table II shows the complexity for boosted SVMs and MLSVM as measured in terms of number of linear classifiers used. For the non-linear SVM, we list the number of support vectors as a measure of complexity. Notice that, for different training data, SVMs may produce different number of support vectors. Our proposed MLSVM method may also select different number of linear classifiers via the training algorithm in Figure 2 with implicit model selection. For each training data size, we have listed the average, minimum and maximum number of linear classifiers/support vectors used over all the training rounds. From the table, note that that the complexity of the non-linear SVM grows with respect to increasing training data sizes. This is as the number of support vectors obtained in the training process is proportional to the size of training set. The proposed MLSVM model, on the other hand, is not affected by the size of the training data set. As can be seen from the table, it is much more efficient than the non-linear SVM. Further, it is orders of magnitude faster for large training data sizes. For instance, it takes the non-linear SVM 27, 46 and 195 seconds, in average, for the prediction of 10000 testing instances given training sets of sizes 5000, 10000 and all available data. On the other hand, the linear SVM takes, in average, only 10 milliseconds and MLSVM 0.7 seconds for the same prediction.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>LSVM (%)</th>
<th>SVM (%)</th>
<th>Bst-SVM (%)</th>
<th>MLSVM (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>80.62 ± 2.33</td>
<td>84.70 ± 1.21</td>
<td>80.62 ± 2.33</td>
<td>82.82 ± 1.97</td>
</tr>
<tr>
<td>500</td>
<td>82.71 ± 0.80</td>
<td>91.71 ± 0.96</td>
<td>82.71 ± 0.80</td>
<td>90.12 ± 1.43</td>
</tr>
<tr>
<td>1000</td>
<td>84.71 ± 0.53</td>
<td>94.59 ± 0.21</td>
<td>83.67 ± 0.24</td>
<td>92.82 ± 0.26</td>
</tr>
<tr>
<td>5000</td>
<td>88.47 ± 0.28</td>
<td>97.30 ± 0.16</td>
<td>87.40 ± 0.24</td>
<td>93.91 ± 0.33</td>
</tr>
<tr>
<td>10000</td>
<td>89.60 ± 0.13</td>
<td>98.01 ± 0.09</td>
<td>88.98 ± 0.21</td>
<td>94.89 ± 0.23</td>
</tr>
<tr>
<td>All</td>
<td>90.28</td>
<td>98.49</td>
<td>89.72</td>
<td>95.20</td>
</tr>
</tbody>
</table>

TABLE I
OCR performance. Comparison of linear SVM (LSVM), non-linear SVM (SVM), boosted SVMs (Bst-SVM) and the proposed MLSVM.
<table>
<thead>
<tr>
<th>Training Size</th>
<th>SVM mean</th>
<th>SVM min</th>
<th>SVM max</th>
<th>Bst-SVM mean</th>
<th>Bst-SVM min</th>
<th>Bst-SVM max</th>
<th>MLSVM mean</th>
<th>MLSVM min</th>
<th>MLSVM max</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>79.2</td>
<td>69</td>
<td>86</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5.2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>500</td>
<td>263.2</td>
<td>236</td>
<td>288</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>428.8</td>
<td>414</td>
<td>446</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>8.8</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>5000</td>
<td>1178.4</td>
<td>1157</td>
<td>1202</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>8.6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>10000</td>
<td>1817.0</td>
<td>1800</td>
<td>1833</td>
<td>61.8</td>
<td>24</td>
<td>100</td>
<td>8.6</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>All</td>
<td>3564</td>
<td></td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE II**

OCR Complexity. Number of support vectors used by the non-linear SVM (SVM) and number of classifiers employed by boosting (Bst-SVM) and the proposed MLSVM.

We repeated the experiments above on the forest covertype data set. This is a data set presented in [32] derived from data originally obtained from US Geological Survey (USGS) data by the Colorado State University. The data set contains 581012 instances from 7 cover types. Each instance has 54 feature attributes capturing cartographic information. The purpose is to predict various cover types from the cartographic variables. We converted this to a binary classification problem by predicting whether each instance belongs to cover type 2, the largest cover type, containing 283301 instances, i.e. approximately half the data set. As for the OCR experiments, we compare the performances of our MLSVM algorithm and the alternatives using different training data set sizes. Training instances were randomly drawn from the full data set and the rest were used for prediction. The procedure was repeated 10 times. The average accuracy rates along with standard deviations are shown in Table III. We show the time complexities in Table IV.

The trend on the forest cover prediction data is similar to that for the OCR. The non-linear SVM achieves the best performance, followed by the MLSVM. Both of them perform better than the Linear SVM. On the other hand, the non-linear SVM has the highest complexity for prediction. It is much more costly than the MLSVM and linear SVM. Moreover, the MLSVM achieves the optimal trade-off in prediction performance and complexity. Given

---

2We found the timing results for training quite unusual, with the non-linear SVM outperforming linear SVMs in training speed. This also greatly affects the speed of our MLSVM method, as it calls linear SVM solvers frequently for training. This is counterintuitive, since LibLinear, the solver we used for linear SVM training, should scale linearly with training sample sizes [8].

3The data set can be found at http://kdd.ics.uci.edu/databases/coverype/
TABLE III

PERFORMANCE COMPARISON OF THE LINEAR SVM (LSVM), NON-LINEAR SVM (SVM), BOOSTED SVMs (Bst-SVM) AND THE PROPOSED MLSVM ALGORITHM ON THE FOREST COVER-TYPE DATA SET.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>LSVM (%)</th>
<th>SVM (%)</th>
<th>Bst-SVM (%)</th>
<th>MLSVM (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>64.79 ± 3.52</td>
<td>66.19 ± 2.47</td>
<td>65.70 ± 2.86</td>
<td>65.98 ± 2.79</td>
</tr>
<tr>
<td>500</td>
<td>73.34 ± 1.10</td>
<td>73.05 ± 0.93</td>
<td>73.30 ± 1.50</td>
<td>73.62 ± 0.89</td>
</tr>
<tr>
<td>1000</td>
<td>74.78 ± 0.47</td>
<td>75.22 ± 0.43</td>
<td>74.76 ± 0.51</td>
<td>74.89 ± 0.36</td>
</tr>
<tr>
<td>5000</td>
<td>75.97 ± 0.20</td>
<td>78.92 ± 0.29</td>
<td>76.06 ± 0.25</td>
<td>77.12 ± 0.15</td>
</tr>
<tr>
<td>10000</td>
<td>76.19 ± 0.12</td>
<td>79.98 ± 0.16</td>
<td>76.13 ± 0.12</td>
<td>77.34 ± 0.18</td>
</tr>
</tbody>
</table>

TABLE IV

COMPLEXITY COMPARISON FOR THE FOREST COVER-TYPE DATA. NUMBER OF SUPPORT VECTORS USED BY THE NON-LINEAR SVM (SVM) AND NUMBER OF CLASSIFIERS EMPLOYED BY THE BOOSTING ALGORITHM (Bst-SVM) AND OUR PROPOSED MLSVM METHOD.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>SVM</th>
<th>Bst-SVM</th>
<th>MLSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>100</td>
<td>74.2</td>
<td>68</td>
<td>80</td>
</tr>
<tr>
<td>500</td>
<td>323.9</td>
<td>286</td>
<td>356</td>
</tr>
<tr>
<td>1000</td>
<td>604.9</td>
<td>571</td>
<td>635</td>
</tr>
<tr>
<td>5000</td>
<td>2643.4</td>
<td>2568</td>
<td>2743</td>
</tr>
<tr>
<td>10000</td>
<td>5028.6</td>
<td>4934</td>
<td>5121</td>
</tr>
</tbody>
</table>

sufficient training data, the cost of MLSVM is almost independent of the training data size. This is in contrast with the non-linear SVM, whose complexity scales with the number of training data used. This is further justified by the average testing speed. The non-linear SVM takes 40, 167 and 281 seconds for prediction given training set sizes of 1000, 5000 and 10000, respectively. Our MLSVM method takes 2 seconds and the linear SVM takes 50 milliseconds for prediction regardless of the sizes of the training data set. Boosted linear SVMs are inferior to the other methods being studied and do not achieve any improvement over linear SVM.

We now turn our attention to the application of our MLSVM algorithm to pixel-level skin classification in hyperspectral images. Here, we aim at exploiting the information-rich representation of the scene in hyperspectral imaging. To this end, we have used a set of 696 × 520 images of human subjects captured in-house against a cluttered background making use of a hyperspectral camera sensor. Each pixel in the image has 33 spectral channels covering the narrow band spectral reflectance in the visible range from 400 to 720nm in steps of 10nm.

Example pseudo-colour images of some of the subjects under study are shown in the top row of Figure 5. For purposes of training, we have selected patches enclosing the skin areas from a few images. The spectral feature vectors were then obtained using the photometric invariant preprocessing method in [33]. Since boosted SVMs do not produce satisfactory results in terms of either classification accuracy or efficiency of prediction, we have focused our comparison on the remaining three algorithms. Some example skin classification results are shown in
the two bottom rows of Figure 5. These correspond, from top-to-bottom, to the results yielded by the MLSVM, the non-linear SVM with RBF kernel and the linear SVM, respectively. We can see that the skin maps recovered by the MLSVM are quite similar to those yielded by the non-linear SVM, whereas the skin maps computed using the linear SVM are less accurate than those corresponding to the other two methods. Furthermore, from the figure, we can see that the skin maps returned by the linear SVM contain more false positives than those yielded by the MLSVM. This is particularly evident on the small regions in the background wallpaper, which have been falsely classified as skin.

In Table V, we show a more quantitative analysis of the performance for the alternatives under study. This includes the average testing accuracy as well as the average testing time in seconds per image. The average testing accuracy is measured by splitting the training data into two sets, one for training and the other one for testing. This is done by evenly splitting the data set and averaging the results over 5 different random partitions. From the table, we again notice that the MLSVM is much more efficient while achieving a performance comparable to that yielded by the non-linear SVM. The linear SVM, on the other hand, is quite efficient for testing but has a lower accuracy rate as compared to the non-linear SVM and our MLSVM approach.

<table>
<thead>
<tr>
<th></th>
<th>MLSVM</th>
<th>SVM</th>
<th>LSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>95.20 ± 0.30</td>
<td>95.45 ± 0.22</td>
<td>91.34 ± 0.15</td>
</tr>
<tr>
<td>Time (s)</td>
<td>6.55 ± 0.25</td>
<td>76.25 ± 1.30</td>
<td>1.65 ± 0.05</td>
</tr>
</tbody>
</table>

TABLE V
AVERAGE CROSS VALIDATION ERROR AND TESTING TIME FOR SKIN CLASSIFICATION.

VI. CONCLUSIONS

In this paper we have proposed a mixture of linear SVMs (MLSVM). This is a novel method to combine linear classifiers for the classification of non-linear data. We have addressed the problem by casting it into a probabilistic setting so as to implicitly perform model selection. This is done by using the EM algorithm on a generative model that permits the use of priors on the mixing coefficients. Our MLSVM method provides a means to efficient non-linear classification by presenting a top-down parameter estimation process. The method is quite general in nature and is particularly well suited for large-scale prediction tasks. We have done experiments on synthetic and real-world data and provided comparison to boosting, linear and non-linear SVMs.

There are a number of possible directions for future research. One potential application of the proposed framework is feature combination, where each example in the data set is represented by multiple feature vectors extracted from different sources. The standard approach to solving this problem with SVM is multiple kernel learning (MKL) [34], [35], where individual kernel matrices computed from different features are linearly combined. MKL is then formulated as a joint optimisation problem over the kernel combination weights and the SVM classifier. Unfortunately, testing can be computationally expensive for large data sets with many input feature types [35].
Fig. 5. Example skin classification results. Top row: pseudo-color images for sample human subjects; Second, third and fourth rows: classification results delivered by our algorithm, the non-linear SVM and linear SVM.

The method proposed here can learn a mixture of linear SVMs. This can be done separately for the individual features so as to combine the output scores. Another option is to combine feature types directly making use of the joint feature vectors. The mixture model can capture the nonlinear information in each individual feature while still maintaining efficiency in prediction. Another possible extension is the use of error generalisation bounds for parameter selection [36], [37]. This is especially effective for small sample size problems and provides a more accurate estimate of the prediction error than that given by cross validation.

VII. ACKNOWLEDGMENT

The work presented here was done while Z. Fu was with the Australian National University, Canberra, Australia.
REFERENCES


