Efficient Minimax Clustering Probability Machine by Generalized Probability Product Kernel

Haiqin Yang, Kaizhu Huang, Irwin King and Michael R. Lyu

***Abstract*— Minimax Probability Machine (MPM), learning a decision function by minimizing the maximum probability of misclassification, has demonstrated very promising performance in classification and regression. However, MPM is often challenged for its slow training and test procedures. Aiming to solve this problem, we propose an efficient model named Minimax Clustering Probability Machine (MCPM). Following many traditional methods, we represent training data points by several clusters. Different from these methods, a Generalized Probability Product Kernel is appropriately defined to grasp the inner distributional information over the clusters. Incorporating clustering information via a non-linear kernel, MCPM can fast train and test in classification problem with promising performance. Another appealing property of the proposed approach is that MCPM can still derive an explicit worst-case accuracy bound for the decision boundary. Experimental results on synthetic and real data validate the effectiveness of MCPM for classification while attaining high accuracy.**

I. Introduction

MINIMAX PROBABILITY MACHINE (MPM) is a recently proposed learning model and has demonstrated advantages in solving classification problem in the literature [10]. By minimizing the maximum probability of misclassification of future data points, MPM has shown competitive classification accuracy against the state-of-the-art classifier, Support Vector Machine (SVM). One appealing feature of MPM is that it can derive an explicit worst-case accuracy bound for the decision boundary. Following the idea of MPM, there have been many important extensions, e.g., the worst-case optimal Bayesian classification model [7], its regression extension [19], the Biased Minimax Probability Machine for imbalanced classification [5] and Medical Diagnosis [6].

However, MPM and its extensions are often challenged for the time-consuming training and test procedures. The training of MPM is equivalent to solving a Second Order Cone Programming (SOCP) problem, whose worst-case complexity is $O(n^3)$ (n is the number of the training samples for the kernelized MPM). The test complexity of the kernelized MPM is also related to the number of training samples. This makes the MPM-based models inefficient for classifying large datasets.

Haiqin Yang, Kaizhu Huang, Irwin King and Michael R. Lyu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong (email: {hqyang,kzhuang,king,lyu}@cse.cuhk.edu.hk).

The work described in this paper is supported by two grants, one from the CUHK Direct Grant #2050346, and the other from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4150/07E).

In solving large-scale classification problems, the state-of-the-art classifier, Support Vector Machine, also faces the same problem. Although various improvements, e.g., Sequential Minimal Optimization [16], [9], Parallel method [2], [4], have been made to speed up the training procedure in the SVMs, the training complexity of SVMs is still high when the number of training samples is large. To solve this problem, clustering-based SVMs, e.g., CB-SVMs [20], Support Clustering Machine (SCM) [21], [12], have been proposed to select representative quanta, e.g., typical points or clusters, for the SVMs training, so as to reduce the training complexity.

Motivated by the idea of clustering-based SVMs, we propose the Minimax Clustering Probability Machine (MCPM) to extend MPM for large-scale classification problems. The main idea of MCPM is as follows. The training samples are clustered in advance to output many generative models. Then the obtained clusters described by certain distributions are input as the training units, while the test samples are explained as special clusters centered on each specific data point. Instead of applying the probability product kernel to measure the similarity as used in [12], we define a novel generalized probability product kernel, especially, Radial Basis Functions on probability product kernel to measure the similarity either between any clusters (in training) or between a cluster and a test vector (in test). Finally, the decision function can be constructed in a kernel form, which is only related to the training clusters. Experiments on both synthetic and real data show that the proposed MCPM reduce the computational costs both in the training phase and the test phase, while preserving the classification accuracies.

The proposed novel generalized probability product kernel has a lot of advantages over the traditional probability product kernel as used in SCM [12]. First, as we show in the paper, the traditional probability product kernel is actually a linear kernel defined in the probability space, while our generalized kernel describes a non-linear kernel which can generate more complex similarity measures. Second, numerical problems such as the large variance in kernel matrix sometimes occur when the traditional probability product kernel is employed. These numerical problems often require careful data adaptation, e.g., scaling up the kernel matrix, making the training sometimes not as straightforward as expected. In contrast, the proposed generalized probability product kernel avoids such problems by projecting the probability into a non-linear space. The whole learning process is easy to implement and requires no data adaptation. Third, the generalized probability product kernel is more flexible in

978-1-4244-1821-3/08/$25.00©2008 IEEE

measuring the similarity. This is analogous to the case that non-linear RBF kernels can usually outperform the linear kernel. Hence the proposed generalized kernel defined over probabilities is often more accurate than the standard kernel. Empirical evidence on real data also support this statement as later seen in experiments.

The contributions of the paper are summarized as follows. (1) The proposed MCPM largely reduces both the computational and spatial costs for both training and test, while keeping the classification accuracy; (2) MCPM keeps the statistical information of training samples by presenting them in generative models; (3) by defining Radial Basis Functions on the probability product kernel, the similarity measurement could deliver more information for classification; (4) MCPM provides a worst-case accuracy bound for classifying future data points; (5) MCPM can be implemented easily by using the generalized kernel.

The rest of this paper is organized as follows. Section II derives the MCPM under a probability framework similar to that of the original MPM. Section III defines the probability product kernel and introduces a generalized probability product kernel to measure the similarity either between a pair of clusters or between a cluster and a test sample. Section IV reports the experimental setup and results on both synthetic dataset and real datasets. Finally, the paper is concluded in Section V.

II. Classification Model

In this section, we first give a sketch introduction to the MPM. We then formulate the MCPM in subsection II-B.

A. Minimax Probability Machine for Binary Classification

Considering a binary classification problem, suppose the data are generated from two classes of data, $\mathbf{x}$ and $\mathbf{y}$. And data of class $\mathbf{x}$ are drawn from a class of distributions with mean and covariance matrices as $\{\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}\}$, while data of class $\mathbf{y}$ are from another class of distributions with mean and covariance matrices as $\{\bar{\mathbf{y}}, \Sigma_{\mathbf{y}}\}$, where $\mathbf{x}$, $\mathbf{y}$, $\bar{\mathbf{x}}$, $\bar{\mathbf{y}} \in \mathbb{R}^d$, and $\Sigma_{\mathbf{x}}$, $\Sigma_{\mathbf{y}} \in \mathbb{R}^{d\times d}$.

Assuming $\{\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}\}$, $\{\bar{\mathbf{y}}, \Sigma_{\mathbf{y}}\}$ for two classes of data are reliable, MPM attempts to determine the hyperplane $\mathcal{H}(\mathbf{a}, b) = \{\mathbf{z} \,|\, \mathbf{a}^\top \mathbf{z} = b\}$ ($\mathbf{a} \in \mathbb{R}^d \backslash \{\mathbf{0}\}$, and $b \in \mathbb{R}$, and the superscript $\top$ denotes the transpose) by separating two classes of data with the maximal probability. The formulation of the MPM model is written as follows:

$$\max_{\alpha, \mathbf{a}\neq\mathbf{0}, b} \quad \alpha \quad \text{s.t.} \quad \inf_{\mathbf{x}\sim\{\bar{\mathbf{x}}, \Sigma_{\mathbf{x}}\}} \mathbf{Pr}\{\mathbf{a}^\top \mathbf{x} \geq b\} \geq \alpha, \quad (1)$$
$$\inf_{\mathbf{y}\sim\{\bar{\mathbf{y}}, \Sigma_{\mathbf{y}}\}} \mathbf{Pr}\{\mathbf{a}^\top \mathbf{y} \leq b\} \geq \alpha,$$

where α represents the worst-case accuracy of classifying future data points. Future points $\mathbf{z}$ when $\mathbf{a}^\top \mathbf{z} \geq b$ are then classified as belonging to the class associated with $\mathbf{x}$, otherwise they are judged as belonging to the class associated with $\mathbf{y}$. This derived decision hyperplane is claimed to minimize the worst-case (maximal) probability of misclassification, or the worst-case error rate, of future data.

Further, applying the generalization of Marshall and Olkin's result [14], [17], the optimization of MPM can be transformed to a Second Order Cone Programming (SOCP) problem as follows [13], [15]:

$$\min_{\mathbf{a}} \quad \|{\Sigma_{\mathbf{x}}}^{1/2}\mathbf{a}\|_2 + \|{\Sigma_{\mathbf{y}}}^{1/2}\mathbf{a}\|_2 \quad \text{s.t.} \quad \mathbf{a}^\top(\bar{\mathbf{x}} - \bar{\mathbf{y}}) = 1. \quad (2)$$

The worst complexity of solving the optimization problem of MPM, i.e., solving the SOCP problem, in Eq. (2), is $O(n^3)$, where n is the number of training samples for the kernelized MPM. In the test phase, the complexity of the kernelized MPM also depends on the number of training samples. This high computational complexity is a main problem of applying MPM to real applications.

B. Minimax Clustering Probability Machine

Aiming at reducing the computation complexity of MPM, we propose the Minimax Clustering Probability Machine (MCPM). The idea is as follows. The training samples of class $\mathbf{x}$ and training samples of class $\mathbf{y}$ are clustered into $M_{\mathbf{c_x}}$ training clusters and $M_{\mathbf{c_y}}$ training clusters, respectively. Following the Gaussian distribution assumption, we could denote the training clusters as generative models, i.e., $\mathbf{c}_j = (P_j, \mu_j, \Sigma_j)$, where P_j, μ_j, Σ_j is the prior (weight), the mean, and the covariance matrix, of the j-th cluster. For the positive clusters, j ranges from 1 to $M_{\mathbf{c_x}}$. For the negative clusters, j ranges from 1 to $M_{\mathbf{c_y}}$. Hence, the total number of training clusters is $M = M_{\mathbf{c_x}} + M_{\mathbf{c_y}}$.

In the following, we denote the space of generative models as $\mathbb{R}^{\mathcal{G}} = \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^{d\times d}$. Therefore, the problem becomes to find a linear decision boundary $\mathcal{H}(\mathbf{c}, b) = \{\mathbf{z} \in \mathbb{R}^{\mathcal{G}} \,|\, \mathbf{c}^\top \mathbf{z} = b\}(\mathbf{c} \in \mathbb{R}^{\mathcal{G}} \backslash \{\mathbf{0}\}, b \in \mathbb{R})$. We then transform the above generative models, $\mathbf{c}_j$, $j = 1, \ldots, M$, from $\mathbb{R}^{\mathcal{G}}$ to a feature space, $\mathbb{R}^f$, via a mapping $\phi : \mathbb{R}^{\mathcal{G}} \mapsto \mathbb{R}^f$. Therefore, a linear decision boundary $\mathcal{H}(\mathbf{c}, b) = \{\phi(\mathbf{z}) \in \mathbb{R}^f \,|\, \mathbf{c}^\top \phi(\mathbf{z}) = b\}$ in the feature space $\mathbb{R}^f$ corresponds to a nonlinear decision decision boundary $\mathcal{D}(\mathbf{c}, b) = \{\mathbf{z} \in \mathbb{R}^{\mathcal{G}} \,|\, \mathbf{c}^\top \phi(\mathbf{z}) = b\}$ in the space of $\mathbb{R}^{\mathcal{G}}$ ($\mathbf{c} \in \mathbb{R}^f \backslash \{\mathbf{0}\}$ and $b \in \mathbb{R}$).

Now, let the training clusters be mapped as

$$\begin{aligned} \mathbf{c_x} \mapsto \phi(\mathbf{c_x}) &\sim \{\overline{\phi(\mathbf{c_x})}, \Sigma_{\phi(\mathbf{c_x})}\}, \\ \mathbf{c_y} \mapsto \phi(\mathbf{c_y}) &\sim \{\overline{\phi(\mathbf{c_y})}, \Sigma_{\phi(\mathbf{c_y})}\}. \end{aligned}$$

A nonlinear decision boundary in $\mathbb{R}^{\mathcal{G}}$ can then be obtained by solving the minimax probability decision problem of Eq. (1), in a feature space $\mathbb{R}^f$:

$$\max_{\alpha, \mathbf{c}\neq\mathbf{0}, b} \alpha \text{ s.t.} \quad \inf_{\phi(\mathbf{c_x})\sim\{\overline{\phi(\mathbf{c_x})}, \Sigma_{\phi(\mathbf{c_x})}\}} \mathbf{Pr}\{\mathbf{c}^\top \phi(\mathbf{c_x}) \geq b\} \geq \alpha,$$
$$\inf_{\phi(\mathbf{c_y})\sim\{\overline{\phi(\mathbf{c_y})}, \Sigma_{\phi(\mathbf{c_y})}\}} \mathbf{Pr}\{\mathbf{c}^\top \phi(\mathbf{c_y}) \leq b\} \geq \alpha.$$

Similar to the optimization of Eq. (1), the above optimization can be solved by

$$\begin{aligned} \tfrac{1}{\tau^*} \quad := \quad & \min_{\mathbf{c}} \quad \|{\Sigma_{\phi(\mathbf{c_x})}}^{1/2}\mathbf{c}\|_2 + \|{\Sigma_{\phi(\mathbf{c_y})}}^{1/2}\mathbf{c}\|_2 \\ & \text{s.t.} \quad \mathbf{c}^\top(\overline{\phi(\mathbf{c_x})} - \overline{\phi(\mathbf{c_y})}) = 1. \end{aligned}$$

By adopting the kernel trick similar to that in [11], [7], we can write $\mathbf{c}$ as a linear combination of the training clusters

and then find the coefficients. Without loss of generality, $\mathbf{c}$ can be written as $\mathbf{c} = \sum_{i=1}^{M_{\mathbf{c_x}}} \nu_i \phi(\mathbf{c_x}_i) + \sum_{j=1}^{M_{\mathbf{c_y}}} \omega_j \phi(\mathbf{c_y}_j)$.

Let $\{\mathbf{t}_i\}_{i=1}^{M}$ denote the set of all $M = M_{\mathbf{c_x}} + M_{\mathbf{c_y}}$ training clusters as $\mathbf{t}_j = \mathbf{c_x}j, j = 1, 2, \ldots, M_{\mathbf{c_x}}$, and $\mathbf{t}_j = \mathbf{c_y}_{j-M_{\mathbf{c_x}}}, j = M_{\mathbf{c_x}} + 1, M_{\mathbf{c_x}} + 2, \ldots, M$.

The Gram matrix $\mathbf{G}$ can be defined as

$$\mathbf{G}_{ij} = G(\phi(\mathbf{t}_i), \phi(\mathbf{t}_j)), \quad i, j = 1, 2, \ldots, M. \tag{3}$$

Denote the first $M_{\mathbf{c_x}}$ rows and the last $M_{\mathbf{c_y}}$ rows of $\mathbf{G}$ as $\mathbf{G}_{\mathbf{c_x}}$ and $\mathbf{G}_{\mathbf{c_y}}$, respectively, we get $\mathbf{G} = [\mathbf{G}_{\mathbf{c_x}}; \mathbf{G}_{\mathbf{c_y}}]$.

The block-row-averaged Gram matrix $\mathbf{K}$ is then obtained by setting the row average of the $\mathbf{G}_{\mathbf{c_x}}$-block and the $\mathbf{G}_{\mathbf{c_y}}$-block equal to zero:

$$\mathbf{K} = \begin{pmatrix} \mathbf{G}_{\mathbf{c_x}} - \mathbf{1}_{M_{\mathbf{c_x}}} \mathbf{k}_{\mathbf{c_x}}^\top \\ \mathbf{G}_{\mathbf{c_y}} - \mathbf{1}_{M_{\mathbf{c_y}}} \mathbf{k}_{\mathbf{c_y}}^\top \end{pmatrix} = \begin{pmatrix} \sqrt{M_{\mathbf{c_x}}} \mathbf{K}_{\mathbf{c_x}} \\ \sqrt{M_{\mathbf{c_y}}} \mathbf{K}_{\mathbf{c_y}} \end{pmatrix},$$

where $\mathbf{1}_n$ is a column vector of ones of dimension n. The row average $\mathbf{k}_{\mathbf{c_x}}^\top$ and $\mathbf{k}_{\mathbf{c_y}}^\top$ are M-dimensional row vectors given by

$$\left(\mathbf{k}_{\mathbf{c_x}}^\top\right)_i = \frac{1}{M_{\mathbf{c_x}}} \sum_{j=1}^{M_{\mathbf{c_x}}} K(\mathbf{c_x}_j, \mathbf{t}_i)$$

$$\left(\mathbf{k}_{\mathbf{c_y}}^\top\right)_i = \frac{1}{M_{\mathbf{c_y}}} \sum_{j=1}^{M_{\mathbf{c_y}}} K(\mathbf{c_y}_j, \mathbf{t}_i)$$

Hence, the objective of MCPM becomes

$$\begin{aligned} \frac{1}{\tau^*} := \min_{\mathbf{v}} \quad & \|\mathbf{K}_{\mathbf{c_x}} \mathbf{v}\|_2 + \|\mathbf{K}_{\mathbf{c_y}} \mathbf{v}\|_2 \\ \text{s.t.} \quad & \mathbf{v}^\top (\mathbf{k}_{\mathbf{c_x}} - \mathbf{k}_{\mathbf{c_y}}) = 1, \end{aligned} \tag{4}$$

where $\mathbf{v} = [\nu_1, \nu_2, \ldots, \nu_{M_{\mathbf{c_x}}}, \omega_1, \omega_2, \ldots, \omega_{M_{\mathbf{c_y}}}]^\top$.

The decision function of MCPM is then calculated by

$$\begin{aligned} f_{MCPM}(\mathbf{c_z}) &= \sum_{i=1}^{M_{\mathbf{c_x}}} \mathbf{v}_i^* \mathbf{K}(\mathbf{c_z}, \mathbf{c_x}_i) \\ &+ \sum_{i=1}^{M_{\mathbf{c_y}}} \mathbf{v}_{M_{\mathbf{c_x}}+i}^* \mathbf{K}(\mathbf{c_z}, \mathbf{c_y}_i) - b_{MCPM}^* \end{aligned} \tag{5}$$

and the bias term is obtained by $b_{MCPM}^* = \mathbf{v}^{*\top} \mathbf{k}_{\mathbf{c_x}} - \tau^* \|\mathbf{K}_{\mathbf{c_x}} \mathbf{v}\|_2 = \mathbf{v}^{*\top} \mathbf{k}_{\mathbf{c_y}} + \tau^* \|\mathbf{K}_{\mathbf{c_y}} \mathbf{v}\|_2$.

From the optimization of Eq. (4), we can see that the optimization is similar to the kernelized MPM. However, the number of training samples is reduced largely to the number of training clusters.

III. Generalized probability product kernel

In solving Eq. (4), we still need a suitable distance definition to measure the similarity between two clusters in the training phase or the similarity between a cluster and a sample vector in the test phase. In the following, we will first introduce general kernels in the feature space. We then derive the generalized probability product kernel. After that, we present how to apply the generalized probability product kernel, more specially, the linear probability product kernel and the radial basis function on probability product kernel, to real applications.

A. Kernel in Feature Space

The kernel is defined in Eq. (3). Considering a linear kernel in the feature space, we can define it as

$$G_L(\phi(\mathbf{t}_i), \phi(\mathbf{t}_j)) = \phi(\mathbf{t}_i)^\top \phi(\mathbf{t}_j). \tag{6}$$

Similarly, we can define a RBF kernel in the feature space:

$$G_{RBF}(\phi(\mathbf{t}_i), \phi(\mathbf{t}_j)) = \exp\{-\gamma \|\phi(\mathbf{t}_i) - \phi(\mathbf{t}_j)\|^2\} \tag{7}$$

We can also extend the kernel in the feature space to general forms by other functions, e.g., the polynomial kernel and the hyperbolic tangent kernel [18]. This can attain generalized kernels in the feature space.

B. Probability Product Kernel

Here we still need to define the inner product of two vectors in the feature space. Considering the property of the generative models we have obtained, we turn to the probability product kernel [8]. The probability product kernel defines the similarity between two distribution p_k and p_l by

$$K(p_k, p_l) = \int_{\mathbb{R}^d} p_k^\rho \, p_l^\rho \, d\mathbf{z}, \tag{8}$$

where $K(p_k, p_l)$ is positive definite, and the exponential ρ will derive a set of candidate kernels. When $\rho = 1$, it leads to the expected likelihood kernel [8].

When p_k and p_l are both Gaussian distributions, i.e., $p_k = P_k p(\mathbf{z} \,|\, \mu_k, \Sigma_k)$ and $p_l = P_l p(\mathbf{z} \,|\, \mu_l, \Sigma_l)$, $K(p_k, p_l)$ can be written as a function of two generative models, i.e., $K(\mathbf{c}_k, \mathbf{c}_l)$. Further, $K(\mathbf{c}_k, \mathbf{c}_l)$ can be computed directly by using the corresponding parameters of two generative models to avoid integrating the probability distributions in the entire input space. Hence, when $\rho = 1$, we have

$$\begin{aligned} & K(\mathbf{c}_k, \mathbf{c}_l) = \phi(\mathbf{t}_i)^\top \phi(\mathbf{t}_j) \\ = \; & P_k P_l (2\pi)^{-\frac{d}{2}} |(\Sigma_k{}^{-1} + \Sigma_l{}^{-1})^{-1}|^{\frac{1}{2}} |\Sigma_k|^{-\frac{1}{2}} |\Sigma_l|^{-\frac{1}{2}} \\ & \exp\left\{ -\frac{1}{2} \left(\mu_k^\top \Sigma_k{}^{-1} \mu_k + \mu_l^\top \Sigma_l{}^{-1} \mu_l - \widetilde{\mu}^\top \widetilde{\Sigma}^{-1} \widetilde{\mu} \right) \right\} \end{aligned} \tag{9}$$

where $\widetilde{\Sigma}^{-1} = (\Sigma_k{}^{-1} + \Sigma_l{}^{-1})^{-1}$, and $\widetilde{\mu} = \Sigma_k{}^{-1} \mu_k + \Sigma_l{}^{-1} \mu_l$.

C. Practical Solution

For real classification problems, in order to avoid computing the inverse matrices in Eq. (9), we simplify the kernel calculation by only using the diagonal entries of the covariance matrices, i.e., $\Sigma_l = \text{diag}((\sigma_{l,1}^2, \ldots, (\sigma_{l,d}^2)$. Thus, the kernel becomes

$$\begin{aligned} & K(\mathbf{c}_k, \mathbf{c}_l) = \phi(\mathbf{c}_k)^\top \phi(\mathbf{c}_l) \\ = \; & \frac{P_k P_l}{\prod_{i=1}^d \sqrt{2\pi(\sigma_{k,i}^2 + \sigma_{l,i}^2)}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^d \frac{(\mu_{k,i} - \mu_{l,i})^2}{\sigma_{k,i}^2 + \sigma_{l,i}^2} \right\}. \end{aligned} \tag{10}$$

In the test phase, a test sample $\mathbf{z}$ is considered as the extreme case of a Gaussian distribution, where only one point in the distribution with fixed prior and all elements of the covariance matrix vanishing, i.e., $\mathbf{c_z} = (P_{\mathbf{z}} = 1, \mu_{\mathbf{z}} =$

$\mathbf{z}, \Sigma_{\mathbf{z}} = \mathbf{0}$). Hence, the similarity between a training cluster and a test vector is defined by

$$K(\mathbf{c_z}, \mathbf{c}_l) = \phi(\mathbf{c_z})^\top \phi(\mathbf{c}_l) \quad (11)$$
$$= P_l \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\sigma_{l,i}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{d} \frac{(\mathbf{z} - \mu_{l,i})^2}{\sigma_{l,i}^2} \right\}.$$

Hence, we term the calculation of Eq. (10) and Eq. (11) as linear probability product kernels (LPPK) for calculating similarity between clusters in training phase and that between a cluster and a test vector in test phase. After plugging Eq. (10) and Eq. (11) into the RBF form of Eq. (7), we therefore define the Radial Basis Function on probability product kernel (RBF-PPK). Similarly, by defining other measurements in the feature space, e.g., Polynomial functions, or Hyperbolic tangent, we can extend and obtain a generalized probability product kernel.

Comparing the linear kernel in Eq. (6) and the RBF kernel in Eq. (7), we can see that the RBF kernel has the advantage of containing unit value when two inputs are the same, while the linear kernel has no such property. For real applications, data are usually in high dimensional space. This makes the calculation values of Eq. (10) and Eq. (7) very small and appear in different scales. It would incur the difficulty of tuning parameters to get good results. However, the RBF-PPK has the ability of normalizing the kernel matrix and hence avoids the problems of the linear PPK.

IV. Experiments

We carry out experiments on a two-class toy dataset and two benchmark datasets to demonstrate the effectiveness of MCPM. In the toy dataset, $2,000$ data points, $1,000$ points for each class, were randomly generated from a mixture of Gaussian distribution in order to visualize the learning results of the MCPM in the 2-D space. The real datasets used are the two benchmark binary classification datasets, the Pima indians diabetes dataset and the Twonorm dataset, from the machine learning repository [1], [11], [7]. The Pima indians diabetes dataset consists of 768 instances with 8 attributes. The twonorm dataset, consisting of $7,400$ samples with 20 attributes, was generated from a multivariate normal distribution [1], [11].

A. Toy Data

B. Experimental Setup and Model Selection

In the experiment, each dataset is partitioned into 90% training and 10% test sets. The final results are the average results over 10 random partitions. Comparisons are performed on MCPM, Support Clustering Machine (SCM) [12] (both in LPPK and RBF-PPK), and the MPM. For fair comparisons, we adopt the Threshold Order Dependent (TOD) algorithm [3], the same clustering method as used in [12]. The experiments are performed on a PC with a $2.13GHz$ Intel Core 2 CPU and $1G$ RAM. We use Matlab 7.1 to conduct the comparisons.

Several parameters need to be tuned in training different models. For the kernelized MPM, we use the Gaussian kernel, $e^{-\|\mathbf{x}-\mathbf{y}\|^2/\sigma}$, with parameter σ. For SCM, the parameters are the trade-off parameter C and the width parameter γ when the RBF-PPK is used. For the MCPM with RBF-PPK, only the width parameter γ needs to be tuned. All these parameters are chosen via cross-validation on the training dataset.

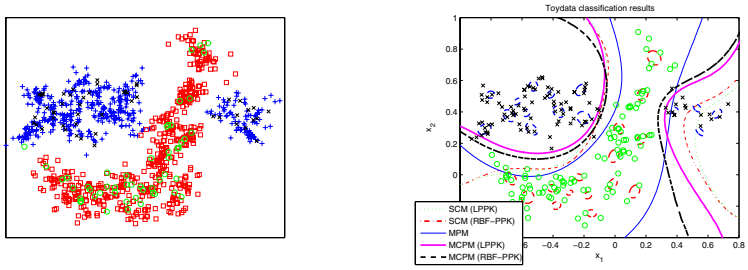
$2,000$ data points, generated from a mixture of Gaussian distribution, are plotted in Fig. 1(a). The TOD algorithm is applied to group the $\mathbf{x}$-class data into 15 positive clusters and to group the $\mathbf{y}$-class samples into 15 negative clusters, respectively. As shown in Fig. 1(b), the training clusters are denoted by ellipses, where the weights are proportional to the sizes of clusters. In the experiment, the obtained weights (priors), means, and covariance matrices of training clusters are used as the input for SCM and MCPM.

Table I reports the average training time, test time and accuracies obtained by the kernelized MPM, SCM and MCPM with LPPK and RBF-PPK. We can see that the time cost of MCPM is largely reduced when compared to that of MPM, while the accuracy is just slightly decreased. Especially, the training time of the kernelized MPM is reduced from 660.7 to 0.1460, which is over $4,500$ times reduction. Meanwhile, there is over 30 times reduction in the test time. The training and test time for SCM and MCPM are nearly the same for LPPK and RBF-PPK, while the SCM and MCPM with RBF-PPK outperforms the SCM and MCPM with LPPK respectively in terms of the accuracy. This shows that the generalized non-linear probability product kernel, i.e., RBF-PPK, is superior to the linear PPK in the toy data. Moreover, different from SCM, both MPM and MCMP can generate an explicit worst-case accuracy bound α. Furthermore, the bound of MCPM with RBF-PPK is tighter than those of MPM and MCPM with LPPK. This again demonstrates the superiority of the MCPM using the generalized PPK over other methods.

C. Benchmark Datasets

We compare the performance of the proposed MCPM with other methods on two benchmark datasets in this section. Table II reports the average training time, test time, worst-case accuracy bounds, and accuracies on the Pima indians diabetes dataset; while Table III reports the average results on the Twonorm dataset. From table II and table III, we have the following observations:

- Although the kernelized MPM has better accuracy than the linear MPM, it costs too much time on the training procedure.
- The MCPM overcomes the shortcoming of the kernelized MPM: it reduces the training time largely, over $10,000$ times reduction, while maintaining a comparable accuracy to that of the kernelized MPM for both datasets.
- The MCPM can also output a worst-case accuracy bound α. The bound is once again tighter than that of the MPM.



(a) Samples

(b) Results of SCM, MPM and MCPM

Fig. 1. Kernelized MPM, SCM and MCPM with LPPK and RBF-PPK in a 2-D space. Training data are indicated with blue $+$s for class $\mathbf{x}$ and red $\square$s for class $\mathbf{y}$. Test samples are indicated with black $\times$s for class $\mathbf{x}$ and green os for class $\mathbf{y}$. The training clusters are represented by ellipses with size proportional to the priors, blue ellipses for class $\mathbf{c_x}$ and red ellipses for class $\mathbf{c_y}$. The decision boundaries constructed by the SCM with LPPK (thin green dotted line), the SCM with RBF-PPK (thick red dash-dot line), the MPM (thin blue solid line), the MCPM with LPPK (thick magenta solid line), and the MCPM with RBF-PPK (thick black dash line) are shown. Notice that SCM and MCPM with RBF-PPK improve the test set performance compared to the SCM and MCPM with LPPK.

TABLE I

AVERAGE RESULTS ON THE SYNTHETIC DATASET.

Methods	**Training** (s)	**Test** (s)	α (%)	**Accuracy** (%)
SCM (LPPK)	0.1418 ± 0.0021	0.0019 ± 0.0001	–	96.6 ± 0.8
SCM (RBF-PPK)	0.1419 ± 0.0015	0.0020 ± 0.0001	–	97.1 ± 0.6
MPM	660.7 ± 10.1	0.0681 ± 0.0001	82.6 ± 0.8	98.1 ± 1.0
MCPM (LPPK)	0.1460 ± 0.0015	0.0020 ± 0.0001	81.7 ± 1.2	97.4 ± 0.9
MCPM (RBF-PPK)	0.1461 ± 0.0015	0.0020 ± 0.0001	83.1 ± 1.1	97.8 ± 0.8

TABLE II

AVERAGE RESULTS ON THE PIMA INDIANS DIABETES DATASET.

Methods	**Training** (s)	**Test** (s)	α (%)	**Accuracy** (%)
SCM (LPPK)	0.0034 ± 0.0001	0.0006 ± 0.0001	–	66.5 ± 1.6
SCM (RBF-PPK)	0.0035 ± 0.0001	0.0006 ± 0.0001	–	74.1 ± 1.6
MPM (Linear)	0.0041 ± 0.0015	0.0010 ± 0.0002	32.3 ± 0.5	73.5 ± 0.9
MPM (Kernel)	240.8 ± 28.5	0.0118 ± 0.0001	33.2 ± 0.8	74.5 ± 0.8
MCPM LPPK	0.0030 ± 0.0001	0.0007 ± 0.0001	35.1 ± 1.5	65.1 ± 1.4
MCPM RBF-PPK	0.0031 ± 0.0001	0.0007 ± 0.0001	40.2 ± 1.4	74.5 ± 1.5

- The proposed generalized probability product kernel, i.e., the RBF-PPK can largely improve the accuracy of the traditional probability product kernel, i.e., the LPPK, in both SCM and MCPM.

The above observations validate the advantages of our proposed method and show that both the training and the test time can be reduced greatly by our MCPM method while the accuracy can be maintained. Moreover, the proposed generalized probability kernel can deliver better accuracies than the traditional probability product kernel.

In order to examine the performance when different cluster numbers are chosen, we show the average test error rates of SCM and MCPM using LPPK and RBF-PPK with respect to the number of training clusters in Fig. 2. From this figure, we have the following observations. First, the best results of SCM and MCPM with LPPK and RBF-PPK are obtained in different number of training clusters. This shows that the number of clusters could indeed influence the overall accuracy. In order to obtain the best performance, we may need to choose this parameter carefully. Second, in all the cases, the learning algorithms using the RBF-PPK consistently outperform those using the LPPK. This clearly demonstrates the advantages of the proposed non-linear generalized PPK.

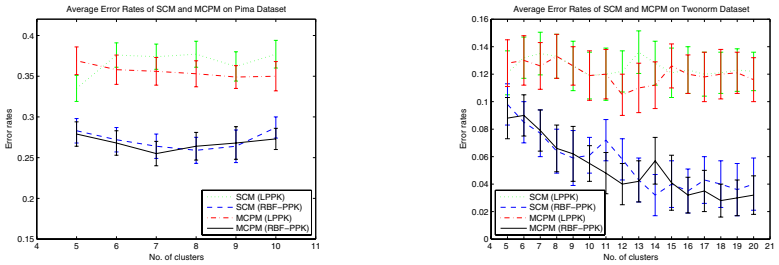
V. CONCLUSION

In this paper, we have proposed an efficient Minimax Clustering Probability Machine model. This model can elegantly incorporate cluster information into the learning process so as to reduce the training and test time complexity greatly. We have proposed a generalized probability product kernel. This kernel has demonstrated desirable properties in measuring the similarity defined either between a pair of clusters or a cluster and a test vector. Experimental results on both synthetic and real data show that the proposed algorithm can reduce the training and test time significantly while preserving the accuracy. Moreover, the proposed generalized probability product kernel has been shown to outperform the traditional linear probability product kernel consistently.

TABLE III

AVERAGE RESULTS ON THE TWONORM DATASET.

Methods	**Training** (s)	**Test** (s)	α (%)	**Accuracy** (%)
SCM (LPPK)	0.1528 ± 0.0001	0.0031 ± 0.0001	–	88.1 ± 1.7
SCM (RBF-PPK)	0.1529 ± 0.0001	0.0031 ± 0.0001	–	96.8 ± 1.5
MPM (Linear)	0.2263 ± 0.0001	0.0052 ± 0.0001	81.3 ± 0.2	96.2 ± 0.5
MPM (Kernel)	25893.1 ± 13.5	0.1112 ± 0.0001	85.4 ± 0.5	97.9 ± 0.5
MCPM (LPPK)	0.1531 ± 0.0001	0.0030 ± 0.0001	86.1 ± 0.6	89.5 ± 1.5
MCPM (RBF-PPK)	0.1532 ± 0.0001	0.0031 ± 0.0001	89.2 ± 0.7	97.2 ± 1.2



(a) Pima Dataset (b) Twonorm Dataset

Fig. 2. Average error rates of SCM and MCPM with respect to the number of clusters on Pima dataset and Twonorm dataset.

Several important issues deserve our attentions in the future. First, the clustering and the classifier learning are currently implemented in two separate steps. It remains interesting whether these two steps can be unified in one step. Second, although both theoretical justification and empirical verification has demonstrated the advantages of the proposed generalized probability product kernel, further explorations on its mathematic properties are still important research topics. Third, we mainly evaluate our algorithm on two-class data in this paper for simplicity. Extensive investigations on large-scale multi-class real data are also necessary. Finally, how to choose the optimal cluster number is also an important research topic in the future.

REFERENCES

[1] L. Breiman. Arcing classifiers. Technical Report 460, Statistics Department, University of California, 1997.

[2] R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[3] M. Friedman and A. Kandel. *Introduction to pattern recognition : statistical, structural, neural, and fuzzy logic approaches*. World scientific, Singapore, 1999.

[4] Hans Peter Graf, Eric Cosatto, Léon Bottou, Igor Dourdanovic, and Vladimir Vapnik. Parallel support vector machines: The cascade svm. In *NIPS 17*, pages 521–528, 2005.

[5] K. Huang, H. Yang, I. King, and M. R. Lyu. Learning classifiers from imbalanced data based on biased minimax probability machine. In *CVPR-2004*, volume 2, pages 558–563, 2004.

[6] K. Huang, H. Yang, I. King, and M. R. Lyu. Maximizing sensitivity in medical diagnosis using biased minimax probability machine. *IEEE Transactions on Biomedical Engineering*, 53:821–831, 2006.

[7] K. Huang, H. Yang, I. King, M. R. Lyu, and L. Chan. The minimum error minimax probability machine. *Journal of Machine Learning Research*, 5:1253–1286, 2004.

[8] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research, Special Topic on Learning Theory*, pages 819–844, 2004.

[9] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[10] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. Minimax probability machine. In *NIPS 15*, 2001.

[11] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

[12] B. Li, M. Chi, J. Fan, and X. Xue. Support cluster machine. In *ICML'07*, pages 505–512, 2007.

[13] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second order cone programming. *Linear Algebra and its Applications*, 284:193–228, 1998.

[14] A. W. Marshall and I. Olkin. Multivariate chebyshev inequalities. *Annals of Mathematical Statistics*, 31(4):1001–1014, 1960.

[15] Y. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming: Theory and applications*. Studies in Applied Mathematics. Philadelphia, 1994.

[16] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

[17] I. Popescu and D. Bertsimas. Optimal inequalities in probability theory: A convex optimization approach. Technical Report TM62, INSEAD, 2001.

[18] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[19] T. Strohmann and G. Grudic. A formulation for minimax probability machine regression. In S. Becker, S. Thrun, and K. Obermayer, editors, *In NIPS 15*. MIT Press, 2003.

[20] H. Yu, J. Yang, and J. Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 306–315, 2003.

[21] J. Yuan, J. Li, and B. Zhang. Learning concepts from large scale imbalanced data sets using support cluster machines. In *ACM Multimedia*, pages 441–450, 2006.