Learning Discriminative Piecewise Linear Models with Boundary Points

Kun Gai and **Changshui Zhang**
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing 100084, China
gaik02@mails.thu.edu.cn, zcs@mail.thu.edu.cn

Abstract

We introduce a new discriminative piecewise linear model for classification. A two-step method is developed to construct the model. In the first step, we sample some boundary points that lie between positive and negative data, as well as corresponding directions from negative data to positive data. The sampling result gives a discriminative nonparametric decision surface, which preserves enough information to correctly classify all training data. To simplify this surface, in the second step we propose a nonparametric approach for linear surface segmentation using Dirichlet process mixtures. The final result is a piecewise linear model, in which the number of linear surface pieces is automatically determined by the Bayesian inference according to data. Experiments on both synthetic and real data verify the effectiveness of the proposed model.

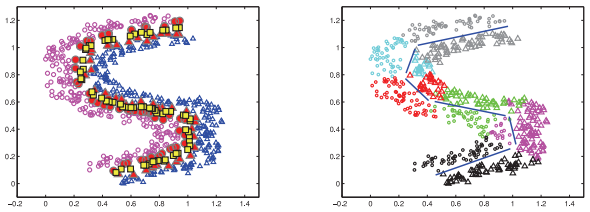
Introduction

In classification, the model complexity of the decision surface plays a crucial role on the prediction performance. If there is a misfit between the model complexity and the amount of data, there will be either under- or over-fitting problems. Some classification methods employ parametric models, such as kernel functions with prespecified parameters in kernel support vector machines (SVM), to control the complexity of decision surface. However, it is difficult to determine suitable parameters. We introduce a discriminative piecewise linear model using Dirichlet process mixtures (DPM), where the unbounded number of linear surface pieces mitigates under-fitting, and the Bayesian inference of posterior over the number of linear surface pieces mitigates over-fitting.

A Dirichlet process (DP), written as $DP(\alpha, G_0)$, with base distribution G_0 and concentration parameter α, is a distribution over distributions, and is first introduced by Ferguson (1973) as a prior over distributions in the Bayesian approach. Blackwell and MacQueen (1973) then using the Blackwell-MacQueen Urn Scheme showed that draws from a DP are discrete almost surely. One of the most common use of DP is in Dirichlet process mixtures (DPM) (Antoniak 1974; Escobar and West 1995; Neal 1992). DPM is a nonparametric mixture model with countably infinite number of clusters, and the actual number of clusters can be automatically determined by the Bayesian posterior inference. More details of DP and DPM can be found in (Teh 2007).

Original DPM aims to model distributions over observed data. In regression and classification, when observed data contain both covariates, x, and responses, y, using DPM to model joint distribution of x and y yields generative nonparametric method for regression (Müller, Erkanli, and West 1996) and classification (Shahbaba and Neal 2009). In the DPM classification model proposed by Shahbaba and Neal (2009), the whole data are divided into different clusters, and in each cluster they use a Gaussian form and a logit linear form to model the marginal distribution of x, $P(x)$, and the conditional distribution of y given x, $P(y|x)$, respectively. Such generative method puts much focus on modeling the marginal distribution $P(x)$, which is indeed not required in prediction. Imagine an area where the relation between y and x is linear, but the distribution of x can not be well fitted by a single Guassian form. This area will tend to be split into several smaller clusters by the Bayesian inference. So the final decision surface tends to be more complex than required, and the confidence of result $P(y|x)$ will also be influenced due to smaller number of data in each cluster. Alternatively, discriminative methods, which only model the conditional distribution $P(y|x)$ or the decision surface, are usually more effective towards the goal of prediction performance. One may ask: could we construct a discriminative model, with the feature of automatically determining model complexity brought by DPM?

We develop a two-step method to construct a discriminative model for binary classification using the Dirichlet process mixtures. In the first step, we sample some certain boundary points which lie between positive and negative data, and also their normal vectors which indicate normal directions of the decision surface. Fig. 1(a) shows a demonstration on s-shape binary data. Triangles (blue) and circles (magenta) represent positive and negative data, respectively, and the bigger makers (red) denote the data which are picked to generate boundary points. Squares (yellow) denote the generated boundary points. These points represent a nonparametric decision surface that preserves enough information to correctly classify all training data. This surface



(a) Sampling boundary points (b) Piecewise linearization

Figure 1: Demonstration of our two-step method.

is very complex, and thus we want to combine boundary points in local areas together to yield a local linear surface with more generalization ability. To achieve this, in the second step we use a new DPM model to piecewise linearize the point set surface, and finally get a piecewise linear model for classification, as shown in Fig. 1(b) (where colors denote different components in the mixture model).

The proposed method brings several interesting points to related problems. First, usually only a small number of training data are selected to generate boundary points, and these selected data preserve zero error on all training data and a bounded generalization error when Nearest Neighbor (NN) principle is used, which results in a new pruning strategy for NN classifier. Second, the piecewise linearization approach using DPM provides a new nonparametric model for subspace segmentation, which is "a fundamental problem in many applications in computer vision (*e.g.*, image/motion/video segmentation), image processing (*e.g.*, image compression), and systems (*e.g.*, hybrid system identification)" (Vidal, Ma, and Sastry 2005). Finally, the introducing of boundary points gives a link between generative and discriminative models: a generative model for boundary points results in a discriminative model for classification.

The rest of this paper is organized as follows. First, a novel strategy of sampling boundary points is presented. Second, we propose a nonparametric Bayesian approach for piecewise linearization using DPM. Then, the performance of the proposed model on real data sets is shown. Finally, we close with a conclusion.

Sampling boundary points

Consider a training set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where x_i is an instance of the d-dimension input space $\mathcal{R}^d$ and $y_i \in \{+, -\}$ is its corresponding class label. For convenience, the training set is divided into a positive set $\mathcal{D}^+ \doteq \{x_i | y_i = +, (x_i, y_i) \in \mathcal{D}\}$ and a negative set $\mathcal{D}^- \doteq \{x_i | y_i = -, (x_i, y_i) \in \mathcal{D}\}$, and $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$. With these training sets, we present a strategy to sample boundary points that lie between positive and negative instances. Then we will prove that the sampling result preserves enough information to correctly classify all training data and to classify new test data with a bounded error rate.

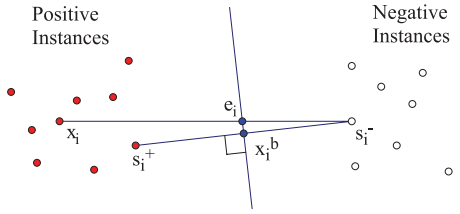


Figure 2: Demonstration of the support point pair (s_i^+ and s_i^-), the edge point e_i and the boundary point x_i^b.

Sampling strategy

For any input instance x_i, without loss of generality, suppose x_i is a positive instance, $x_i \in \mathcal{D}^+$, and consider the following question: which instance in the training set is most dangerous to x_i, *i.e.*, will most influence the final decision surface towards misclassifying x_i?

To make the question tractable, we employ the simple and powerful nearest neighbor principle. Suppose x_i is removed from the training set $\mathcal{D}$, and we use nearest neighbor classifier to classify x_i. The prediction of x_i will be determined by the relation between d_i^+ and d_i^-, which is defined as:

$$d_i^+ \doteq \min_{x_j \in \mathcal{D}^+, j \neq i} \|x_j - x_i\|, \quad (1)$$

$$d_i^- \doteq \min_{x_j \in \mathcal{D}^-} \|x_j - x_i\|. \quad (2)$$

If d_i^- is smaller than d_i^+, x_i will be misclassified to negative instance. So the most dangerous instance to x_i is the negative instance which determines the value of d^-, *i.e.*, is the nearest instance with a different label. Such most dangerous instance is called x_i's negative support point, denoted by s_i^- (see Fig. 2 for example), and it satisfies

$$s_i^- = \arg\min_{x_j \in \mathcal{D}^-} \|x_j - x_i\|. \quad (3)$$

The instance x_i and its negative support point s_i^- are predicted by NN classifier (with the whole training set) to be positive and negative, respectively. So the line segment from x_i to s_i^- must cross the decision surface of nearest neighbor principle. Then we want to obtain the intersection point. There are some properties that are helpful to find it.

Property 1 *For any point z that lies on the line segment between a positive instance x_i and its negative support point s_i^-, the nearest negative training instance of z is s_i^-, i.e., for any $z = \lambda x_i + (1 - \lambda) s_i^-$ $(0 \leq \lambda \leq 1)$, it satisfies*

$$\|z - s_i^-\| = \min_{x_j \in \mathcal{D}^-} \|z - x_j\|. \quad (4)$$

According to Property 1, the predicting label of any point z on the line segment from x_i to s^- is determined by whether we can find a positive training instance $x_k \in \mathcal{D}^+$ that satisfies

$$\|z - x_k\| < \|z - s_i^-\|. \quad (5)$$

The perpendicular bisector of x_k and s_i^- can be used to judge the correctness of (5) (note the bisector is a $(d-1)$-dimension linear surface, and it is a line in the 2-dimension case in Fig. 2). Given any positive training instance x_k, the

perpendicular bisector of x_k and s_i^- will intersect the beeline crossing x_i and s_i^- (if not parallel), on the intersection point o_{ik}, which satisfies

$$\|o_{ik} - s_i^-\| = \frac{\|x_k - s^-\|^2 \|x_i - s_i^-\|}{2\langle x_i - s_i^-, x_k - s_i^- \rangle}, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product. If o_{ik} is between x_i and s_i^-, for any point z on the line segment between o_{ik} and x_i, it satisfies (5), and therefore is located in the positive area of the NN principle. Considering all positive training instances, we have the following property.

Property 2 *For any positive training instance x_i and its negative support point s_i^-, the line segment between x_i and s_i^- intersects the decision surface of NN classifier at only one point. The intersection point is called x_i's edge point, denoted by e_i, and it satisfies*

$$\begin{aligned} \|e_i - s_i^-\| = \quad & \min_{x_k \in \mathcal{D}^+} g(x_k), \\ \text{s.t.} \quad & g(x_k) = \frac{\|x_k - s_i^-\|^2 \|x_i - s_i^-\|}{2\langle x_i - s_i^-, x_k - s_i^- \rangle}, \\ & g(x_k) \geq 0. \end{aligned} \tag{7}$$

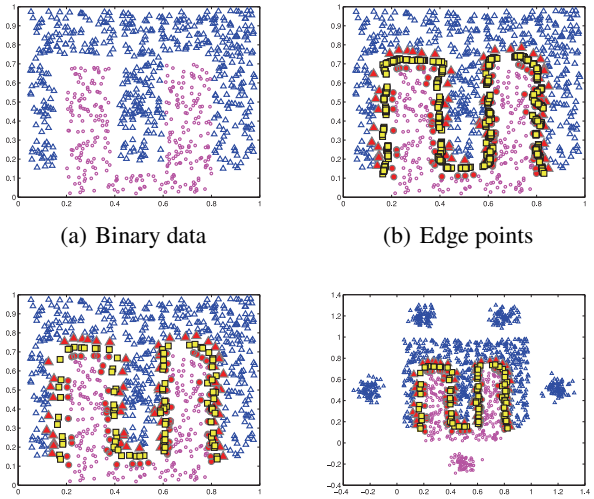
The positive training instance which is the optimal solution of (7) is called x_i's positive support point, denoted by s_i^+. The negative and positive support points form a support point pair. Fig. 2 shows a positive instance x_i, its support point pair s_i^- and s_i^+, and its edge point e_i. Support points s_i^- and s_i^+ are located near the decision surface of NN, and their perpendicular bisector determines a piece of the decision surface, which is similar to support vectors in SVM and thus is the reason why we call them support points.

Now conclude the sampling strategy. For any positive training instance x_i, use (3) to find the negative support point s_i^-, and get the optimal solution and optimal value of (7) as the positive support point s_i^+ and the value of $\|e_i - s_i^-\|$, respectively. Then the edge point is given by

$$e_i = s_i^- + (x_i - s_i^-)\|x_i - s_i^-\|^{-1}\|e - s_i^-\|. \tag{8}$$

For any negative training instance, by exchanging the labels of training instances, we still use the same strategy to get the corresponding support point pair and edge point. The overall computational complexity of sampling edge points and support points from n training instances is $O(n^2)$, which is the same with predicting all training instances by NN classifier.

Fig. 3 shows sampling results from a horseshoe shape data set. The original data with binary labels are shown in Fig. 3(a). 500 positive instances, plotted by (blue) triangles, are uniformly sampled from an M-shape area, and 300 negative instances, plotted by (magenta) circles, are uniformly sampled from a U-shape area. The sampled support points and edge points are shown in Fig. 3(b), where the former are plotted by bigger (and red) markers and the latter are plotted by (yellow) squares. As different instances may correspond the same support point, usually only a small number of data are selected to be support points. Besides, usually most of support points are also support points of themselves. In Fig. 3(b), 127 out of 800 instances are selected to be support points (15.88%), in which 99 instances are support points of themselves.



(a) Binary data (b) Edge points

(c) Boundary points (d) Discriminative property

Figure 3: Horse Shoe data.

Usually the middle point of a support point pair is very near the corresponding edge point, as shown in Fig. 2. Thus we use the middle points as boundary points to give a more compact representation. Fig. 3(c) shows the boundary points by (yellow) squares. In the image, there are 92 different boundary points, far fewer than the edge points which have the number of 800. Besides the boundary point, the direction vector from a negative support point to a corresponding positive support point $(s^+ - s^-)$ shows the norm direction of the decision surface of NN classifier, which is also helpful to represent the decision surface. Consequently, we represent the decision surface by the following terms:

$$x_i^b = (s_i^+ + s_i^-)/2, \quad v_i = (s_i^+ - s_i^-), \tag{9}$$

where the boundary point x_i^b indicates the location that the decision surface should cross, and v_i indicates the norm direction of the decision surface on x_i^b. Points x_i^b and directions v_i result in a complex nonparametric surface. To achieve simpler representations, we will combine nearby x_i^b to yield a piecewise linear decision surface using a nonparametric Bayesian approach. Before that, there are still questions to be discussed. Although edge points lie exactly on the decision surface of NN classifier, the boundary points, which are usually close to edge points, are not guaranteed to be on the NN decision surface. A natural question is whether such points x_i^b together with directions v_i preserves enough information for classification. Another concerned question is: is this sampling strategy a generative or discriminative method?

Zero training error and the bound of test error

Suppose we only use the support points sampled from $\mathcal{D}$, and employ NN classifier to classify all the instances in $\mathcal{D}$.

For any instance x_i, without loss of generality, suppose it is positive, $x_i \in \mathcal{D}^+$. Its negative support point s_i^- is its nearest negative instance in $\mathcal{D}$, also in all support points. Thus, whether x_i can be correctly classified is determined by whether we can find a positive support point closer than s_i^-. Since x_i and its positive support point s_i^+ must be in the same side of the bisector of the support point pair (see Fig. 2), s_i^+ must be nearer x_i than s_i^-. Consequently, x_i will be correctly predicted to be positive. Now we can get:

Theorem 1 *Suppose the training set $\mathcal{D}$ satisfies: $\forall$ i and j, if $x_i = x_j$, then $y_i = y_j$. We use the support points sampled from $\mathcal{D}$ and employ NN principle to predict all the instances in $\mathcal{D}$, and then we will get zero error rate.*

The above property shows that our sampling strategy yields a new pruning strategy for NN classifier that preserves zero error on the original unpruned set. The points x_i^b and directions v_i, which are one-to-one mapped from support point pairs, also preserve enough information for correctly classifying all training instances. This implies that the point set surface is complex enough to mitigates under-fitting. Then, a Bayesian approach will be used to simplify the decision surface to further mitigate over-fitting.

With our pruning strategy, the test error is also bounded:

Theorem 2 *Given a training set $\mathcal{D}$ of size N. We use the support points sampled from $\mathcal{D}$ and employ NN principle to predict new instances. This machine's test error satisfies:*

$$E\{p_{error}^{N-1}\} \leq E\{2N_{support}(\mathcal{D}) - N_{self}(\mathcal{D})\}/N, \quad (10)$$

where p_{error}^{N-1} is the probability of test error for the machine trained on a sample of size $N-1$, $N_{support}(\mathcal{D})$ is the number of support points sampled from $\mathcal{D}$, and $N_{self}(\mathcal{D})$ is the number of instances which are support points of themselves.

The proof is given in the appendix. This theorem shows that with proper use of support points, we can guarantee a bounded generalization error.

Discriminative property

Generally, the strategy of sampling x_i^b and v_i does not explore the generative mechanism of instances in $\mathcal{D}$, and thus it is a discriminative method rather than a generative method. Fig. 3(d) gives an example. We add five gaussian components to the horseshoe shape data: the bottom one is added to negative instances, and the rest are added to positive instances. Using such instances with a different distribution, the sampling results of support points are the same with ones from the original data without added components: there are still 127 different support points, and 92 different boundary points (However, the weights of some support points and boundary points increase, as more instances will result in them.). As shown in the image, the sampling strategy does not put much focus on modeling the overall instance distribution, and directly gives representations of the decision surface, thus the classification as a discriminative method.

Piecewise linearization

Piecewise linear model

Now we have a decision surface represented by x_i^b and v_i, where x_i^b indicates a point that the surface should cross, and v_i indicates the norm direction of the surface. Then we model the decision surface by piecewise linear model using DPM. The whole model is made up by an undetermined number of local components (cluster). In each component, the corresponding x_i^b are supposed to be on a linear surface, with the norm direction represented by w, and thus x_i^b should have a very small variance at the direction w. We use a Gaussian form which has a large precision at the direction w to model x_i^b. Besides, in each component, the directions v_i are supposed to be along w with variant lengths, and thus they have a larger variance at the direction w and smaller variances at other directions. We use a zero-mean Gaussian form which has a small precision at the direction w and large precisions at other directions to model v_i. Considering different components, the overall model using DPM is a piecewise linear model. Our model has the following form:

$$\begin{aligned} x_i^b, v_i|\theta_i &\sim F(\theta_i), & (11)\\ \theta_i|G &\sim G, & (12)\\ G &\sim DP(G_0, \alpha). & (13) \end{aligned}$$

The component parameters $\theta = (\mu, w, \lambda, r, t)$. In our model, the component distribution $F(\theta)$ has the form of:

$$\begin{aligned} p(x_i^b, v_i|\theta) &= p(x_i^b|\theta)p(v_i|\theta), & (14)\\ x_i^b|\theta &\sim N(\mu, (\lambda I + ww^T)^{-1}), & (15)\\ v_i|\theta &\sim N(0, ((r+t)w^T w I - tww^T)^{-1}). & (16) \end{aligned}$$

Here, parameters μ and w are d-dimension vectors that represent the center and the norm direction of the linear surface in each component, respectively, and parameters λ, r, t are scalars to control precisions at the direction w and other directions. In each component, the precision matrix of x_i^b is defined by $(\lambda I + ww^T)$, which has a larger precision $(w^T w + \lambda)$ at the direction w and smaller precision (λ) at directions orthogonal to w. Besides, in each component the precision matrix of v_i is defined by $((r+t)w^T w I - tww^T)$, which has a smaller precision $(rw^T w)$ at the direction w and lager precision $((r+t)w^T w)$ at directions orthogonal to w. Using such Gaussian forms, we linearize points x_i^b and directions v_i in each component to be a linear surface piece with the center μ and the norm direction w. The linear surface only exists locally in the corresponding component, and the overall linearization model is a piecewise linear surface.

The base distribution G_0 of DP is defined as:

$$\begin{aligned} p_0(\theta) &= p(\lambda)p(w)p(\mu|\lambda, w)p(r)p(t), & (17)\\ \lambda &\sim \text{Gamma}(\alpha_0^\lambda, \beta_0^\lambda), & (18)\\ w &\sim N(0, (P_0)^{-1}), & (19)\\ \mu|\lambda, w &\sim N(\mu_0, (\eta_0\lambda I + \eta_0 ww^T)^{-1}), & (20)\\ r &\sim \text{Gamma}(\alpha_0^r, \beta_0^r), & (21)\\ t &\sim \text{Gamma}(\alpha_0^t, \beta_0^t). & (22) \end{aligned}$$

Although G_0 is not a conjugate prior of $F(\theta)$, the Bayesian inference of DPM with non-conjugate priors can still be handled by Markov chain approaches (Neal 2000). In our experiments, we use the Gibbs sampling method with temporary auxiliary parameters (Neal's algorithm 8).

Alternatively, with component distribution $F(\theta)$, if we specify a component number and directly use G_0 as the prior rather than a DP, then we get a parametric Bayesian mixture model for piecewise linearization or subspace segmentation. The MAP estimates of this parametric model can be obtained through Expectation-Maximization (EM) iterations, where the the derivation of updating equations is straightforward and not presented here (due to lack of space). However, in such parametric model it is difficult to determine the component number, which explicitly controls the model complexity. The DP prior endows us a Bayesian way to learn the component number from data. With such prior, the DPM model has been shown to be very effective to find a good balance between fitting accuracy and the model complexity according to data in extensive literature, *e.g.*, (Müller, Erkanli, and West 1996; Teh et al. 2006; Shahbaba and Neal 2009). Thus here we use DPM to efficiently infer a reasonable component number according to observed data.

When prediction, we want to use a fixed piecewise linear model rather than its posterior distribution for simplicity and efficiency. So we make a point estimate from the posterior sampling results. Here we employ maximum a posteriori (MAP) estimates for the the number of components and component parameters. After coverage, we run the Markov chain for an enough long length, and count the occurring frequencies of different component numbers. The final MAP estimate of the component number is given by the most frequently occurring one. After determining the component number, which we suppose to be K, we judge the posterior of component parameters by

$$q(\theta) = \sum\nolimits_{i,1\le j\le K} z_{ij} p(x_i^b, v_i|\theta_j) p_0(\theta_j), \qquad (23)$$

where $z_{ij} = 1$ if the component indicator c_i of (x_i^b, v_i) is equal to j, and $z_{ij} = 0$ otherwise. We use the Markov chain state which results in maximum $q(\theta)$ in the condition of K components as estimates of component parameters, as well as component indicators of data. In fact, after the component number is determined, the situation is similar to the parametric model with a determined number of components. So component parameters can also be determined by the parametric MAP approach with EM iterations.

Predicting rule

Using MAP estimation, we get the component number K, and component indicator c_i $(1 \le c_i \le K)$ for each data (x_i^b, v_i) $(1 \le i \le n)$, as well as the component parameters $(\mu_j, w_j, \lambda_j, r_j, t_j)$ $(1 \le j \le K)$ for each component. These estimates result in a piecewise linear surface, and we want to predict labels of new instances according to this surface. For any instance x_t, we first determine its component indicator l by its nearest boundary point's cluster label, *i.e.*, $l = c_{i^*}$, where

$$i^* = \arg\min_i \|x_t - x_i^b\|. \qquad (24)$$

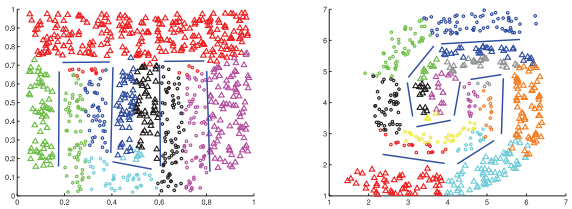


Figure 4: Piecewise linear model on synthetic data.

Then, we use the linear machine of the l-th component to predict x_t's label. We can use $f_i(x) = w_i^T x - w_i^T \mu_i$ as the linear machine for the i-th component, and the sign of of this linear machine can be easily determined by training instances in the i-th component. Besides, according to training instances in each component, the corresponding linear machine can also be given by other linear classification approaches, like linear SVM. Due to the process in (24), the computational complexity of once prediction is linear to the number of different boundary points x_i^b.

Results on synthetic data

In this subsection, we illustrate our model on synthetic data. The first data set is the horseshoe shape data set which has been shown in Fig. 3. We run the Markov Chain for 10000 iterations, and discard the first 1000 iterations. The iterations from 1000 to 5000 is used to determine the component number, and the Markov chain still run to iteration 10000 to refine the MAP estimates on component parameters and component indicators of data. The overall run time is about 6 minutes (with matlab code and 3.0 Ghz CPU). Left image in Fig. 4 shows the final result on the horseshoe shape data. Each color represents a different component, and in all there are six different components and six different linear surfaces (here the linear surface is given by $f_i(x) = w_i^T x - w_i^T \mu_i$). Note the linear surface in the top (red) component is split into two line segments by other components. So we get a piecewise linear model that contains seven line segments. As illustrated, the M-shape positive instances and U-shape negative instances are almost perfectly separated by our piecewise linear model. The right image of Fig. 4 shows the piecewise linear model on another data set. Data are partitioned into nine components, and positive data and negative data are perfectly separated by nine line segments. Besides, our model on S-shape data have been illustrated in Fig. 1.

In both Fig. 1 and 4, our approach achieves high accuracies with as few linear surface pieces as possible. In all these demo sets, if we use fewer numbers of linear surfaces than shown, the binary data can not be well separated. Our approach gives a low and reasonable model complexity. (For the horseshoe shape data, the authors expect 7 clusters with 7 line segments. Surprisingly, our model separates data well using only 6 clusters.) Along with the decision surface, more and shorter / less and longer line segments are given to the place where the decision surface needs to turn rapidly / slowly, respectively. For example in Fig. 1(b), there is a long line segment in the topmost area because one line is

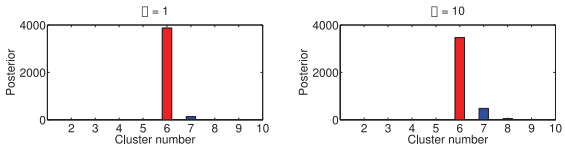


Figure 5: The cluster number posterior.

enough for this large area. Then in the underside area, the overall decision surface need to turn the direction rapidly with more turning points, and thus more and shorter line pieces are distributed here. In the whole space, the partitioning of clusters, line pieces' lengths, positions and directions are all very suitable for good separation. In all, our model achieves a good balance between experiential accuracy and model complexity.

Differences from parametric model

Some other works on parametric models can also result in piecewise linear decision surfaces, *e.g.*, mixtures of SVMs in (Collobert, Bengio, and Bengio 2002) and Profile SVM in (Cheng, Tan, and Jin 2007). Both mixtures of SVMs and Profile SVM partition training data into a determined number of clusters (using K-means or its derivative) and then builds linear SVM for each cluster. These methods does not exploit discriminative information in the clustering step, and thus the partitioning is usually not fit for building linear classification model. In our model the partitioning is very suitable for local linear surfaces, as has been shown. An example on horseshoe shape data can be found in (Cheng, Tan, and Jin 2007), where Profile SVM uses 11 clusters and still can not well separate data. Our model on horseshoe shape data uses only 6 clusters and achieves good separation. Besides, the most significant difference between these parametric models and our nonparametric model is that, they need a prespecified cluster number, whereas in our model the cluster number can be automatically determined by the posterior inference from data.

Generally, the concentration parameter α of DP influences the cluster number prior. The average prior cluster number is directly proportional to α (Teh 2007). One may ask: what is the difference between specifying α in our model and specifying the cluster number in parametric models?

Although α influences the cluster number prior, the cluster number posterior is expected to be mainly determined by data in our Bayesian model. Fig. 5 shows the posterior occurring frequencies of cluster numbers on the horseshoe shape data when α is set to be different values. When $\alpha = 1$, the average posterior cluster number is 6.03, and the MAP estimate is 6. Then we increase α to 10. The average posterior cluster number increases to 6.15, and the MAP estimate is still 6. As shown, the MAP result of our model is invariant to α to some extend, and it is mainly determined by data.

Table 1: Experiments on real data sets.

	SVM		DPLM	
Data Set	Acc.	SV	Acc.	BP
IRIS	92.96	96.67	98.89	20.67
AUSTRALIAN	79.16	91.88	79.86	62.61
DIABETES	68.48	85.94	71.35	71.74
HEART	71.32	96.67	74.65	69.63
LIVER	60.61	98.84	61.00	72.46

Experiments on real data

We test our model on five real data sets from UCI repository (Asuncion and Newman 2007) and Statlog data sets (Brazdil and Gama). For comparison, we also use SVM with RBF kernels, which is one of the most widely used nonlinear classifiers. In particular, the attributes of the data sets are all normalized based on their maximum values. The SVM tool we used is LIBSVM (Chang and Lin 2001). The parameter γ in RBF kernels and C in SVM are determined by grid search based on 10-fold cross validation on the training data.

In our model, the base distribution G_0 of DP is set as:

$$\lambda \sim \text{Gamma}(1 + 0.05d, (1/d + 0.05)\text{tr}(\Sigma_{x^m})), \quad (25)$$

$$w \sim N(0, \text{tr}(\Sigma_{x^m})I/d), \quad (26)$$

$$\mu|\lambda, w \sim N(E(x^m), (0.1\lambda I + 0.1ww^T)^{-1}), \quad (27)$$

$$r \sim \text{Gamma}(1 + 0.5d, 10 + 5d), \quad (28)$$

$$t \sim \text{Gamma}(1 + 0.5d, 0.5 + 0.25d), \quad (29)$$

and the concentration parameter α of DP is set to be 0.1. The basic principle of this setting is to decrease the influence of the prior on results. The accuracies are obtained based on 10-fold cross validation. To make problems more challenging, we use only 1/10 data as training sets, and the other 9/10 data as test sets. Table 1 summarizes accuracies of SVM and our discriminative piecewise linear model (referred to as DPLM), as well as proportions of support vectors (SV) in SVM and proportions between different boundary points (BP) and training data in DPLM. Note that it is not a completely fair comparison, as parameters in SVM can be tuned but parameters in prior of DPLM are fixed. However, DPLM outperforms SVM in all these data sets. One probable reason is that in SVM it is difficult to choose suitable kernel parameters to avoid both over- and under-fitting, whether cross validation is used. In DPLM, reasonable model complexity is automatically determined by the Bayesian inference according to data, and thus our model provides better results. Moreover, the boundary points used in DPLM are always far less than support vectors in SVM, which makes prediction of DPLM much faster.

Conclusion

We develop a two-step method to construct a discriminative piecewise linear model using boundary points, where the model complexity is automatically determined by data. We use the unbounded number of linear surface pieces to mitigate under-fitting, and employ the Bayesian inference of

posterior over the linear surface piece number to mitigate over-fitting. Both synthetic and real data verify the effectiveness of our model.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (NSFC, Grant No. 60835002 and No. 60721003).

Appendix

Proof of Theorem 2. The proof follows the leave-one-out procedure as in (Vapnik and Chapelle 2000). Remove one instance (x_i, y_i) from the training set $\mathcal{D}$, use the remaining instances to construct a classification rule and then test the removed instance. In such way we test all N instances and finally obtain an leave-one-out error rate. Luntz and Brailovsky (Luntz and Brailovsky 1969) proved that the leave-one-out error is an almost unbiased estimate of the probability of test error.

Now consider the the leave-one-out error of the classification rule in Theorem 2. Without loss of generality, we suppose the removed instance, denoted by x_i, is positive, i.e., $x_i \in \mathcal{D}^+$. If x_i is not a support point, and there exist another positive instance $x_j \in \mathcal{D}^+$ $(j \neq i)$ that x_j and x_i have the same positive support point s_i^+, then x_i will be correctly classified in the leave-one-out procedure. It is because x_j and its support point pair remain in the training set, and thus its positive support point s_i^+ will be still selected as a positive support point after x_i is removed. As s_i^+ is nearer x_i than any negative instance in $\mathcal{D}$, x_i will be correctly predicted to be positive according to NN principle.

Let us use N^+, $N^+_{support}$ and N^+_{self} to denote the number of positive instances in $\mathcal{D}$, the number of positive support points (selected from the whole training set $\mathcal{D}$) and the number of positive self-support-points (which are the positive support points of themselves), respectively. Then suppose the N^+_{self} positive self-support points and other $N^+_{support} - N^+_{self}$ positive support points correspond to N_1^+ and N_2^+ positive non-support-points, respectively. For the first group of N_1^+ positive non-support-points, because their positive support points are the different positive instances that result in the same positive support points with them, all these N_1^+ instances will be correctly predicted in the leave-one-out procedure. For the second group of N_2^+ positive non-support-points, because they correspond to $N^+_{support} - N^+_{self}$ positive support points, there exist at most $N^+_{support} - N^+_{self}$ positive non-support-points which have one-to-one correspondences with positive support points. Other $N_2^+ - N^+_{support} + N^+_{self}$ positive non-support-points must have many-to-one correspondences with positive support points, and will be correctly predicted in the leave-one-out procedure. In all, for all positive instances, there exist at least $N_1^+ + N_2^+ - N^+_{support} + N^+_{self} = N^+ + N^+_{self} - 2N^+_{support}$ (due to $N_1^+ + N_2^+ + N^+_{support} = N^+$) instances that will be correctly predicted. Taking both positive and negative instances into account, there are at least $N + N_{self} - 2N_{support}$ instances that will be correctly classified in the leave-one-out procedure. Then, applying Luntz and Brailovsky's theorem (Luntz and Brailovsky 1969) gives the result of Theorem 2. Q.E.D.

References

Antoniak, C. 1974. Mixture of dirichlet process with applications to bayesian nonparamteric problems. *Annals of Statistics* 2(6):1152–1174.

Asuncion, A., and Newman, D. 2007. UCI machine learning repository.

Blackwell, D., and MacQueen, J. B. 1973. Ferguson distributions via pólya urn schemes. *Annals of Statistics* 1:353–355.

Brazdil, P., and Gama, J. Statlog datasets. http://www.liacc.up.pt/ML/old/statlog/datasets.html.

Chang, C.-C., and Lin, C.-J. 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Cheng, H.; Tan, P. N.; and Jin, R. 2007. Localized support vector machine and its efficient algorithm. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM 2007)*, 461–466. Philadelphia, Penn.: Society for Industrial and Applied Mathematics Publications.

Collobert, R.; Bengio, S.; and Bengio, Y. 2002. A parallel mixture of SVMs for very large scale problems. *Neural computation* 14(5):1105–1114.

Escobar, M. D., and West, M. 1995. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association* 90:577–588.

Ferguson, T. S. 1973. A bayesian analysis of some nonparametric problems. *Annals of Statistics* 1(2):209–230.

Luntz, A., and Brailovsky, V. 1969. On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica* 3(6).

Müller, P.; Erkanli, A.; and West, M. 1996. Bayesian curve fitting using multivariate mormal mixtures. *Biometrika* 83(1):67–79.

Neal, R. M. 1992. Bayesian mixture modeling. In *the Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis*, volume 11, 197–211.

Neal, R. M. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics* 9:249–265.

Shahbaba, B., and Neal, R. 2009. Nonlinear models using dirichlet process mixtures. *Journal of Machine Learning Research* 10:1829–1850.

Teh, Y.; Jordan, M.; Beal, M.; and Blei, D. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association* 101(476):1566–1581.

Teh, Y. W. 2007. Dirichlet processes. Encyclopedia of Machine Learning. Forthcoming.

Vidal, R.; Ma, Y.; and Sastry, S. 2005. Generalized principal component analysis (gpca). *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(12):1945–1959.