Discriminative Learning of Mixture of Bayesian Network Classifiers for Sequence Classification

Minyoung Kim
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
mikim@paul.rutgers.edu

Vladimir Pavlovic
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
vladimir@cs.rutgers.edu

Abstract

A mixture of Bayesian Network Classifiers(BNC) has a potential to yield superior classification and generative performance to a single BNC model. We introduce novel discriminative learning methods for mixtures of BNCs. Unlike a single BNC model where the discriminative learning resorts to a gradient search, we can exploit the properties of a mixture to alleviate the complex learning task. The proposed method adds mixture components recursively via functional gradient boosting while maximizing the conditional likelihood. This method is highly efficient as it reduces to generative learning of a base BNC model on weighed data. The proposed approach is particularly suited to sequence classification problems where the kernels in the base model are usually too complex for effective gradient search. We demonstrate the improved classification performance of the proposed methods in an extensive set of evaluations on time-series sequence data, including human motion classification problems.

1. Introduction

Recently, it has been shown that applying a generative model(*e.g.* a Bayesian Network) to a classification task yields performance comparable to sophisticated discriminative classifiers such as SVMs and C4.5 [4]. A model of this class, the Bayesian Network Classifier(BNC), can be used in a wide range of applications including speech recognition and motion time-series classification [1]. Instead of the traditional Maximum Likelihood(ML) learning that fits a BNC model to data, maximizing a conditional likelihood(CML) is known to achieve better classification performance. For instance, in speech recognition, the extended Baum-Welch algorithm for HMM kernels is used to approximate CML [23]. Unfortunately, the CML optimization problem is, in general, complex with non-unique solutions. Typical CML solutions resort to gradient-based numerical optimization methods. Despite improved classification performance, the gradient search makes standard approaches computationally demanding.

Instead of working on a single BNC model, one may benefit from an ensemble-based approach. For example, in [8] AdaBoost of [2] is successfully applied to parameter boosting of a set of BNCs to minimize the exponential loss. However, the resulting model is *not* a generative model which may limit its domain of applications to classification tasks only. Rather, in this paper, we focus on a *mixture* of BNCs. A mixture model has a potential to yield superior classification performance to a single BNC model, as well as a rich density estimator. For instance, [14] formulated a recursive mixture-based approach to motion clustering. The recursive approach has benefits such as the optimal order estimation and insensitive to the initial parameters. However, its aim is not classification and it lacks a proper discriminative objective. This paper formulates a theoretically sound approach to discriminate mixture modeling.

Unlike the single BNC model where the discriminative learning resorts to a gradient search, we can exploit the properties of a mixture to alleviate the complicated learning task. The proposed algorithm learns a mixture recursively. At each iteration, it finds a new component f that, when added to the current mixture F, maximally decreases the conditional loss. A crucial benefit of this method is its efficiency; finding a new f requires ML learning of a base BNC model on weighed data. Therefore, the proposed method is particularly applicable to application domains where the mixture kernels in the base model are so complex that it is prohibitive to compute gradients with respect to model's parameters.

We compare this method to two new heuristic approaches that learn a mixture model discriminatively via AdaBoost. These methods keep the components(or hypotheses) obtained from AdaBoost parameter boosting, and

Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)
0-7695-2597-0/06 $20.00 © 2006 IEEE

learn only the mixing proportions of the mixture model. In an extensive set of experiments that include non-generative discriminative approaches such as kNN, we show that the newly proposed approaches can yield performance comparable or better than that of many standard methods.

2. Discriminative mixture learning

2.1. Preliminaries

Let $f(c, a)^1$ denote a BNC with a class label c and attributes a. In sequence classification where a is a sequence of measurement, $f(c, a)$ can be modeled as a class prior $f(c)$ and the conditional $f(a|c)$. The latter is possibly a collection of HMM kernels, one for each class. Note that this BNC may include many latent variables, such as the state labels in an HMM. Given a BNC $f(c, a)$ and measurements a, the task of predicting the corresponding label c^* is done by MAP decision, $c^* = arg\max_c f(c|a)$. Given some training data D, the traditional ML learning optimizes $max_f \sum_{(c,a)\in D} log\ f(c, a)$, whereas the discriminative CML learning optimizes $max_f \sum_{(c,a)\in D} log\, f(c|a)$.

A mixture $F(c, a)$ is composed of M components, each of which is a BNC $f(c, a)$. That is, $F(c, a) = \sum_{m=1}^{M} \alpha_m f_m(c, a)$, where $\{\alpha_m\}$ are the mixing proportions (*i.e.* $\alpha_m \geq 0$ and $\sum_m \alpha_m = 1$). The class label of sequence a is predicted by the prediction rule: $c^* = arg\max_c F(c|a) = arg\max_c F(c, a) = arg\max_c \sum_{m=1}^{M} \alpha_m f_m(c, a)$.

2.2. Learning via functional gradient boosting

Here we assume binary classification, that is, $c \in \{0, 1\}$, where the extension to multi-class case can be done without any effort. Let $\mathbb{A}$ denote the space of a(e.g., a space of sequences). Given an empirical data set $D_{emp} = \{(c_i, a_i)\}_{i=1}^n$, learning a mixture model discriminatively corresponds to minimizing the negative *conditional* log-likelihood cost:

$$J_{Dis}(F) = \sum_{i=1}^{n} -logF(c_i|a_i) = \sum_{i=1}^{n} -log\frac{F(c_i, a_i)}{F(a_i)}. \quad (1)$$

We want to search for a new BNC component f such that when we replace F with $(1-\epsilon)F + \epsilon f$ for some small positive ϵ, the objective $J_{Dis}(\cdot)$ is maximally decreased. As described in [14], f should make the projection of the negative functional gradient of J_{Dis} onto $f - F$ maximized.

In order to derive a negative functional gradient of $J_{Dis}(F)$, we partition the data space $\mathbb{D} = \{0, 1\} \times \mathbb{A}$ into $\mathbb{D}_1 = \{(c = 1, a)|a \in \mathbb{A}\}$ and $\mathbb{D}_0 = \{(c = 0, a)|a \in \mathbb{A}\}$, where the gradient is separately evaluated for each of these two sub-spaces. For $x = (c, a) \in \mathbb{D}_1$,

$$\begin{aligned} - \frac{\partial J_{Dis}(F)}{\partial F(x)} &= \frac{\partial}{\partial F(c=1,a)} log\frac{F(c=1,a)}{F(a)} \\ &= \frac{\partial}{\partial F(c=1,a)} log\frac{F(c=1,a)}{F(c=1,a)+F(c=0,a)} \\ &= \frac{F(c=1,a)+F(c=0,a)}{F(c=1,a)} \cdot \\ &\qquad \frac{(F(c=1,a)+F(c=0,a)) - F(c=1,a)}{(F(c=1,a)+F(c=0,a))^2} \\ &= \frac{F(c=0|a)}{F(c=1,a)} = \frac{1-F(c=1|a)}{F(c=1,a)} \end{aligned} \quad (2)$$

In the same manner, we can derive the gradient for $x = (c, a) \in \mathbb{D}_0$. As a result, for any $(c, a) \in \mathbb{D}$,

$$-\frac{\partial J_{Dis}(F)}{\partial F(c,a)} = \frac{1-F(c|a)}{F(c,a)}. \quad (3)$$

(3) also holds for the multi-class cases.

Returning to the function optimization problem, the optimal f^* that maximizes $< f - F, \frac{-\partial J_{Dis}(F)}{\partial F(c,a)} >$ is:

$$f^* = arg\max_f \Big[\sum_{i=1}^{n} f(c_i, a_i) \cdot \frac{1-F(c_i|a_i)}{F(c_i,a_i)}\Big]. \quad (4)$$

(4) indicates that the new f is learned with weighed data where the weight of (c_i, a_i) is $(1 - F(c_i|a_i))/F(c_i, a_i)$. Hence the data points having less likelihood($F(c, a)$) and/or less conditional likelihoods($F(c|a)$) by the current mixture are focused on by f in the next stage. This is intuitively a correct argument.

In summary, given an initial mixture $F = f_1{}^2$, the discriminative mixture learning can be done recursively by the following three steps:

Step 1: Search for f^*: (4) can be solved via EM. That is, by taking a log, (4) is a log-sum maximization which is then optimized by the conventional lower-bound maximization technique via Jensen's inequality.

$$\text{(E-step)} \qquad q_i = \frac{w_i f(c_i, a_i)}{\sum_{i=1}^{n} w_i f(c_i, a_i)}, \quad (5)$$

$$\text{where } w_i = \frac{1-F(c_i|a_i)}{F(c_i,a_i)}, \ (i = 1, \ldots, n)$$

$$\text{(M-step)} \qquad \max_f \sum_{i=1}^{n} q_i\ log\, f(c_i, a_i). \quad (6)$$

Step 2: Search for α^*: Given F and f^*, find $\alpha^* \in [0, 1]$

[1] For notational convenience, we use $f(c, a)$ to represent either a BNC or a likelihood at a data point (c, a) interchangeably.

[2] The choice of the initial component f_1 is not significant, however, one can use MLE typically.

such that

$$\alpha^* = arg \min_{\alpha} \sum_{i=1}^{n} -log \left((1-\alpha)F + \alpha f^* \right) (c_i | a_i) \quad (7)$$
$$= arg \max_{\alpha} \sum_{i=1}^{n} log \Big(\frac{(1-\alpha)F(c_i, a_i) + \alpha f^*(c_i, a_i)}{(1-\alpha)F(a_i) + \alpha f^*(a_i)} \Big).$$

Any line search method can be used to solve (7).

Step 3 (optional)[3]**: Refinement** : Refine all the parameters in F, f^*, and α^* such that $J_{Dis}((1-\alpha^*)F + \alpha^* f^*)$ is minimized.

Note that the main complexity of the algorithm is in the M-step of Step 1. This is simply a generative learning of the base model on weighed data, requiring a trivial modification of the original ML learning on un-weighed data. For instance, refer to [1] for the HMM ML learning on weighed data.

2.3. Heuristic learning via AdaBoost

We also suggest two other methods to learn a mixture discriminatively. When a set of BNCs(or weak hypotheses) is obtained from the AdaBoost parameter boosting [8], we can heuristically construct a mixture by keeping these BNCs as its components, and learning only the mixing proportions (*i.e.* α_m). The α-learning can be done either generatively or discriminatively. Formally speaking, given the M components $\{f_m\}$ from the AdaBoost, the generative α-learning finds $\alpha^*_{Gen} = arg \max_\alpha \sum_{i=1}^{n} log\big(\sum_{m=1}^{M} \alpha_m f_m(c_i, a_i)\big)$, which can be solved by the following EM algorithm:

$$\text{(E-step)} \qquad q_{m,i} = \frac{\alpha_m f_m(c_i, a_i)}{\sum_{m=1}^{M} \alpha_m f_m(c_i, a_i)}, \quad (8)$$
$$(m = 1, \ldots, M, \;\; i = 1, \ldots, n)$$
$$\text{(M-step)} \qquad \alpha_m^{new} = \frac{\sum_{i=1}^{n} q_{m,i}}{\sum_{m=1}^{M} \sum_{i=1}^{n} q_{m,i}} \quad (9)$$
$$= \frac{\sum_{i=1}^{n} q_{m,i}}{n}, \; (m = 1, \ldots, M)$$

On the other hand, the discriminative α-learning optimizes:

$$\alpha^*_{Dis} = arg \max_{\alpha} \sum_{i=1}^{n} log \Big(\frac{\sum_{m=1}^{M} \alpha_m f_m(c_i, a_i)}{\sum_{m=1}^{M} \alpha_m f_m(a_i)} \Big),$$
$$subject\ to \sum_{m=1}^{M} \alpha_m = 1, \;\; \alpha_m \geq 0 \quad (10)$$

The above optimization problem has to be solved by numerical methods (*e.g.* conjugate gradient search). However, it still remains computationally efficient because the number of parameters to be optimized, M, is a small constant in most cases.

3. Prior work

Time-series data classification is a very important problem in the computer vision and data mining communities. The recent studies can be largely divided into two categories: instance-based vs. model-based. In the former, the distance measure(or kernel) between a pair of sequences is defined, which is then delivered to discriminative classifiers such as kNN or SVMs. Dynamic Time Warping(DTW) is often used to estimate Euclidean distances between sequences with unequal lengths. Recently in [15], the constrained DTW was proposed to improve the classification performance on the time-series data. In [7] the kernel of two sequences was defined as the inner product of their Fisher scores with respect to the underlying generative model. Even though these methods may achieve high accuracies, the computational overhead to estimate the distance(or kernel) between sequences and a lack of clear generative interpretation is a major disadvantage. In a different set of approaches generative models can be used as prototypes for sequence clusters. The HMM is often used in this context in speech recognition [9], gesture recognition [22, 18], and gait recognition [5, 1]. The approach in [1] is closely related to the base BNC model in our paper, however it is trained to optimize a non-discriminative ML criteria.

Prior approaches to estimation of mixtures of Bayesian Networks have emerged in recent years [21, 16, 12]. Our recursive boosting algorithm for discriminative mixture learning is based on the functional gradient optimization of convex additive models. While similar gradient approaches have been introduced in the past [3, 11], they only provided heuristic methods for the component search or did not focus on mixtures of generative models. In [14], a mixture fitting problem, reduced to the joint log-likelihood cost functional optimization in the supervised setting, was solved in a non-heuristic way. Our algorithm is similar to the framework of [14], however, it is the first to derive the data weighing schemes for the discriminative cost functional, an appropriate cost model for the classification task.

There have been considerable researches that directly learn HMMs discriminatively via the maximum mutual information objective [13, 23]. Its major drawback is the computational overhead due to the gradient-based search.

4. Experiments

In this section, we demonstrate the classification performance of the proposed methods. We focus on the task of classifying sequences, a challenging problem in many areas of pattern recognition, also relevant to the human motion

[3] This optimization is very hard − at least as hard as CML of a single BNC model. We can skip this step without any harm.

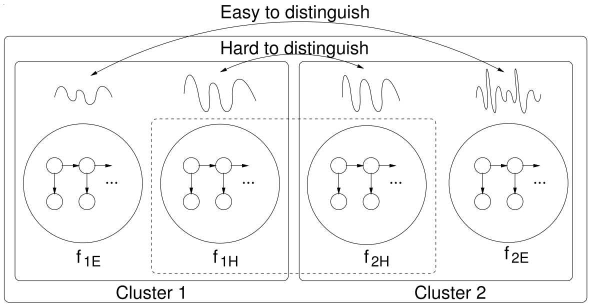
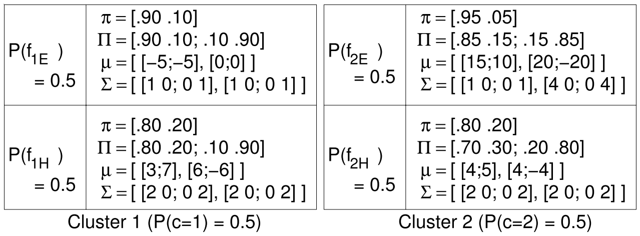


Figure 1. Schematic view of true model



$P(f_{1E})$ = 0.5	π = [.90 .10] Π = [.90 .10; .10 .90] μ = [[–5;–5], [0;0]] Σ = [[1 0; 0 1], [1 0; 0 1]]	$P(f_{2E})$ = 0.5	π = [.95 .05] Π = [.85 .15; .15 .85] μ = [[15;10], [20;–20]] Σ = [[1 0; 0 1], [4 0; 0 4]]
$P(f_{1H})$ = 0.5	π = [.80 .20] Π = [.80 .20; .10 .90] μ = [[3;7], [6;–6]] Σ = [[2 0; 0 2], [2 0; 0 2]]	$P(f_{2H})$ = 0.5	π = [.80 .20] Π = [.70 .30; .20 .80] μ = [[4;5], [4;–4]] Σ = [[2 0; 0 2], [2 0; 0 2]]
Cluster 1 (P(c=1) = 0.5)		Cluster 2 (P(c=2) = 0.5)	

Figure 2. Parameters of true model

modeling problem in computer vision. A NB with Gaussian HMM(GHMM) class conditionals serves as a base BNC model. We test the algorithms extensively on five datasets: two synthetic datasets and three real world time-series datasets. For each dataset, we compare the followings: (a) **ML**: base BNC model learned generatively(*e.g.* [1]), (b) **BML**: AdaBoost parameter boosting of [8], (c) **MixBML**: generative heuristic extension of BML learned via EM of Sec. 2.3, (d) **MixCML**: discriminative heuristic extension of BML via gradient search of Sec. 2.3, (e) **BoostML**: generative boosted mixture learning of [14], and (f) **BoostCML**: discriminative boosted mixture learning of Sec. 2.2. All algorithms are implemented in Matlab.

4.1. Synthetic data generated by GHMM clusters

In this experiment, the true model is devised as a mixture of four GHMMs, where its positive cluster is composed of two GHMMs, f_{1E} and f_{1H}, and its negative cluster is another mixture of two GHMMs, f_{2E} and f_{2H}. f_{1E} and f_{2E} are chosen in a way that they generate sequences which look very different, thus it is *easy* to distinguish the two. However, we constrain f_{1H} and f_{2H} to generate sequences similar to each other, that is, *hard* to classify. Thus they emit sequences on the classification boundary. The base BNC model to be learned has a *suboptimal* structure and contains only two GHMM kernels, one for each class. Figure 1 shows a schematic view of this synthetic model. For simplicity, we made all four GHMMs have the same order(the number of states) as 2. The dimension is 2. For the details, the parameters of the model are chosen as in Figure 2. The example sequences generated from this model are also depicted in Figure 3.

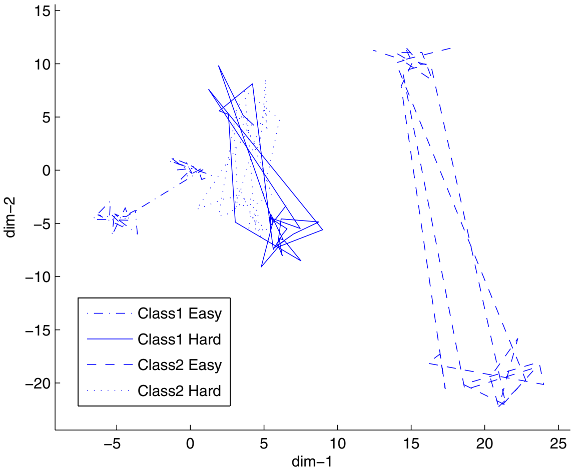


Figure 3. Example sequences generated by true model

The experiment was conducted by random 5-fold validation, where each fold comprised of a 50-sequence training set[4] generated by the true model. The test data consisted of 100 sequences held constant during the experiment. The sequence length was fixed to 30 samples, in order to compare the method with standard distance-based approaches. For simplicity, the order of the GHMMs in the base model was fixed at 2, which equaled that of the true model. The first component of BoostML and BoostCML was chosen as the ML model, even though this is not required by the algorithms. The maximum number of iterations of BoostML and BoostCML was set as 4. BoostCML, however, sometimes stopped earlier than 4 when the conditional log-likelihood score on the train data reached a value sufficiently close to 0. We also ran BML for 10 iterations, which seemed sufficiently large for convergence.

The average test errors and the normalized log-likelihood scores on the test data are shown in Table 1. The table also contains the test errors of 1NN classifiers, where the distance between sequences is estimated by either Dynamic Time Warping(DTW) or naive Euclidean distance. Among the set, BoostCML has the highest classification accuracy, meaning that it *boosts* the incorrect base model structure effectively. The other mixture models show significant improvement over the single ML model. However, the methods that utilize a generative objective tend to perform worse.

[4] We also performed the experiment with smaller(20-sequence) train data under the same configuration as that of the larger one. The test errors were: ML(19.00%), BML(6.80%), MixBML(7.00%), MixCML(6.00%), BoostML(2.80%), BoostCML(1.60%), DTW(6.20%), and Euc.(25.4%).

0-7695-2597-0/06 $20.00 © 2006 IEEE

Table 1. Results for GHMM synthetic data: Average test error and log-likelihood on the test data are shown. The log-likelihood score is normalized with respect to the sequence length.

	Test Error (%)	Log-Likelihood
ML	19.60	-5.51
BML	6.40	N/A
MixBML	5.40	-4.96
MixCML	5.20	-5.06
BoostML	4.20	-4.64
BoostCML	0.60	-5.15
1-NN DTW	2.60	N/A
1-NN Euc.	18.60	N/A

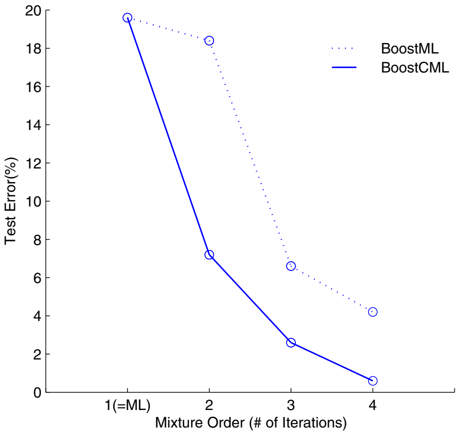


Figure 4. Classification performance improvement over the boosting iterations: BoostML vs. BoostCML

The log-likelihood scores of the mixture models are greater than that of the single model, which implies that the mixture models with proposed learning methods still enjoy the benefits of generative models, such as the richness in synthesis.

Furthermore, we compared BoostML and BoostCML in terms of performance improvement over the boosting iterations. The test errors of BoostML and BoostCML, as the mixture order varies from 1 to 4, are shown in Figure 4. Note that the order-1 mixture is ML. As the order increases, BoostCML achieves more dramatic decrease in the test error than BoostML, especially from order 1 to 2. From this, we can conclude that the discriminative mixture learning outperforms the generative counterpart.

4.2. Control Chart synthetic data

Table 2. Test error(%) on Control Chart dataset

ML	BML	MixBML	MixCML
9.31 ± 3.79	5.67 ± 2.50	5.51 ± 2.36	4.76 ± 2.16
BoostML	**BoostCML**	**1-NN DTW**	**1-NN Euc.**
8.07 ± 2.75	4.11 ± 1.44	2.73 ± 0.93	7.93 ± 1.42

The *control chart* dataset[5] contains 600 time-series sequences synthetically generated by a statistical process. There are six different classes[6], each of which has 100 sequences. Every sequence is 1D with the same length 60. Since it is visually very easy to distinguish class 2(Cyclic) from others, we eliminate class 2, thereby yielding a 5-class problem. The experiments were conducted with a random 10-fold validation, where each fold consists of 50 training sequences (10 sequences randomly from each of 5 classes) and 450 testing. The GHMM order was fixed at 2, which seemed best for generalization. The maximum number of iterations of the boosting algorithms was set as 10. For the algorithms based on the AdaBoost (*i.e*. BML, MixBML, and MixCML), we applied AdaBoost.M1 of [2].

The test error is shown in Table 2. DTW shows excellent classification performance, however, the drawback of DTW is that the computational complexity is quadratic in the number of sequences. On the other hand, all the proposed mixture learning algorithms have complexities linear in the number of sequences[7]. Among the generative models, BoostCML yields the best performance, which is comparable to 1-NN with DTW. We can also see that MixCML slightly improves BML. From this experiment, we can conclude that a mixture learned discriminatively works well in multi-class problem setting.

4.3. Gun / Point data

The binary class dataset contains 200 sequences (100 per class) of *gun draw*(class 1) and *finger point*(class 2).[8] The sequence is 1D, representing the x-coordinate of the centroid of the right hand, and the same length 150. This time-series dataset is a typical example where 1-NN with either a naive Euclidean distance or a DTW with small Sakoe-Chiba [17] band size constraints extremely works well[9].

Our experimental setting is as follows: 20 sequences (10 from each class) was randomly chosen as the training set, and the remaining 180 for the test set, repeated 10 times to form 10 random folds. The sequences were pre-processed by Z-normalization, making mean = 0, stdev = 1. The

[5] The dataset is obtained from [10].

[6] Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, and Downward shift.

[7] Here, we assume that the number of numerical steps in EM or gradient search for the proposed algorithms is considered constant with respect to the number of train/test sequences.

[8] For more information about the data, please refer to [10] from which the dataset is obtained.

[9] For the notational convenience, the ordinary DTW which has no band size constraints is denoted as either "DTW (∞%)" or just "DTW".

Table 3. Test error(%) on Gun / Point dataset

ML	36.22 ± 9.62	BoostCML	17.28 ± 5.67
BML	28.78 ± 13.75	DTW (∞%)	22.33 ± 5.75
MixBML	27.67 ± 11.73	DTW (10%)	15.72 ± 3.47
MixCML	27.39 ± 12.06	DTW (3%)	13.28 ± 4.83
BoostML	19.28 ± 6.15	1-NN Euc.	19.39 ± 5.60

GHMM order was fixed at 10 to describe 2-3 states for delicate movement around the subject's side, 2-3 states for hand movement from/to the side to/from the target, 1-2 states at the target, and 2-3 states for returning to the gun holster. The maximum number of iterations of the boosting algorithms was set as 10.

As shown in Table 3, 1-NN based on DTW with *properly* chosen Sakoe-Chiba band size(3%) outperformed the generative models. BoostCML still achieves competitive recognition accuracy, and best among generative models, on this difficult problem. Note also that the performance of 1NN varies significantly with the choice of the band parameter, yielding subpar performance of unconstrained DTW.

4.4. Australian Sign Language(ASL)

This dataset[10] is composed of about 100 signs generated by five signers with different levels of skills. In this experiment, we consider only 10 signs(hello, sorry, love, eat, give, forget, know, exit, yes, no). In the original ASL dataset, each time slice of a sequence consists of 15 features, corresponding to the hand position, hand orientation, finger flexion, and more. As recommended, we ignored the 5^{th}, 6^{th}, and 11^{th}-15^{th} features, yielding dimension 8.[11] In contrast with the Gun / Point dataset, the DTW is not very helpful because the lengths of sequences in the dataset are very diverse(*e.g.* $\mu = 57$, $\sigma = 16$, max = 296, min = 17). To prevent occasional noisy spikes in the original sequences, we preprocessed them by 1D median filter.

We consider to distinguish one sign from another, facing 45 binary classification problems. Instead of using all the sequences in the original dataset, we selected 20 sequences (4 from each of 5 signers) from each sign for the leave-one-out test. In other words, each problem has 40 folds, each of which is composed of a train set of 39 sequences and 1 test sequence. The GHMM order is chosen small(3) to prevent overfitting. The boosting algorithms iterate up to 10 steps. All the results are recorded in Table 4. Note that we included only the DTW with no band-size constraint, which was superior to those with constraints. Figure 5 depicts the scatter plots of test errors comparing BoostCML against others. As before, the mixture models significantly outperform a single model. In this instance, two heuristic methods yield highest accuracies, while BoostCML performs similarly well.

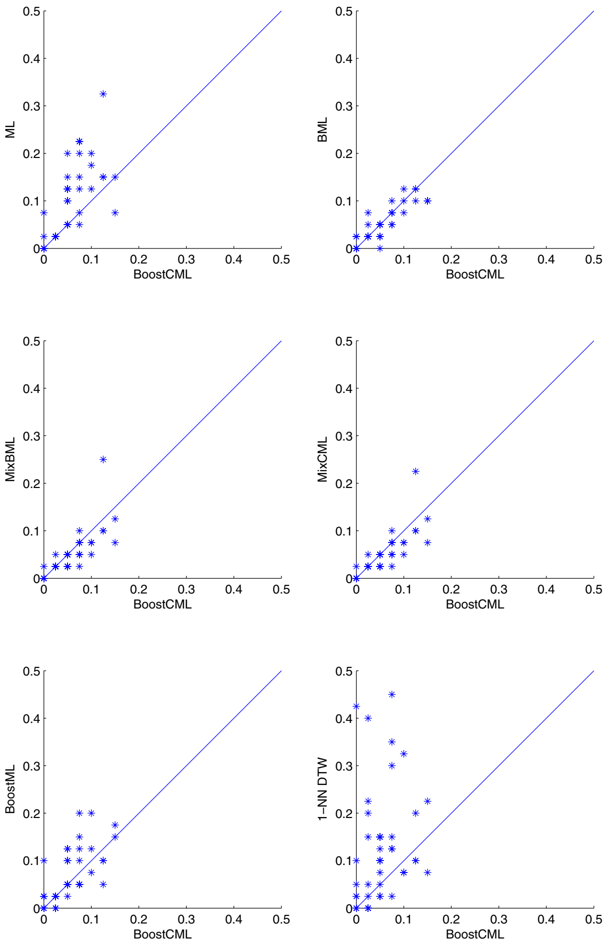


Figure 5. Test error scatter plots for Australian Sign Language dataset: Each point corresponds to each of 45 binary class problems.

4.5. Georgia-Tech Speed-Control Gait database

The proposed algorithms were also tested on the human gait recognition problem. The data of interest is the speed-control gait data[12] which was collected from Human Identification at a Distance(HID) project performed by computational perception laboratory at Georgia-Tech. The database is originally intended for studying distinctive characteristics(*e.g.* a stride length or a cadence) of human gait over

[10] The dataset is from UCI KDD archive [6].

[11] It seems visually reasonable to ignore the 7^{th}-10^{th} features because they correspond to less significant finger flexion. The authors conducted the same experiment by using only 6 features(1^{st}-6^{th}), where the result was more or less the same.

[12] The database is available online at "ftp://ftp.cc.gatech.edu/pub/gvu/cpl/walkers/speed_control_data/".

Table 4. Classification accuracy on Australian Sign Language for 45 binary class problems: The numbers represent how many folds (out of 40) are correctly classified. Higher scores are better.

Class 1	Class 2	ML	BML	Mix BML	Mix CML	Boost ML	Boost CML	1NN DTW	Class 1	Class 2	ML	BML	Mix BML	Mix CML	Boost ML	Boost CML	1NN DTW
hello	sorry	32	37	37	37	35	37	35	love	no	35	39	39	39	36	38	37
hello	love	32	36	37	37	35	36	37	eat	give	39	37	39	39	39	39	34
hello	eat	37	39	39	39	36	40	38	eat	forget	35	37	38	38	37	36	27
hello	give	40	40	40	40	40	40	39	eat	know	39	39	39	39	39	39	40
hello	forget	31	38	39	39	38	37	22	eat	exit	38	38	38	38	36	38	34
hello	know	39	39	39	39	40	39	40	eat	yes	40	40	40	40	39	40	40
hello	exit	34	37	36	36	34	37	39	eat	no	32	38	38	38	35	38	36
hello	yes	39	39	40	40	40	40	40	give	forget	39	39	39	39	39	39	24
hello	no	36	38	39	39	38	38	39	give	know	40	40	40	40	40	40	39
sorry	love	34	35	30	31	36	35	32	give	exit	35	40	39	39	35	38	39
sorry	eat	37	36	37	37	33	34	31	give	yes	39	39	39	39	39	39	40
sorry	give	38	38	38	38	39	38	34	give	no	36	38	39	39	38	38	38
sorry	forget	38	38	38	38	38	37	28	forget	know	39	39	39	39	39	39	38
sorry	know	39	39	39	39	40	39	40	forget	exit	39	39	39	39	39	39	32
sorry	exit	33	35	37	37	32	36	37	forget	yes	37	37	37	37	38	37	26
sorry	yes	40	40	40	40	39	40	36	forget	no	34	39	38	38	38	38	36
sorry	no	31	37	38	38	36	37	34	know	exit	39	39	39	39	39	39	39
love	eat	35	36	37	37	32	37	35	know	yes	34	35	36	36	38	35	36
love	give	39	38	38	38	39	39	31	know	no	38	38	38	38	38	38	36
love	forget	40	40	40	40	39	40	23	exit	yes	38	38	38	38	38	38	35
love	know	39	39	39	39	40	39	40	exit	no	35	38	38	38	38	38	34
love	exit	34	36	35	35	34	34	37	yes	no	27	36	36	36	36	35	36
love	yes	40	40	40	40	40	40	40	Average		36.5	38.0	38.2	38.2	37.5	37.9	35.2

different speeds [19, 20]. For each of 15 subjects, and for each of 4 different walking speeds(0.7m/s, 1.0m/s, 1.3m/s, 1.6m/s), 3D motion capture data of 22 marked points (as depicted in [19]) were recorded in a sophisticated way, for 9 sessions. The data was sampled at 120 Hz evenly for exactly one walking cycle, meaning that slower sequences were longer than faster ones. The sequence length ranges from approximately 100 to 200. Each marked point has a 3D coordinate, yielding 66(=22 $\cdot$ 3)-dim sequences.

Apart from the original purpose of the data, we are interested in recognizing *subjects* regardless of their walking speeds. Taking only the first 5 subjects into consideration, and not distinguishing their walking speeds, we formulated a 5-class problem where each class consists of 36(= 4 speeds $*$ 9 sessions) sequences. However, the dataset contains extremely good features for classification. For example, the single ML classifies almost all sequences correctly, which implies there is nothing to improve(or *boost*). In order to distinguish the performances of the algorithms, it is necessary to make the problem more difficult. We considered two modifications: (1) Instead of using the original one-cycle sequences, we took sub-sequences randomly where the sub-sequence length was normally distributed $\mathcal{N}(\mu = 80, \sigma = 0.1 * \mathrm{original} - \mathrm{length})$. Given the length, the starting point of the sub-sequence was selected uniformly at random. (2) The features related only to the *lower* body part were used: the joint angles of torso-femur, femur-tibia, and tibia-foot. Thus total 6 features (left/right each) were used.

With this manipulated data, each of 10 random folds of train/test data was constructed as follows: For each subject, and for each speed, we randomly picked 5 sequences (out of 9 sessions) for the train data. In other words, the train data had 100(=$5 \cdot 5 \cdot 4$) sequences from all subjects with all speeds evenly. The other 80(=$4 \cdot 5 \cdot 4$) sequences were put into a test dataset. The GHMM order was fixed at 3, the maximum number of boosting iterations was set as 20. Table 5 summarizes the results. We included only DTW with no band size constraints because the constrained DTWs work very poorly (*e.g.* test errors $> 25\%$). As in the ASL case, MixCML has the smallest test error. BoostCML, comparable to MixCML, outperforms 1-NN DTW. Therefore, the proposed algorithms are shown to work well in the human gait recognition problem as well.

Table 5. Test error(%) on Georgia-Tech Speed-Control Gait dataset

ML	BML	MixBML	MixCML	BoostML	BoostCML	1-NN DTW
11.50 ± 4.78	10.13 ± 3.61	4.50 ± 3.55	4.00 ± 3.48	11.87 ± 5.11	5.75 ± 2.78	8.38 ± 3.68

5. Conclusions

We introduced novel discriminative methods for learning mixtures of generative models. Unlike traditional approaches to discriminative learning of generative models, the proposed methods are highly computationally efficient. This makes them suitable for domains described by complex generative models, such as the spaces of time-series sequences. We show, through an extensive set of evaluations, that a mixture learned discriminatively is not only superior to a single generative model, but also comparable in performance to ensembles of discriminative models. Moreover, the mixture model continues to enjoy other benefits of generative models such as the potential for powerful synthesis of data. In our future work, we will further examine the proposed model's performance in data synthesis contexts. We will also consider principled ways to determine the order of the mixture, a problem related to model selection. Finally, we will examine other families of models, such the Markov random fields, that may benefit from our approach.

Acknowledgements

This work was supported by the NSF IIS grant 0413105.

References

[1] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data, 2003. IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[2] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1995. European Conference on Computational Learning Theory.

[3] J. Friedman. Greedy function approximation: a gradient boosting machine, 1999. Technical report, Dept. of Statistics, Stanford University.

[4] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers, 1997. Machine Learning.

[5] Q. He and C. Debrunner. Individual recognition from periodic activity using hidden Markov models, 2000. Workshop on Human Motion.

[6] S. Hettich and S. D. Bay. The UCI KDD archive [http://kdd.ics.uci.edu], 1999. Irvine, CA. University of California, Dept. of Information and Computer Science.

[7] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers, 1998. In Advances in Neural Information Processing Systems.

[8] Y. Jing, V. Pavlovic, and J. M. Rehg. Efficient discriminative learning of Bayesian Network Classifier via boosted augmented Naive Bayes, 2005. International Conference on Machine Learning.

[9] B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden Markov models, 1985. AT&T Technical Journal.

[10] E. Keogh and T. Folias. The UCR time series data mining archive [http://www.cs.ucr.edu/∼eamonn/TSDMA/], 2002. Riverside CA. University of California - Computer Science & Engineering Department.

[11] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses, 1999. In Advances in Large Margin Classifiers, MIT Press.

[12] C. Meek, B. Thiesson, and D. Heckerman. Staged mixture modelling and boosting, 2002. 18th Conference on Uncertainty in Artificial Intelligence, pp. 335-343.

[13] N. Oliver and A. Garg. MMIHMM: Maximum mutual information hidden Markov models, 2002. International Conference on Machine Learning.

[14] V. Pavlovic. Model-based motion clustering using boosted mixture modeling, 2004. IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[15] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints, 2004. In proceedings of SIAM International Conference on Data Mining, pp. 11-22.

[16] S. Rosset and E. Segal. Boosting density estimation, 2002. In Advances in Neural Information Processing Systems.

[17] H. Sakoe and C. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[18] T. Starner and A. Pentland. Real-time American sign language recognition from video using hidden Markov models, 1995. International Symposium on Computer Vision.

[19] R. Tanawongsuwan and A. Bobick. Characteristics of time-distance gait parameters across speeds, 2003. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, TR GIT-GVU-03-01.

[20] R. Tanawongsuwan and A. Bobick. Performance analysis of time-distance gait parameters under different speeds, 2003. 4th International Conference on Audio and Video Based Biometric Person Authentication, Guildford, UK.

[21] B. Thiesson, C. Meek, D. M. Chickering, and D. Heckerman. Learning mixtures of DAG models, 1998. In Proc. of the Conf. on Uncertainty in Artificial Intelligence.

[22] A. D. Wilson and A. F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[23] P. Woodland and D. Povey. Large scale discriminative training for speech recognition, 2000. Proceedings of the Workshop on Automatic Speech Recognition.