Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 13, No. 3, September 2013, pp. 186-199
http://dx.doi.org/10.5391/IJFIS.2013.13.3.186

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

Semi-Supervised Recursive Learning of Discriminative Mixture Models for Time-Series Classification

Minyoung Kim
Department of Electronics & IT Media Engineering,
Seoul National University of Science & Technology, Seoul, Korea

Abstract

We pose pattern classification as a density estimation problem where we consider mixtures of generative models under partially labeled data setups. Unlike traditional approaches that estimate density everywhere in data space, we focus on the density along the decision boundary that can yield more discriminative models with superior classification performance. We extend our earlier work on the recursive estimation method for discriminative mixture models to semi-supervised learning setups where some of the data points lack class labels. Our model exploits the mixture structure in the functional gradient framework: it searches for the base mixture component model in a greedy fashion, maximizing the conditional class likelihoods for the labeled data and at the same time minimizing the uncertainty of class label prediction for unlabeled data points. The objective can be effectively imposed as individual mixture component learning on weighted data, hence our mixture learning typically becomes highly efficient for popular base generative models like Gaussians or hidden Markov models. Moreover, apart from the expectation-maximization algorithm, the proposed recursive estimation has several advantages including the lack of need for a pre-determined mixture order and robustness to the choice of initial parameters. We demonstrate the benefits of the proposed approach on a comprehensive set of evaluations consisting of diverse time-series classification problems in semi-supervised scenarios.

Keywords: Mixture models, Bayesian networks, Semi-supervised learning, Functional gradient boosting, Time-series classification

Received: Jan. 9, 2013
Revised : Jun. 5, 2013
Accepted: Jun. 7, 2013

Correspondence to: Minyoung Kim
(mikim@snut.ac.kr)

1. Introduction

In a number of data-driven modeling tasks, a generative probabilistic model such as a Bayesian network (BN) is an attractive choice, advantageous in various aspects including the ability to easily incorporate domain knowledge, factorize complex problems into self-contained models, handle missing data and latent factors, and offer interpretability to results, to name a few [1, 2]. While such models are implicitly employed for joint density estimation, for the last few decades they have gained significant attention as *classifiers*. A model of this class, the Bayesian network classifier (BNC) [3], has been used in a wide range of applications subsuming speech recognition and motion time-series classification [4–9] and has been shown

to yield performance comparable to dedicated discriminative classifiers such as support vector machines (SVMs).

A BNC model represents a density $P(c, \mathbf{x})$ over the class variable c and observation $\mathbf{x}$. Learning its parameters with fully labeled data is traditionally posed as a joint likelihood maximization (ML). However, as the ML learning aims to fit the density for all points in the training data, it may not be directly compatible with the ultimate goal of class prediction. Instead, discriminative learning, typically the conditional likelihood maximization (CML), optimizes the conditional distribution of class *given* observation, i.e., $P(c|\mathbf{x})$, achieving better classification performance than ML learning in a variety of situations [10–12]. Unfortunately, CML optimization is, in general, complex with non-unique solutions. Typical CML learning methods are based on gradient search that can be computationally intensive.

A mixture model, as a rich density estimator, can potentially yield more accurate class prediction than a *single* BNC model. Mixture models have received significant attention in related fields and achieved success in diverse application areas [13–15]. In our earlier work [16] we proposed a quite efficient approach to the discriminative density estimation of *mixture models*. Here we briefly describe the algorithm introduced in [16]. The main goal is to exploit the properties of a mixture to alleviate the complexity of a learning task. This can be done in a greedy fashion, where a mixture component is added recursively to the current mixture with the objective of maximizing conditional likelihoods. Formulated within the functional gradient boosting framework [17], the procedure yields the weight distribution on the data with which a new mixture component can be learned. The derived weighting scheme effectively emphasizes the data points at the decision boundary, a desirable property similarly observed in SVMs.

The method is particularly efficient and easy to implement in that searching for a new mixture component can be done by ML learning with *weighted* data, and hence is suited to domains with complex component models such as hidden Markov models (HMMs) in time-series classifications that are usually computationally intensive for parametric gradient search. Compared to the conventional expectation-maximization (EM) algorithms, the recursive estimation approach has the crucial advantages of ease of model selection (i.e., estimating mixture orders) and robustness to initial model parameter choice.

Although our earlier approach was limited to fully supervised settings, in this paper we extend it to semi-supervised learning setups where we can make use of a large portion of unlabeled data points in conjunction with a few labeled data. We incorporate the minimum entropy principle in [18] into our recursive mixture estimation framework, where the unlabeled data points are exploited in such a way that the model's uncertainty in class prediction is maximally reduced. This leads to an objective function comprising the conditional log-likelihoods on labeled data and the negative entropy terms for unlabeled data. Within the functional gradient boosting framework, we derive the stage-wise data weight distribution for this semi-supervised objective.

The paper is organized as follows. In the next two sections, we formally set up the problem and review our earlier approach to the discriminative learning of mixtures in a fully supervised setup. Our proposed semi-supervised discriminative mixture learning algorithm is described in Section 4. In the experimental evaluation in Section 5, we demonstrate the benefits of the proposed algorithms in an extensive set of time-series classification problems on many real-world datasets in semi-supervised scenarios.

2. Problem Setup and Notation

Consider a classification problem where a class label is denoted by $c \in \{1, \ldots, K\}$ for the observation/feature $\mathbf{x} \in \mathcal{X}$. The input feature $\mathbf{x}$ is either vector-valued or structured like sequences of time-series. Let $f(c, \mathbf{x})$ denote a BNC[1] with a class variable c and the input attribute variables $\mathbf{x}$. A BNC can be usually factorized into a (multinomial) class prior $f(c)$ and the class conditional densities $f(\mathbf{x}|c) = f_c(\mathbf{x})$. For example, $f_c(\mathbf{x})$ could be a class(c)-specific Gaussian when $\mathbf{x}$ is a real-valued vector. Often, $f_c(\mathbf{x})$ may also contain latent variables (e.g., in the sequence classification where $\mathbf{x}$ is a sequence of measurements, $f_c(\mathbf{x})$ can be modeled as an HMM with hidden state variables).

[1] We use the notation $f(c, \mathbf{x})$ interchangeably to represent either a BNC or a likelihood at data point $(c, \mathbf{x})$.

As a classifier, the class prediction of a new observation $\mathbf{x}$ can be accomplished by the decision rule: $c^* = \arg\max_c f(c|\mathbf{x}) = \arg\max_c f(c, \mathbf{x})$. Given the (fully supervised) training data $\mathcal{D} = \{(c^i, \mathbf{x}^i)\}_{i=1}^n$, we learn a joint density $f(c, \mathbf{x})$ that minimizes the prediction error. The traditional ML learning optimizes the data joint likelihood $\sum_{i=1}^n \log f(c^i, \mathbf{x}^i)$. However, the ML learning does not necessarily yield optimal prediction performance unless we are given not only the correct model structure but also a large number of training samples.

The discriminative learning of BNCs effectively represents the class boundaries, and exhibits superior classification performance to ML learning that merely focuses on fitting the density to all points in the training data. CML learning, one of the most popular discriminative estimators, maximizes the conditional likelihood of c given $\mathbf{x}$, an objective directly related to the goal of accurate class prediction. The conditional log-likelihood objective for the training data $\mathcal{D}$ is defined as

$$\begin{aligned} CLL &= \sum_{i=1}^{n} \log f(c^i|\mathbf{x}^i) \\ &= \sum_{i=1}^{n} \left[\log f(c^i, \mathbf{x}^i) - \log f(\mathbf{x}^i) \right]. \end{aligned}$$

CML optimization in general does not admit closed-form solutions for most generative models. One typically maximizes it using gradient search. Although it has been shown that CML outperforms ML when the model structure is suboptimal [6, 10, 11, 19], the computational overhead demanded by gradient-based approaches is high, especially for complex models such as HMMs and general BN structures.

3. Previous Recursive Mixture Estimation in Fully Supervised Setups

Motivated by the fact that a single BNC can be insufficient for modeling complex decision boundaries (e.g., Gaussian class-conditionals merely represent ellipsoidal clusters), one can enlarge representational capacity by forming a *mixture*. Let $F(c, \mathbf{x})$ denote a mixture of BNCs, that is, $F(c, \mathbf{x}) = \sum_{m=1}^{M} \alpha_m f_m(c, \mathbf{x})$, where $\alpha_m \geq 0$ and $\sum_m \alpha_m = 1$. Note[1] that each component of the mixture is a BNC $f_m(c, \mathbf{x})$. Instead of the usual EM learning for mixture models, a greedy recursive approach was proposed in [16]. At each stage, we add a new BNC component $f(c, \mathbf{x})$ to the current mixture so that it optimizes a certain criterion.

Within the functional gradient optimization framework [17], one considers how to maximize a given objective functional $J(F)$ with respect to the (mixture) function $F(\mathbf{z})$ where $\mathbf{z} \in \mathcal{Z}$. In the classification setting, $\mathbf{z} = (c, \mathbf{x})$, and $\mathcal{Z}$ is the class-measurement joint input domain for the BNC likelihood function $f(c, \mathbf{x})$. The greedy optimization proceeds as follows: for the current mixture estimate F, we seek a new component f such that when F is locally varied as $(1-\epsilon)F + \epsilon f$ for some small positive ϵ, $J((1-\epsilon)F + \epsilon f)$ is maximally increased. The update equation is:

$$F \leftarrow (1-\epsilon)F + \epsilon f = F + \epsilon(f - F). \tag{1}$$

Maximizing $J(F)$ can be done by gradient ascent (in function space) described by the update rule:

$$F \leftarrow F + \delta \cdot \nabla_F J(F), \tag{2}$$

where δ is the step size and $\nabla_F J(F) = \partial J(F)/\partial F(\mathbf{z})$ is the functional gradient of $J(F)$ that is also a function obtained by a point-wise partial derivative.

Contrasting (2) with the greedy mixture update rule of (1), the optimal f would be the one that attains the maximal alignment between $(f - F)$ and $\nabla J(F)$, namely

$$\begin{aligned} f^* &= \arg\max_f \langle f - F, \nabla J(F) \rangle \\ &= \arg\max_f \langle f, \nabla J(F) \rangle. \end{aligned} \tag{3}$$

In the case of a finite number of samples $\{(c^i, \mathbf{x}^i)\}_{i=1}^n$, we estimate (3) as

$$f^* = \arg\max_f \sum_{i=1}^{n} w(c^i, \mathbf{x}^i) \cdot f(c^i, \mathbf{x}^i), \tag{4}$$

where $w(c, \mathbf{x}) = \nabla_{F(c,\mathbf{x})} J(F) = \partial J(F)/\partial F(c, \mathbf{x})$. Thus, $\nabla_{F(c,\mathbf{x})} J(F)$ serves as a weight for data point $(c, \mathbf{x})$ with which the new f will be learned. Optimization in (4) can be

[1] It is also worth noting that, if viewed from the generative perspective, this corresponds to modeling each class with the same number (M) of mixture components (i.e., $F(\mathbf{x}|c)$ for all c that have the same mixture order).

accomplished using a generic gradient ascent-based approach, however, a more efficient recursive EM-like lower-bound maximization was suggested in [16].

Once the optimal component f^* is selected, its optimal contribution to the mixture α^* is obtained as

$$\alpha^* = \arg \max_{\alpha \in [0,1]} J((1-\alpha)F + \alpha f^*). \tag{5}$$

This optimization can easily be done with any line search algorithm.

It is important to discuss the choice of the objective functional $J(F)$. For discriminative mixture learning, the conditional log-likelihood is employed in [16] by:

$$\begin{aligned} J_{Dis}(F) &= \sum_{i=1}^{n} \log F(c^i|\mathbf{x}^i) \\ &= \sum_{i=1}^{n} \log \frac{F(c^i, \mathbf{x}^i)}{\sum_c F(c, \mathbf{x}^i)}. \end{aligned} \tag{6}$$

In this case, the functional gradient becomes:

$$\begin{aligned} \frac{\partial J_{Dis}(F)}{\partial F(c^i, \mathbf{x}^i)} &= \frac{\partial \log F(c^i, \mathbf{x}^i)}{\partial F(c^i, \mathbf{x}^i)} - \frac{\partial \log F(\mathbf{x}^i)}{\partial F(c^i, \mathbf{x}^i)} \\ &= \frac{1}{F(c^i, \mathbf{x}^i)} - \frac{1}{F(\mathbf{x}^i)} \\ &= \frac{1 - F(c^i|\mathbf{x}^i)}{F(c^i, \mathbf{x}^i)}, \end{aligned} \tag{7}$$

yielding the discriminative data weight:

$$w_{Dis}(c, \mathbf{x}) = \frac{1 - F(c|\mathbf{x})}{F(c, \mathbf{x})}. \tag{8}$$

The discriminative weight indicates that the new component f is learned from the weighted data where the weights are directly proportional to $1 - F(c|\mathbf{x})$ and inversely proportional to $F(c, \mathbf{x})$. Hence the data points *unexplained by the model*, i.e., $F(c, \mathbf{x}) \to 0$, and *incorrectly classified* by the current mixture, i.e., $(1 - F(c|\mathbf{x})) \to 1$, are focused on in the next stage. This is an intuitively desirable strategy for improving the classification performance.

The time complexity of discriminative mixture learning is of the order $O(M \cdot (N_{ML} + N_{LS}))$ where N_{ML} stands for the complexity of the ML learning and N_{LS} is the complexity of the line search. Hence, the discriminative mixture learning algorithm complexity is a constant factor of simple generative learning of the base model on weighted data.

4. Semi-Supervised Recursive Discriminative Mixture Estimation

So far, we have considered the case where the data is fully labeled. In the semi-supervised setting, we are given the labeled set $\mathcal{L} = \{(c^i, \mathbf{x}^i)\}_{i=1}^{l}$ and the unlabeled data $\mathcal{U} = \{\mathbf{x}^j\}_{j=l+1}^{n}$. Among several known semi-supervised classification approaches, an effective way to exploit the unlabeled data is the entropy minimization method proposed by [18]. The main idea is that we minimize the classification error for the labeled data (e.g., maximizing the conditional likelihood), while forcing the model to have minimal uncertainty in predicting class labels for the unlabeled data. This minimum entropy principle is motivated by minimization of the Kullback-Leibler divergence between the model-induced distribution and the empirical distribution on the unlabeled data that has been shown to effectively partition the unlabeled data into clusters.

Having the negative entropy term for the unlabeled data, the semi-supervised discriminative (SSD) objective can be defined as

$$\begin{aligned} J_{SSD}(F) = &\sum_{i \in \mathcal{L}} \log F(c^i|\mathbf{x}^i) \\ &+ \gamma \sum_{j \in \mathcal{U}} \sum_{c} F(c|\mathbf{x}^j) \log F(c|\mathbf{x}^j), \end{aligned} \tag{9}$$

where $\gamma \geq 0$ is a controllable parameter that balances the loss term against the negative entropy term. The functional gradient for the new objective is now

$$\frac{\partial J_{SSD}}{\partial F(\mathbf{u})} = \begin{cases} \dfrac{1 - F(c^i|\mathbf{x}^i)}{F(c^i, \mathbf{x}^i)} & \text{if } \mathbf{u} = (c^i, \mathbf{x}^i) \in \mathcal{L} \\ \gamma \dfrac{\partial \sum_c F(c|\mathbf{x}^j) \log F(c|\mathbf{x}^j)}{\partial F(\mathbf{u})} & \text{if } \mathbf{u} = \mathbf{x}^j \in \mathcal{U}. \end{cases}$$

Notice that for the labeled data we have a functional gradient identical to that of supervised discriminative mixture learning. For the unlabeled data, however, the gradient terms require further consideration. The main difficulty is that for the unla-

beled data $\mathbf{x}^j$ ($\in \mathcal{U}$), we have no assigned class labels. We next consider two different approaches for treating this latent label.

4.1 Marginalization Over Full Label Set

A possible treatment is to assume that we are given all K class labels attached to the unlabeled data $\mathbf{x}^j$. That is, for each data point $\mathbf{x}^j$, we pretend that all possible K pairs $\{(c^j, \mathbf{x}^j)\}_{c^j=1}^K$ are observed in the training data. Then it follows that:

$$\begin{aligned}
&\frac{\partial}{\partial F(c^j,\mathbf{x}^j)} \sum_c F(c|\mathbf{x}^j) \log F(c|\mathbf{x}^j) \\
&= \frac{1}{F(\mathbf{x}^j)^2} \cdot \left(\frac{\partial \sum_c F(c,\mathbf{x}^j) \log F(c|\mathbf{x}^j)}{\partial F(c^j,\mathbf{x}^j)} \cdot F(\mathbf{x}^j) \right. \\
&\qquad \left. - \sum_c F(c,\mathbf{x}^j) \log F(c|\mathbf{x}^j) \right) \\
&= \frac{1}{F(\mathbf{x}^j)^2} \cdot \left(\frac{\partial F(c^j,\mathbf{x}^j) \log F(c^j,\mathbf{x}^j)}{\partial F(c^j,\mathbf{x}^j)} \cdot F(\mathbf{x}^j) \right. \\
&\qquad - \frac{\partial F(\mathbf{x}^j) \log F(\mathbf{x}^j)}{\partial F(c^j,\mathbf{x}^j)} \cdot F(\mathbf{x}^j) \\
&\qquad \left. + H(F(c|\mathbf{x}^j)) \cdot F(\mathbf{x}^j) \right) \\
&= \frac{1}{F(\mathbf{x}^j)^2} \cdot \left(\left(\log F(c^j,\mathbf{x}^j) - \log F(\mathbf{x}^j) \right) \cdot F(\mathbf{x}^j) \right. \\
&\qquad \left. + H(F(c|\mathbf{x}^j)) \cdot F(\mathbf{x}^j) \right) \\
&= \frac{\log F(c^j|\mathbf{x}^j) + H(F(c|\mathbf{x}^j))}{F(\mathbf{x}^j)}, \qquad (10)
\end{aligned}$$

where $H(\cdot)$ is the entropy function.

Hence, the unlabeled data point $\mathbf{x}^j$ induces K data weights:

$$w_{SSD}(c^j,\mathbf{x}^j) \propto \frac{\log F(c^j|\mathbf{x}^j) + H(F(c|\mathbf{x}^j))}{F(\mathbf{x}^j)}, \quad \text{for } c^j = 1,\ldots,K \qquad (11)$$

The unlabeled data weight can be interpreted as follows: (i) The denominator $F(\mathbf{x}^j)$ implies the need to focus on the samples that are less highlighted by the current model (regardless of their class labels) in the next stage. (ii) The first term in the numerator $\log F(c^j|\mathbf{x}^j)$ encourages the model to keep attending to its current decision (c^j) on $\mathbf{x}^j$. (iii) The entropy term in the numerator assigns more weights to the unlabeled samples $\mathbf{x}^j$ that have a higher prediction uncertainty in the current model. So, by (ii) and (iii), one can achieve entropy minimization for the unlabeled data.

Despite this intuitive interpretation, one practical issue with this weighting scheme is that the weights can be potentially negative, in which case the optimization in (4) may not be tackled by the lower bound maximization technique. In this case, one can directly optimize it using a parametric gradient search. Alternatively, the pseudo label-based technique presented next can circumvent the negative weight issue.

4.2 Pseudo Labels

For the current model, we define the *pseudo label* for $\mathbf{x}^j$ as: $c_*^j = \arg\max_c F(c|\mathbf{x}^j)$. Instead of dealing with all K possible labels for $\mathbf{x}^j$, we consider only a single pseudo-labeled pair $(c_*^j, \mathbf{x}^j)$. That is, the unlabeled $\mathbf{x}^j$ is assumed to be accompanied by c_*^j having the following weight:

$$w_{SSD}(c_*^j,\mathbf{x}^j) \propto \frac{\log F(c_*^j|\mathbf{x}^j) + H(F(c|\mathbf{x}^j))}{F(\mathbf{x}^j)}, \quad \text{where } c_*^j = \arg\max_c F(c|\mathbf{x}^j) \qquad (12)$$

The intuition discussed in Section 4.1 follows immediately, however, we can now guarantee that the weights for the pseudo-labeled data points (11) are always non-negative.

Although dealing with only the best predicted label is convenient for optimization and is a rational strategy to pursue, it is important to note that unlike the *all-label* approach in Section 4.1 the *pseudo-label* approach is suboptimal in the objective perspective, essentially amounting to ignoring the negative-weight (pushing-away) effects enforced by the non-best labels.

5. Evaluation

We evaluated the performance of the proposed recursive mixture learning in semi-supervised learning settings. We focused on the structured data classification task of classifying *sequences* or *time-series*. This is, in general, more difficult than the static multivariate data classification. We used Gaussian-emission HMMs (GHMMs)[1] to model the class conditional densities $f_c(\mathbf{x})$ for the real multivariate sequence $\mathbf{x}$. In our recursive

[1] Selecting the number of hidden states in GHMMs is an important task of model selection that we accomplish using cross validation.

mixture learning, we need to learn GHMMs with weighted data samples, and this can be done by a fairly straightforward extension of the regular EM-based GHMM learning. The detailed EM steps can be found in [20]. The competing approaches whose performance will be contrasted are summarized in Section 5.1, while in Section 5.2 we describe the datasets and report the results.

5.1 Competing Methods

A simple and straightforward approach to dealing with partially labeled data is to ignore the unlabeled data points. In this section, we first summarize the fully supervised classification algorithms with which we compare our approach. These algorithms are then extended to handle unlabeled data by the well-known and generic adaptive semi-supervised method called *self-training*.

The first approaches we describe are model-based, where we use (single) BNC models trained by ML and CML. In CML, the gradient search starts with the ML estimate as the initial iterate. Related to the proposed recursive discriminative mixture learning, we compare the proposed method with [21]'s *boosted Bayesian network* (BBN), an ensemble-based discriminative learning method for BNCs that treat $f(c, \mathbf{x})$ as a (weak) hypothesis, namely $c = h(\mathbf{x}) = \arg\max_c f(c|\mathbf{x})$, within a boosting [22] framework. For each stage, AdaBoost's weights w on data $(c, \mathbf{x})$ are used to learn the next hypothesis (BNC) via *weighted* ML learning: $\arg\max_f \sum_{i=1}^{n} w_i \log f(c^i, \mathbf{x}^i)$. This approach has been shown to inherit certain benefits from AdaBoost such as good generalization by maximizing the margin. However, the resulting ensemble cannot be simply interpreted as a generative model since the learned BNCs are just weak classifiers to be combined for the classification task.

In addition to the model-based approaches, we also consider two alternative similarity-based approaches that have exhibited good performance in the past, especially on sequence classification problems: dynamic time warping (DTW) and the Fisher kernel [23]. DTW is a dynamic programming algorithm that searches for the globally best warping path. Often, imposing certain constraints on the feasible warping paths has been empirically shown to improve the classification performance [24–26]. For instance, the Sakoe-Chiba band constraint [24] restricts the maximum deviation of matching slices from the diagonal by $p\%$ of the sequence length. Thus $p = 0$ and $p = \infty$ correspond to the naive Euclidean distance (defined only if the lengths of two sequences are equal) and the standard (unconstrained) DTW, respectively. Recently, [26] proposed an adaptive band approach that estimates the function spaces of time warping paths. In this setting, class-specific warping-path constraints are learned for each class that reflect the warping variations of the samples within it.

The Fisher kernel between two sequences $\mathbf{x}$ and $\mathbf{x}'$ is defined as the radial basis function (RBF) evaluated on the distance between their Fisher scores with respect to the underlying generative model. More specifically, in binary classification, $k(\mathbf{x}, \mathbf{x}') = e^{-||U_{\mathbf{x}} - U_{\mathbf{x}'}||^2/(2\sigma^2)}$, where $U_{\mathbf{x}} = \nabla_\theta \log P_{c=+}(\mathbf{x})$. Here $P_{c=+}(\mathbf{x})$ indicates the likelihood of the HMM usually learned by ML from the examples of the positive class only. The RBF scale σ^2 is determined as the median distance between the Fisher scores corresponding to the training sequences in the positive class and the closest Fisher score from the negative class in the training data [23]. The multi-class extension is made using a set of one-vs-rest binary problems.

As a baseline, we also consider a static classifier (e.g., SVM) that treats fixed-length (window) segments from a sequence as iid multivariate samples. Specifically, for a window of size r, the class-sequence data pair $(c, \mathbf{x})$ is converted to rd-dim iid samples, $(c, vec([\mathbf{x}_{t-\lfloor \frac{r}{2} \rfloor}, \ldots, \mathbf{x}_{t+\lfloor \frac{r-1}{2} \rfloor}]))$ for $t = \lfloor \frac{r}{2} \rfloor + 1, \cdots$. At the test stage, the class label is determined by majority voting over the predicted segment labels.

The competing methods are summarized below:

- **ML**: ML learning of $f(c, \mathbf{x})$.
- **CML**: CML learning of $f(c, \mathbf{x})$.
- **RDM**: Recursive discriminative mixture learning [16].
- **BBN**: Boosted Bayesian networks [21].
- **NN-DTW** ($B\%$): The Nearest Neighbor classifier based on the DTW distance measure where B is the best Sakoe-Chiba band constraint selected by cross validation over the candidate set: $\{\infty\%, 30\%, 10\%, 3\%\}$.

- **FS-DTW**: The function-space DTW learning [26].
- **SVM-FSK**: The SVM classifier based on the Fisher kernel. The SVM hyperparameters are selected by cross validation. To handle multi-class settings, we perform binarization in the *one-vs-rest* manner. We then employ the *winner-takes-all* (WTA) strategy which predicts the multi-class labels by majority voting from the outputs of the one-vs-others binarized problems[1].
- **SVM-Win** ($R\%$): An SVM classifier that treats fixed-length window segments as iid multivariate samples where R is the relative window size with respect to the sequence length ($R = 100r/T$). We use the RBF kernel in rd dimensional vector space. We report the best (relative) window size R selected by cross validation over a candidate set: $\{0\%$ (window size $r = 1), 10\%, 20\%, 30\%, 50\%\}$.
- **SSRDM**: Semi-supervised recursive discriminative mixture learning (proposed approach).

In the experiments, we split the data three-fold: labeled training data, unlabeled training data, and test data. All the other approaches listed above are *fully supervised*, making use of only the labeled data for training. On the other hand, our semi-supervised discriminative mixture learning algorithm (denoted by SSRDM) exploits the unlabeled training data in conjunction with the labeled data. Throughout the evaluation we make use of the *all-label* strategy in (11) as it consistently demonstrated performance superior to the pseudo-label alternative.

We not only demonstrate the improvement in prediction performance achieved by SSRDM compared to supervised methods that ignore the unlabeled data, but we also contrast it with the generic *self-training* algorithm, a generic method of extending fully supervised classifiers to semi-supervised setups, often very successful and the most popular method in use. We apply the self-training algorithm to each of the supervised methods listed above. The self-training algorithm is described in pseudocode in Algorithm 5.1.

[1]Alternatively, one can directly tackle multi-class problems via multi-class SVM [27]. The other possibility in binarization is the *one-vs-one* treatment [28, 29]. In our evaluation, however, WTA in one-vs-others settings slightly outperforms these two alternatives almost all the time, hence we only report the results of WTA.

Algorithm 1 Self-Training.

Input: Labeled ($\mathcal{L} = \{(c^i, \mathbf{x}^i)\}_{i=1}^{l}$) and unlabeled ($\mathcal{U} = \{\mathbf{x}^j\}_{j=l+1}^{n}$) data.
Output: Learned classifier $c = h(\mathbf{x})$.
Procedure:
Set $\mathcal{D} = \mathcal{L}$.
Repeat the following steps until convergence:
Do supervised learning with $\mathcal{D}$ to get new model h^{new}.
Determine the labels for $\mathcal{U}$ using h^{new}.
Add $\{(c^j, \mathbf{x}^j)\}_{j\in\mathcal{U}}$ to $\mathcal{D}$. (Replace old ones if available.)

Unless stated otherwise, for the mixture/ensemble approaches (i.e., BBN, RDM, and SSRDM), the maximum number of iterations (i.e., the number of BNC components) was set to ten. The test errors for the datasets (described in the next section) are shown in Table 1.

5.2 Datasets and Results

5.2.1 Gun/Point dataset

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

Weform five folds for cross validation with $10\%/40\%$ labeled/unlabeled training data and the remaining 50% for the test data, randomly . The sequences were pre-processed by Z-normalization so that the mean = 0 and standard deviation = 1. The GHMM order was chosen to be ten as it is also meaningful for describing 2–3 states for delicate movements 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 test errors (means and standard deviations) are shown in Table 1. NN-DTW with *properly* chosen Sakoe-Chiba band size (10%) outperforms ML, CML, and sequence kernel based SVM, while it is comparable to RDM. The semi-supervised learning results indicate that the SSRDM outperforms the other semi-supervised methods and significantly improves on supervised

[1]For further details about the data, please refer to [30].

Table 1. Test errors (%) for the semi-supervised settings

	Gun/Point	Australian Sign Language	GaTech Gait	USF Gait	Traffic	Face	Mouse
ML	33.89 ± 4.84	39.50 ± 6.47	18.62 ± 2.53	57.50 ± 5.98	23.17 ± 2.04	68.75 ± 1.65	53.09 ± 1.52
(Self-Tr ML)	(33.78 ± 4.67)	(39.25 ± 6.10)	(17.93 ± 1.62)	(56.07 ± 5.59)	(21.95 ± 2.67)	(66.25 ± 1.58)	(52.73 ± 2.23)
CML	26.89 ± 3.88	35.25 ± 6.87	16.00 ± 1.02	55.36 ± 5.65	19.51 ± 2.67	67.50 ± 1.55	49.82 ± 2.07
(Self-Tr CML)	(25.67 ± 3.65)	(35.25 ± 6.87)	(15.86 ± 1.29)	(55.00 ± 5.98)	(18.70 ± 2.95)	(66.67 ± 1.80)	(49.82 ± 2.07)
BBN	33.78 ± 4.88	34.00 ± 4.18	15.59 ± 4.67	57.50 ± 5.98	18.29 ± 2.56	66.04 ± 2.03	52.00 ± 2.76
(Self-Tr BBN)	(34.22 ± 3.16)	(32.75 ± 7.52)	(15.03 ± 5.05)	(56.07 ± 5.59)	(17.89 ± 2.52)	(66.46 ± 2.70)	(50.55 ± 2.37)
NN-DTW	17.44 ± 2.38	38.00 ± 2.59	23.03 ± 3.07	60.00 ± 1.60	47.15 ± 2.95	68.13 ± 0.93	56.00 ± 1.52
(Self-Tr NN-DTW)	(17.44 ± 2.38)	(38.00 ± 2.59)	(23.03 ± 3.07)	(60.00 ± 1.60)	(47.15 ± 2.95)	(68.13 ± 0.93)	(56.00 ± 1.52)
FS-DTW	19.33 ± 0.91	42.50 ± 3.85	30.76 ± 3.26	65.36 ± 3.24	49.19 ± 3.91	65.63 ± 2.33	57.09 ± 1.63
(Self-Tr FS-DTW)	(19.00 ± 0.91)	(42.25 ± 4.28)	(31.31 ± 3.40)	(65.36 ± 4.11)	(48.78 ± 5.12)	(64.58 ± 2.33)	(57.09 ± 1.63)
SVM-FSK	29.11 ± 6.65	38.25 ± 7.10	19.17 ± 2.73	56.43 ± 5.14	20.33 ± 1.99	66.88 ± 3.07	50.55 ± 5.66
(Self-Tr SVM-FSK)	(28.67 ± 5.38)	(37.50 ± 7.55)	(18.21 ± 2.05)	(56.07 ± 4.82)	(20.33 ± 1.99)	(66.67 ± 3.29)	(49.45 ± 6.85)
SVM-Win	28.11 ± 7.75	65.00 ± 3.64	25.24 ± 3.43	66.07 ± 2.19	40.24 ± 2.56	76.88 ± 4.74	59.27 ± 1.63
(Self-Tr SVM-Win)	(29.56 ± 11.97)	(67.00 ± 5.42)	(26.21 ± 3.48)	(66.43 ± 2.65)	(41.06 ± 2.85)	(77.29 ± 5.18)	(58.91 ± 1.63)
RDM	14.22 ± 2.79	34.50 ± 3.71	12.14 ± 1.86	51.43 ± 2.93	14.63 ± 2.18	65.83 ± 1.55	49.82 ± 2.44
(Self-Tr RDM)	(13.89 ± 2.08)	(33.75 ± 5.08)	(12.83 ± 1.25)	(50.71 ± 3.70)	(16.67 ± 1.84)	(62.92 ± 3.09)	(50.00 ± 1.05)
SSRDM	12.44 ± 2.31	29.75 ± 3.47	10.07 ± 1.73	49.29 ± 2.40	13.01 ± 1.99	62.71 ± 2.89	47.27 ± 2.23

The proposed SSRDM, located at the bottom with the boldfaced title, is compared with supervised classifiers that simply ignore unlabeled data and their self-training extensions (depicted in parentheses). ML, likelihood maximization; CML, conditional likelihood maximization; BBN, boosted Bayesian network; NN, nearest neighbor; DTW, dynamic time warping; FS, function space; SVM, support vector machine; FSK, Fisher kernel; RDM, recursive discriminative mixture; SSRDM, semi-supervised RDM.

RDM by taking advantage of the large number of unlabeled data.

5.2.2 Australian Sign Language (ASL)

This UCI-KDD dataset contains about 100 signs generated by five signers with different levels of skill [31]. In this experiment, we considered 10 selected signs ("hello," "sorry," "love," "eat," "give," "forget," "know," "exit," "yes," and "no"), forming a $K = 10$-way classification problem. 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 so on. As recommended, we ignored the 5^{th}, 6^{th}, and 11^{th}–15^{th} features. To prevent occasional noisy spikes in the original sequences, we additionally preprocessed them with a median filter. In contrast to the Gun/Point dataset, DTW is not very effective here because the lengths of sequences in the dataset are diverse, ranging from 17 to 196. We split the data randomly into 60% labeled and 20% unlabeled training data

with 20% test data in five folds. For the HMM-based models, the GHMM order was chosen to be three from cross validation.

Results in Table 1 show the test errors averaged over the five test folds. The DTW with the best-chosen band constraint ($B = 30$) exhibits a rather poor performance, statistically indistinguishable from ML, as expected due to the large deviation in sequence lengths. Compared to ML, the discriminative approaches like CML and RDM improve the prediction accuracy considerably. Despite the small number of unlabeled data, the proposed SSRDM effectively takes advantage of them, yielding the lowest test error significantly below the random guess error rate of 90%.

5.2.3 Georgia-Tech speed-control gait database

We next tested the proposed mixture learning algorithms on the human gait recognition problem. The data of interest is the speed-control gait data collected by the Human Identification at a Distance (HID) project at Georgia-Tech. The database was originally intended for studying distinctive characteristics (e.g., stride length or cadence) of human gait over different speeds [32, 33]. For 15 subjects, and four different walking speeds (0.7 m/s, 1.0 m/s, 1.3 m/s, 1.6 m/s), 3D motion capture data of 22 marked points (as depicted in [32]) were recorded for nine repeated sessions. The data was sampled at 120 Hz evenly for exactly one walking cycle, meaning that slower sequences were longer than the faster ones. The sequence length ranged from approximately 100 to 200 samples. Each marked point had a 3D coordinate, yielding 66 ($= 22 \times 3$) dimensional sequences.

Apart from the original purpose of the data, we were interested in recognizing *subjects* regardless of their walking speeds. Taking only the first five subjects into consideration without distinguishing their walking speeds, we formulated a 5-class problem where each class consisted of 36 ($= 4$ speeds $\times$ 9 sessions) sequences. The original dataset provided high-quality 3D motion capture features on which most of the competing methods performed equally well. To make the classification task more challenging, we considered two modifications: (1) From the original 1-cycle gait sequence, we took sub-sequences randomly where the starting positions were chosen uniformly at random and the lengths were around 100. (2) Only the features related to the *lower* body part were used: the joint angles of the torso-femur, femur-tibia, and tibia-foot.

After this manipulation, we randomly partitioned the data five time into 20% labeled and 50% unlabeled training data with the remaining 30% test data. The GHMM order was chosen to be three, and the maximum number of mixture learning iterations was set to 20. As Table 1 demonstrates, the proposed SSRDM again attains the lowest errors.

5.2.4 USF human ID gait dataset

The USF human ID gait dataset consists of about 100 subjects periodically walking in elliptical paths in front of a set of cameras. We considered the task of motion-based subject identification, where the motion videos were recorded in diverse circumstances: the subject walking on grass or concrete, with or without a briefcase. From the processed human silhouette video frames, we computed the 7^{th}-order Hu moments that are translation and rotation invariant descriptors of binary images. The extracted features were then Z-normalized, yielding 7-dimensional sequences of duration ~ 200. While the original investigation of the set focused on how well the classifiers adapted to new circumstances (i.e., a different combination of covariates), we concentrated on identifying humans regardless of the covariates. For this, we chose seven humans from the database (a 7-class problem), each of which had 16 associated sequences containing all combinations of circumstances.

After randomly splitting the 112 sequences into 50% labeled-training, 25% unlabeled-training, and 25% test sets five times, we recorded the average test errors in Table 1. The GHMM order was chosen to be three from cross validation. The maximum number of iterations for recursive mixture models was set to 20. Again, RDM and SSRDM have the lowest test errors with small variances, reaffirming the importance of the recursive estimation of discriminative mixture models when combined with the use of unlabeled data.

5.2.5 Traffic dataset

We next tackle a video classification problem that has demonstrated the utility of dynamic texture methods [34, 35] in the computer vision community. Dynamic texture is a generative model that represents a video as a sample from a linear dynam-

ical system. Dynamic texture can extract the visual or spatial components in the image measurements using PCA while capturing the temporal correlation by the latent linear dynamics. Hence, a video, potentially of varying length, can be succinctly represented by two matrices (A, C), where A is the dynamics matrix on the low-dimensional latent space, and C is the emission matrix that maps the latent state to the image observation. To apply dynamic texture to the video classification problems, the Martin distance [34] is often employed. It defines the similarity measure (or kernel) between a pair of videos based on the principal angles between subspaces represented by their matrix parameters. Once the distance measure is estimated, one can readily employ standard classifiers such as nearest neighbors or SVMs.

The dataset we used in this experiment is traffic data (also used in [36]) that contains videos of highway traffic taken over two days from a stationary camera. The videos were labeled manually as light, medium, and heavy traffic, posing a 3-class problem. The videos are around 50 frames long, where each image frame is of size (48×48), yielding a 2304-dimensional vector.

For the dynamic texture approach, we set the latent space dimension to be eight. We used the same dimension for our GHMM-based competing approaches, where we used the PCA dimension-reduced observation in the GHMMs. We collected 131 videos (with a nearly equal number of videos for each class), and randomly split them into 60% labeled-training, 10% unlabeled-training, and 30% test sets five times. The GHMM order was chosen to be two, and the maximum number of iterations for recursive mixture models was set to 20. The SVM classifier with the Martin distance measure estimated from the learned dynamic texture recorded a test error of $16.67 \pm 1.84\%$, outperforming many of the competing approaches as shown in Table 1. However, the proposed SSRDM (and RDM) achieved still higher prediction accuracies than the dynamic texture model.

5.2.6 Behavior recognition

Finally, we deal with a behavior recognition task, a very important problem in computer vision. We used the facial expression and mouse behavior datasets from the UCSD vision group[1]. The face data are composed of video clips of two individuals, each displaying six different facial expressions (anger, disgust, fear, joy, sadness, and surprise) under two different illumination settings. Each expression was repeated eight times, yielding a total of 192 video clips. We used 96 clips from one subject (regardless of illumination conditions) as training data, and predicted the emotions of other subjects in the video clips (a 6-class problem). We further randomly partitioned the training data into 50% labeled and 50% unlabeled sets five times. The mouse data contained videos of five different behaviors (drink, eat, explore, groom, and sleep). From the original dataset, we formed a smaller set comprising 75 video clips (15 videos for each behavior). We then randomly split the data into 25% labeled training, 40% unlabeled training, and 35% test sets five times.

In both cases, from the raw videos, we extract the *cuboid* features of [37] that are spatio-temporal 3D interest point features. Similar to [37], we constructed a finite dictionary of descriptors, and replaced each cuboid descriptor by a corresponding word in the dictionary. More specifically, we collected cuboid features from all training videos, clustered them into C centers using the k-means algorithm, and replaced each cuboid by its closest center ID.

For the classification, we first ran the static mixture approach in [37] as a baseline, where they represented a video as a histogram of cuboid types, essentially forming a bag-of-words representation. They then applied the nearest neighbor prediction using the χ^2 distance measure over the histogram space. Setting $C = 50$ with other cuboid parameters properly chosen, we obtained test errors of $68.75 \pm 2.95\%$ for the face dataset and $52.36 \pm 0.81\%$ for the mouse dataset. (Note that random guessing would yield 83.33% and 80.00% error rates, respectively.)

Instead of representing the video as a single histogram, we considered a sequence representation for our GHMM-based sequence models. For each time frame t, we collected all cuboids that spread over t and formed a histogram of cuboid types for it. Hence, we formed a C-dimensional histogram feature vector for each time slice t, where we used GHMMs to model the non-

[1] Available for download at http://vision.ucsd.edu.

negative quantities (histograms). Note that some time slices did not contain any cuboids, in which case the feature vector was a zero-vector. To avoid a large number of parameters in GHMM learning, we further reduced the dimensionality of the features to five dimensions with PCA.The test errors of the competing approaches for this sequence representation are recorded in Table 1. Here the best GHMM orders are three for the face dataset and four for the mouse dataset. For both cases, our discriminative recursive mixture learning algorithms (RDM and SSRDM) consistently exhibited the best performance within the margin of significance, outperforming [37]'s baseline method.

5.3 Discussion

The experimental results for the semi-supervised classification settings imply that SSRDM is significantly better than self-training. This can be attributed to the SSRDM's effective and discriminative weighting scheme that discovers the most important unlabeled data points for classification. Compared to our SSRDM, the performance improvement achieved by the self-training algorithm is small and can sometimes deteriorate classification accuracy.

6. Conclusion

In this paper we have introduced a novel semi-supervised discriminative method for learning mixtures of generative BNC models. Under semi-supervised settings, we utilized the minimum entropy principle leading to stage-wise data weight distributions for both labeled and unlabeled data. Unlike traditional approaches to discriminative learning, the proposed recursive algorithm is computationally as efficient as learning a single BNC model while achieving significant improvement in classification performance. Our recursive mixture learning is amenable to a pre-determined mixture order as well as robust to the choice of initial parameters.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgments

This study was supported by Seoul National University of Science & Technology.

References

[1] S. Ko, D. W. Kim, and B. Y. Kang, "A matrix-based genetic algorithm for structure learning of Bayesian networks," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 11, no. 3, pp. 135-142, Sep. 2011. http://dx.doi.org/10.5391/IJFIS.2011.11.3.135

[2] H. C. Cho, M. S. Fadali, and K. S. Lee, "Online parameter estimation and convergence property of dynamic Bayesian networks," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 7, no. 4, pp. 285-294, Dec. 2007. http://dx.doi.org/10.5391/IJFIS.2007.7.4.285

[3] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 131-163, 1997.

[4] T. Starner and A. Pentland, "Real-time American sign language recognition from video using hidden Markov models," in *Proceedings of 1995 International Symposium on Computer Vision*, Coral Gables, FL, 1995, pp. 265-270. http://dx.doi.org/10.1109/ISCV.1995.477012

[5] A. D. Wilson and A. F. Bobick, "Parametric hidden Markov models for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 884-900, Sep. 1999. http://dx.doi.org/10.1109/34.790429

[6] P. C. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 25-47, Jan. 2002. http://dx.doi.org/10.1006/csla.2001.0182

[7] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Wisconsin, 2003, pp. 375-381.

[8] S. Y. Lee and K. J. Lee, "Pattern classification model design and performance comparison for data mining of time series data," *Journal of Korean Institute of Intelligent Systems*, vol. 21, no. 6, pp. 730-736, Dec. 2011. http://dx.doi.org/10.5391/JKIIS.2011.21.6.730

[9] Y. K. Bang and C. H. Lee, "Design of fuzzy system with hierarchical classifying structures and its application to time series prediction," *Journal of Korean Institute of Intelligent Systems*, vol. 19, no. 5, pp. 595-602, Oct. 2009. http://dx.doi.org/10.5391/JKIIS.2009.19.5.595

[10] R. Greiner and W. Zhou, "Structural extension to logistic regression: discriminative parameter learning of belief net classifiers," in *Proceeding 18th National Conference on Artificial Intelligence*, Edmonton, AB, 2002, pp. 167-173.

[11] F. Pernkopf and J. Bilmes, "Discriminative versus generative parameter and structure learning of Bayesian Network Classifiers," in ref*Proceedings of the 22nd International Conference on Machine Learning*, Bonn, 2005, pp. 657-664.

[12] J. Salojarvi, K. Puolamaki, and S. Kaski, "On discriminative joint density modeling," in *Proceedings of the 16th European Conference on Machine Learning*, Berlin, 2005, pp. 341-352.

[13] Q. N. Dinh and C. H. Lee, "Model-based clustering of DOA data using von mises mixture model for sound source localization," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 13, no. 1, pp. 59-66, Mar. 2013. http://dx.doi.org/10.5391/IJFIS.2013.13.1.59

[14] J. Lee, S. Cho, J. Kim, and S.-T. Chung, "Layered object detection using adaptive gaussian mixture model in the complex and dynamic environment," *Journal of Korean Institute of Intelligent Systems*, vol.18, no. 3, pp. 387-391, Jun. 2008. http://dx.doi.org/10.5391/JKIIS.2008.18.3.387

[15] S. S. Kim, K. C. Kwak, J. W. Ryu, and M. G. Chun, "A Neuro-Fuzzy Modeling using the Hierarchical Clustering and Gaussian Mixture Model," *Journal of Korean Institute of Intelligent Systems*, vol. 13, no. 5, pp. 512-519, Oct. 2003. http://dx.doi.org/10.5391/JKIIS.2003.13.5.512

[16] M. Kim and V. Pavlovic, "Recursive method for discriminative mixture learning," in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007, pp. 409-416. http://dx.doi.org/10.1145/1273496.1273548

[17] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189-1232, Oct. 1999. http://dx.doi.org/10.1214/aos/1013203451

[18] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proceeding of Advances in Neural Information Processing Systems*, Vancouver, BC, 2004.

[19] A. Nadas, "A decision theorectic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, no. 4, pp. 814-817, Aug. 1983. http://dx.doi.org/10.1109/TASSP.1983.1164173

[20] V. Pavlovic, "Model-based motion clustering using boosted mixture modeling," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, 2004, pp. 811-818.

[21] Y. Jing, V. Pavlovic, and J. M. Rehg, "Efficient discriminative learning of Bayesian network classifier via boosted augmented naive Bayes," in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, 2005, pp. 369-376. http://dx.doi.org/10.1145/1102351.1102398

[22] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," in *Proceedings of the 2nd European Conference*, Barcelona, 1995, pp. 23-37. http://dx.doi.org/10.1007/3-540-59119-2_166

[23] T. Jaakkola, M. Diekhans, and D. Haussler, "Using the Fisher kernel method to detect remote protein homologies," in *Proceedings of the 7th International Conference on*

Intelligent Systems for Molecular Biology, Heidelberg, 1999, pp. 149-158.

[24] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43-49, Feb. 1978. http://dx.doi.org/10.1109/TASSP.1978.1163055

[25] C. A. Ratanamahatana and E. Keogh, "Making timeseries classification more accurate using learned constraints," in *Proceedings of the 4th SIAM International Conference on Data Mining*, Lake Buena Vista, FL, 2004, pp. 11-21.

[26] A. Veeraraghavan, R. Chellappa, and A. K. Roy-Chowdhury, "The function space of an activity," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, 2006, pp. 959-968. http://dx.doi.org/10.1109/CVPR.2006.304

[27] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265-292, Dec. 2001.

[28] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*, Denver, CO, 1997, pp. 507-513.

[29] K. B. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Proceedings of the 6th International Conference on Multiple Classifier Systems*, Seaside, CA, 2005, pp. 278-285. http://dx.doi.org/10.1007/11494683_28

[30] E. Keogh and T. Folias, "The UCR time series data mining archive," Department Computer Science & Engineering, University of California, Riverside CA, 2002.

[31] S. Hettich and S. D. Bay, "The UCI KDD archive," Department of Information and Computer Science, University of California, Irvine, CA, 2009.

[32] R. Tanawongsuwan and A. F. Bobick, "Characteristics of time-distance gait parameters across speeds," Available https://smartech.gatech.edu/bitstream/handle/1853/85/03-01.pdf?sequence=1

[33] R. Tanawongsuwan and A. Bobick, "Performance analysis of time-distance gait parameters under different speeds," in *Proceedings of the 4th International Conference on Audio- and Video-Based Biometric Person Authentication*, Guildford, 2003, pp. 715-724. http://dx.doi.org/10.1007/3-540-44887-X_83

[34] P. Saisan,G. Doretto,Y. N. Wu, and S. Soatto, "Dynamic texture recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, HI, 2001, pp. 58-63.

[35] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91109, Feb. 2003. http://dx.doi.org/10.1023/A:1021669406132

[36] A. B. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005, pp. 846-851. http://dx.doi.org/10.1109/CVPR.2005.279

[37] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proceedings of 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Beijing, 2005, pp. 65-72. http://dx.doi.org/10.1109/VSPETS.2005.1570899

Minyoung Kim received his BS and MS degrees both in Computer Science and Engineering from Seoul National University, South Korea. He earned a PhD degree in Computer Science from Rutgers University in 2008. From 2009 to 2010 he was a postdoctoral researcher at the Robotics Institute of Carnegie Mellon University. He is currently an Assistant Professor in

the Department of Electronics and IT Media Engineering at Seoul National University of Science and Technology in Korea. His primary research interest is machine learning and computer vision. His research focus includes graphical models, motion estimation/tracking, discriminative models/learning, kernel methods, and dimensionality reduction.