MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Finding Temporal Patterns by Data Decomposition

David C. Minnen and Christopher R. Wren

TR2004-054 June 2004

Abstract

We present a new unsupervised learning technique for the discovery of temporal clusters in large data sets. Our method performs hierarchical decomposition of the data to find structure at many levels of detail and to reduce the overall computational cost of pattern discovery. We present a comparison to related methods on synthetic data sets and on real gestural and pedestrian flow data.

Appears in the 6^{th} *International Conference on Automatic Face andGesture Recognition*

MERL Note 2003-77.
Patent application filed May 12.
Appeared in the 6^{th} International Conference on Automatic Face and Gesture Recognition, Seoul, Korea, May 17–19, 2004
MERL Note 2003-77 becomes MERL Technical Report 2004-054

1 Introduction

The occupants of a building generate patterns as they move from place to place, stand at a corner talking, or loiter by the coffee machine. These patterns leave their mark on every object in a building. Even something as lowly as a carpet will eventually come to represent some of these patterns by how it wears. However, our automated systems are largely blind to these patterns: elevator, heating and cooling, lighting, information, safety, and security systems all depend on humans to translate these patterns into policies and action.

Such patterns are influenced by the configuration of the environment, the individuals occupying it, and possibly even external context such as the time and date. The set of patterns observed by any particular system will certainly be unique, and possibly even non-stationary. It is necessary for any automatic interpretation of them to adapt to the local structure of the data. Embedded systems will need to efficiently discover the patterns of behavior that are interesting in the particular host context.

Temporal patterns created by pedestrians moving through a building represent a form of gestural expression that exists at a scale larger than a single individual. As such, the algorithmic machinery needed to learn to recognize particular patterns of pedestrian flow is not fundamentally different than that needed for many traditional gesture recognition applications. We present experimental results that demonstrate this point.

We use hidden Markov models (HMMs) to represent the motion patterns. HMMs have been used successfully in many previous research efforts to model sequences representing pedestrian flow [3], complex hand gestures [8, 10], DNA sequences [2], and human speech [6]. Here, we use HMMs in two capacities. First, as a probabilistic generative model that can represent and distinguish between different motion patterns, and second as a means of clustering temporal sequences in order to learn distinct underlying processes. In these respects, we agree with recent literature[7, 1] that mixtures of Hidden Markov Models represent a powerful model for discovering temporal patterns in human motion data, gestural and otherwise. However the clustering literature has so far taken a very classical approach to statistical modeling that discounts computational concerns. If we hope to utilize temporal clustering to create adaptive, embedded systems, then we need to find methods for discovering temporal patterns that are computationally lightweight.

We present a framework that hierarchically decomposes a training set into a number of classes, each representing a single motion cluster modeled by an individual HMM. This framework involves making a series of binary decompositions of the data during the learning process that results in the discovery of temporal clusters at the leaves of the resulting tree. This approach allows us to discover large numbers of clusters in large datasets without the need to evaluate every sequence with every leaf model. Our method has recognition performance that is similar to the method proposed by Smyth[7] but is significantly more efficient. Our shared dependence on hard clustering means

that, like Smyth's method, our modeling performance may not be as good as that reported by Alon, et. al.[1] when learning to separate highly similar clusters. However, our computational efficiency provides a means to trade learning time for fidelity when choosing a learning engine.

The algorithm described here, and much of the work cited above, focuses on the task of unsupervised discovery of structure in large databases of arbitrary motion data. However, it is interesting to note that it is also useful to consider the task that we will call meta-supervised learning: the case where an oracle may provide labels that group data into some high-level, semantically interesting group that may nevertheless consist of a great diversity of perceptually distinct gestures. An example of one such high-level event is people approaching the elevator and pressing the call button. This the end of this activity is exactly labeled by the elevator call button press. However, this same label will be applied to all the gestures that terminate in a call button press, regardless of how the person approached the elevator or where they originated in the building. If a person may approach an elevator from the north or the south, it would be difficult to learn a single, accurate model of "person approaching elevator" if no attempt is made to model these distinct subprocesses. It is therefore important to perform unsupervised learning within each semantically labeled data set to expose the perceptually distinct gestures that comprise the high-level class.

We will discuss the details of the learning algorithm in the next section. Then we will show experimental results on both synthetic data and a variety of real data sources in Section 3.

2 Learning by Data Decomposition

The framework consists of a tree of nodes. Each of the nodes contains a composite HMM with a small number of independent paths. Each path through the HMM carries a one-to-one association with a node in the next level of the tree. In this paper we will present results for 2-path nodes and, therefore, binary trees. Learning proceeds from root to leaf. Each node HMM is learned from the data, and then the data is decomposed into subsets: with each sequence assigned to only a single subtree, specifically the subtree associated with the maximum likelihood path label. At the leaves of the tree, the N sequences will be associated with the M models. However, only a single, low-complexity, low-path-count model was trained on the entire dataset of N sequences. The M leaf models, which may be significantly more complex than the root node models, are only trained on a small subset of the sequences.

For the HMMs that comprise the hierarchy, $P(O_i|\lambda_j^k)$ is the probability that the i^{th} sequence of observations O_i was generated by the j^{th} model on the k^{th} level of the hierarchy, which is parameterized by λ_j^k. The model λ consist of the standard HMM parameterization $\{\pi_m, T_{pq}, b_m\}$, where π are the prior probabilities for the states, T_{pq} is the matrix of state transition probabilities, and b is the parameterization of the output distribution. We will present

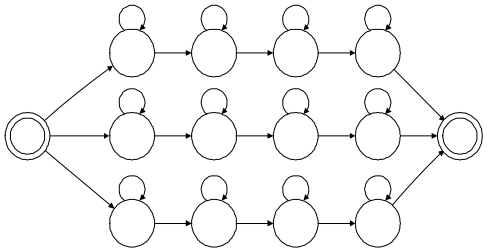

Figure 1: A composite HMM consisting of three path HMMs. The start and end state are non-emitting.

results in Section 3 utilizing both unimodal Gaussian distributions over continuous data, and multinomial distributions over discreet data.

Section 2.1 presents the Smyth-style clustering, and Section refsec:trees presents our modifications.

2.1 Composite HMMs

For the composite HMM illustrated in Figure 1, the transition matrix, T_{pq}, would be block diagonal, with each of the three paths contributing a block of non-zero transition probabilities between intra-path states, while the inter-path transition probabilities would all be zero. This is the composite HMM structure proposed by Smyth [7].

To train such a model, we first estimate M parallel HMMs via an agglomerative, temporal clustering algorithm, after which we construct the composite model, and then retrain using the Baum-Welch algorithm to allow soft assignments of examples to HMM paths. Clustering is complicated by the lack of a natural distance metric between temporal patterns or between the parameters that define the models. Smyth proposes using a derived distance metric where a different HMM is trained for each training observation, and the distance between two observations, O_i and O_j is then given by

$$D(O_i, O_j) = \frac{1}{2}[p(O_j \mid \lambda_i) + p(O_i \mid \lambda_j)] \tag{1}$$

where $p(O_j \mid \lambda_i)$ is the probability of generating the j^{th} observation from the i^{th} model. Intuitively, if the two observations are similar, then they sould give rise to similar models which would then generate the opposite observation with high likelihood. A similar method was used by Wang et. al. [10] but they proposed a more complicated distance metric:

$$\begin{aligned} D(O_i, O_j) = \frac{1}{2}[\quad & \tfrac{1}{T_i}(P(O_i \mid \lambda_i) - P(O_i \mid \lambda_j)) + \\ & \tfrac{1}{T_j}(P(O_j \mid \lambda_j) - P(O_j \mid \lambda_i))] \end{aligned} \tag{2}$$

Empirical tests, however, showed that this more complicated distance metric did not perform as well as Smyth's simpler metric.

Given this definition of a distance metric, agglomerative clustering of temporal sequences proceeds as follows:

1. Train an HMM for each of the N observations
2. Assign each observation to its own cluster
3. Compute the triangular matrix of distances between all pairs of clusters
4. Find the two closest clusters, $A, B : A \neq B$
5. Merge these two clusters
   (a) Update the distance matrix: $D(i, A) = max(D(i, A), D(i, B))$, for all clusters i
   (b) Remove the B^{th} row and column from the distance matrix
   (c) Record that these two clusters were merged at this step to form the hierarchy
6. If more than one cluster remains, go to step 4

This procedure will create a full range of decompositions starting at N nodes for N sequences, and ending at a single node model. To determine how many clusters should be used, a separate analysis can be applied to this data structure to find a "natural break" in the clustering (i.e., a merge between significantly more distant clusters than the previous merge). Alternately, a predefined number of clusters can be selected. In the experiments described below, the number of desired clusters was chosen empirically by comparing generalization performance between composite HMMs with different numbers of paths. Note also that, as defined above, the distance between two models will *increase* as they become more similar. Thus, we use the negative log-likelihood in place of the probabilities shown in Equation 1.

Once the temporal clustering is finished, the M selected clusters are used as the dataset to train new HMMs. Each of these HMMs represents a different underlying process as identified by the clustering algorithm. However, these HMMs are trained after "hard" assignment, which means that each training observation is a member of exactly one cluster. It has been shown that in many cases "soft" assignment leads to better models due to similarity and ambiguity between classes [1]. To accomplish such probabilistic training, a composite HMM is constructed and then retrained on all of the data.

Constructing the composite HMM is straightforward. Each cluster HMM becomes a distinct path through the composite HMM. This means that if each of the M cluster HMMs has s states, then the composite HMM will have $S = M \times s$ states, leading to a $S \times S$ transition matrix. The transition matrices of each path HMM are copied directly into the transition matrix of the composite HMM along the main diagonal. The prior state probabilities and final

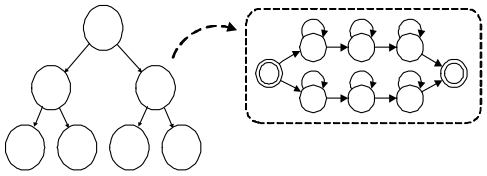

Figure 2: A tree of HMMs with three levels. Each node is composed of two HMMs that determine what subset of the incoming data is used for each child node.

transition probabilities (i.e., the probability of exiting each path HMM) are then copied into the composite HMM and normalized. Finally, the observation distributions from the path HMMs are also copied into the corresponding states of the composite model. Once the composite model has been constructed, the standard Baum-Welch algorithm can be used to train the paths with soft observation assignments.

Although this algorithm is straightforward, it can be inefficient to retrain the composite model. The standard Baum-Welch algorithm uses time on the order of $O(N \cdot S^2)$ per iteration, where the model has S states and is trained on N observations. Even though most of the state transitions are known to be zero, they still figure in to the calculation. Thus, training a composite model composed of M paths will take at least

$$\frac{N \cdot S^2}{N \cdot M \cdot s^2} = \frac{N \cdot (M \cdot s)^2}{N \cdot M \cdot s^2} = M \tag{3}$$

more time than training the path HMMs individually. Furthermore, the composite model will almost certainly require many more iterations to converge due to the extra parameters to estimate, which will further increase the training time. Alon et. al. have developed a more efficient algorithm for training with soft assignments [1] that is similar to the Baum-Welch algorithm except that the probability of membership of each observation to each path is taken to be a hidden variable along with the typical hidden state memberships.

2.2 The Tree of Composite HMMs

Two factors motivate the use of trees of HMMs to automatically learn temporal clusters. First, compared to the composite model described above, a tree of HMMs will require less time to train. Second, the tree can potentially decompose the data more sensibly since each level need only split the dataset into two parts rather than M separate clusters. For datasets that exhibit natural divisions that match this model, we would expect to see an improvement in modeling performance as well as a gain in efficiency.

A tree of HMMs is defined here as a binary tree where each node contains two HMMs (see Figure 2). The purpose of the two HMMs is to model different temporal clusters or groups of clusters in the dataset, thus dividing the data

into two distinct parts. Each of these parts is then sent to the child node corresponding to the HMM that better models it. This continued bifurcation of the dataset becomes fully decomposed at the leaves of the tree. It is important to note that the tree of HMMs is *not* a binary decision tree. Instead, it is a hierarchical model of all of the data. Thus, the leaf nodes do not represent decisions, rather they form the final step in the decomposition of the data into distinct clusters.

Each node in the tree will use time proportional to $O(ns^2)$, where each HMM has s states and is trained on n examples. Furthermore, the number of iterations needed for training to converge in each node is far lower than for the composite model, which also contributes to faster training.

One important question about the tree model is how the tree as a whole assigns a probability (i.e., $p(O \mid \lambda)$), where λ represents the tree model, to a particular observation. Three possibilities were tested empirically:

1. $p(O \mid \lambda)$ = probability of best leaf
   Only the leaf nodes matter in this model. Every HMM in a leaf node is evaluated for the given observation and the highest probability is chosen as the probability assigned by the tree.

2. $p(O \mid \lambda)$ = probability of best path node
   In this model, the tree of HMMs is treated like a decision tree in that are selected by recursively traversing down the tree from the root by seeing which of the two HMMs in each node better models the given observation. In this mode, the model can be seen as a hierarchical decomposition within which the appropriate scale for the observation is unknown.

3. $p(O \mid \lambda)$ = probability of best path leaf
   This model is similar to the previous model in that the tree of HMMs is treated as a decision tree. Here, however, the probability of the observation given the tree is taken to be the probability assigned by the leaf HMM.

Extensive testing indicates that, at least for our datasets, the leaf models generalize to novel data better than the internal decision models.

3 Experiments

In this section we relate the results of several experiments. In the first experiment we replicate the synthetic, ergodic HMM discrimination task from [1]. We then demonstrate cluster discovery on real data from a handwriting gesture task. Finally, we present results from the motivating task of discovering gestural structure in a large dataset of people moving around in an office building.

$\frac{\delta\mu}{\sigma}$	1.50	1.75	2.00	2.25	2.50
Alon	0	8	50	46	50
Smyth	0	0	50	50	50
Trees	0	0	50	50	50

Table 1: The results of the ergodic HMM experiment showing the number of correct trials, out of 50, given a relative separation between the means.

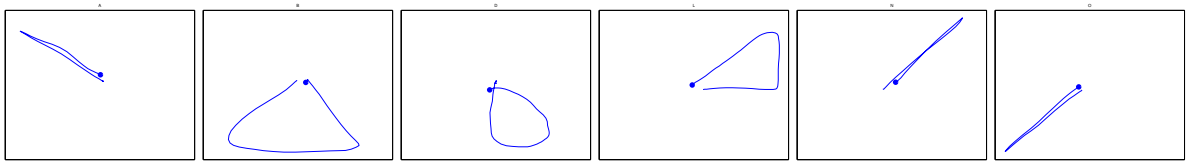

Figure 3: Example letter gestures from the Quickwriting data set.

3.1 Ergodic Process Discrimination

The goal of this experiment is to elicit differences between learning algorithms as the similarity between clusters increases. Fifty observation sequences of length 200 are generated from each of two known HMMs. Each HMM has two states, and the corresponding states between the models have the same Gaussian distributions but different transition matrices. The independent variable in the experiment is mu_2, the mean of the second state of each HMM. As mu_2 approaches the mean of the first state, mu_1, the clusters become more ambiguous. Table 3.1 shows the number of times out of 50 trials that the system correctly assigned at least 90% of the observation sequences to the correct cluster for each value of mu_2. Since a degenerate tree of only 2 clusters is equivalent to the 2-path composite HMM produced by Smyth, this experiment primarily serves to validate our implementations in that our results agree with [1].

3.2 Quickwriting Data

In this section we present results on real data collected in a pen gesture context. We collected data from a small number of subjects writing words with a pen interface utilizing Perlin's Quickwriting method [5]. Despite the fact that Quickwriting was designed to be easily recognized, we felt that it was a good evaluation dataset for this paper because of its balance between complexity, in terms of the number of clusters, and the ease of training study participants to perform the gestures. The complexity of of the gestures places it between handwriting recognition and multiphase hand gestures in complexity. See Figure 3 for examples of the individual gestures.

The dataset consists of 5 examples each of 5 distinct words from 5 sub-

	Accuracy	Speed
Smyth	68.0% $\pm$ 0.07%	98.2s
Alon	68.6% $\pm$ 0.09%	41.4s
Tree	68.9% $\pm$ 0.20%	31.6s

Table 2: The results of the letter gesture experiment, averaged over 25 runs.

jects. We extracted the 13 letters that comprise the dataset and randomly selected 30 examples of each letter. Of the 390 examples, 20% of each letter were randomly selected as a test set. The remaining 80% were used to train composite HMMs with the Smyth, Alon, and Tree algorithms. The Smyth and Alon algorithms were given the correct number of clusters. The tree algorithm was told to build 4 levels, or 16 models, so there were three spare classes in the Tree model.

Despite this handicap, the Tree model achieved the same classification performance as the other algorithms. We believe the high error rate maybe attributable to a few very similar gestures in the dataset, for example see 'B' (2^{nd} from left) and 'D' (3^{rd} from left) in Figure 3.

Furthermore, the Tree model trained in a third of the time that it took the Smyth method to converge, and slightly faster than the Alon method. We will see below that as the models increase in complexity and the data set increases in size, the computational performance advantage of the tree framework becomes more pronounced.

3.3 Pedestrian Flow Data

Finally we present the original motivating domain for this work: detecting macroscopic gestures generated by people purposefully moving about in an office building. Significant attention has been paid to manual gestures and facial motions, but only recently have computational and perceptual mechanisms made it possible to begin to study the macroscopic behaviors that people exhibit as they move about large spaces, such as office buildings. Because these gestures are not overtly communicative, they do not fit into the standard conceptual frameworks that people rely on when modeling manual gestures[4]. It is therefore necessary to consider data-driven models of these gestures, to expose the temporal patterns hidden in the data. Our approach is uniquely suited to discovering these temporal patterns in the large data sets that are required to capture the diverse behavior observed in these spaces.

The data is comprised of readings taken from 17 virtual motion detectors that observe a $175m^2$ contiguous section of an office building. The motion detectors are simulated using cameras running background subtraction and primitive motion detection algorithms[9]. Each raw observation vector consists of 17 binary values, with each binary value representing the presence or absence of motion in the view of one sensor. The sensors report their state

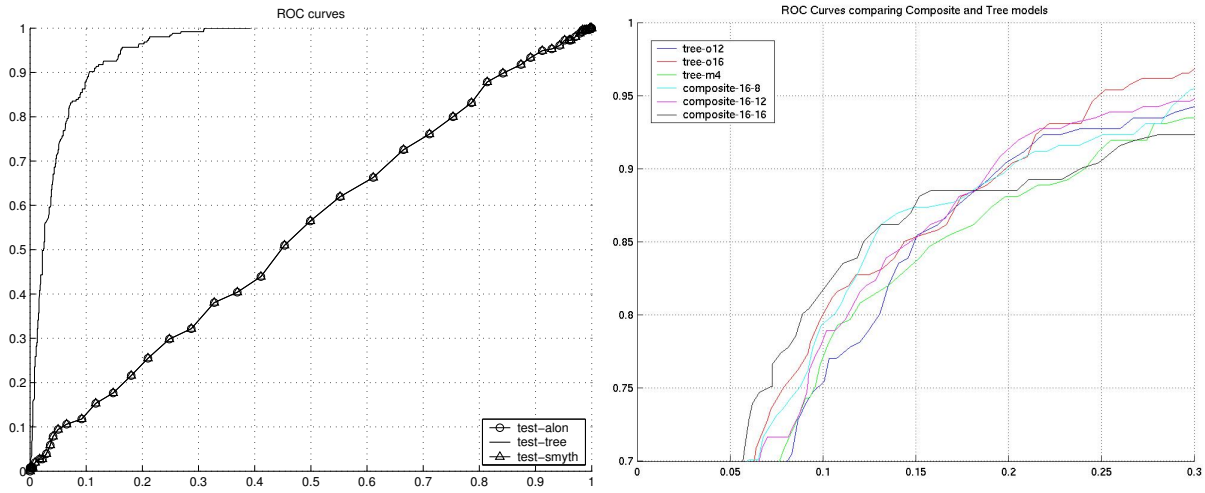

Figure 4: **Left:** Receiver Operating Characteristic (ROC) curves showing the classification performance of the Tree, Smyth, and Alon algorithms on the pedestrian flow data. **Right:** ROC curves comparing Tree and Smyth algorithms on a similar task.

7.5 times per second. The dataset, collected over 3 weeks, therefore potentially consists of 13.5 million observations, however only about 20%, or 2.7 million observations, include at least one positive sensor reading. In the results below, we only consider these 20% of observations that include at least one moving person in the area.

These raw 17-dimensional binary values (with 2^{17} possible states) were spatially clustered and then labeled with one of the 300 learned cluster labels. These spatial symbols together make up the range of the 300-element multinomial distribution used as the observation probability distribution in the HMM models for this data.

The task in this context is the meta-supervised situation described above. Given a subset of the training data labeled as, for example, "person approaching the elevator call button", the task is to build a classifier that correctly detects new examples. We accomplish this by building two trees: one to discover and model structure in the positive class and one to discover and model structure in the negative class. The outputs of the two trees are compared in a likelihood ratio test framework.

The left side of Figure 4 shows the performance results for the classifiers on the pedestrian flow data. The Tree algorithm trained both models, positive and negative, in 1783s. The Alon algorithm required 4152s to train both models, and the Smyth method completed after 50,356s. The tree and Smyth methods produce models with near-identical classification performance. This is because agglomerative clustering followed by Baum-Welch re-estimation in the Smyth case is very similar to agglomerative clustering followed by the

more efficient Alon-style re-estimation.

The right side of Figure 4 shows results form a similar data set where Smyth and Trees produced similar classification performance. In this case the training time for the Trees model was 660s, and the training time for the Smyth models was 7440s. We expect that in general the two kinds of models should have comparable classification performance. We have witnessed occasional failures with the Smyth and Alon methods of the kind demonstrated in the left side of Figure 4, but we do not yet have a good understanding of why these failures occur. In cases where the data happens to have a natural hierarchy that matches the structure of our model, we would expect Trees to have superior performance, while in cases where that hierarchical structure does not well match the data, we might expect Trees to take a performance hit relative to the Alon approach. However the drastic difference in performance shown in the left plot is hard to understand, and seems to depend on some characteristic of the dataset used for training.

4 Discussion

We have presented a framework for efficiently discovering the hidden temporal structures in large datasets. The frameworks provides efficiency on large, complex datasets by hierarchically decomposing the data into smaller, self-similar chunks. Furthermore this gain in efficiency is attainable without sacrificing a significant classification performance. We have shown the classification performance and computational efficiency with two datasets: a manual gesture data set, and a very large collection of pedestrian flow data.

References

[1] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 375–381, 2003.

[2] Darya Chudova and Padhraic Smyth. Sequential pattern discovery under a markov assumption. Technical Report 02-08, Information and Computer Science Dept., University of California, Irvine, 2002.

[3] Dimitrios Makris and Tim Ellis. Automatic learning of an activity-based semantic scene model. In *Proc. of IEEE Conference on Advanced Video and Signal Based Surveillance*, July 2003.

[4] David McNeill. *Hand and Mind: What Gestures Reveal about Thought*. The University of Chicago Press, 1992.

[5] Ken Perlin. Quikwriting: Continuous stylus-based text entry. In *User Interface Software and Technology*. ACM, 1998. Technical Sketch.

[6] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–285, 1989.

[7] Padhraic Smyth. Clustering sequences with hidden markov models. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 648. The MIT Press, 1997.

[8] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision*, Coral Gables, FL, USA, 1995. IEEE Computer Society Press.

[9] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV*, pages 255–261. IEEE, 1999.

[10] Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. Unsupervised analysis of human gestures. In *IEEE Pacific Rim Conference on Multimedia*, pages 174–181, 2001.