CONTEXT–BASED CLASSIFICATION VIA DATA–DEPENDENT MIXTURES OF LOGISTIC AND HIDDEN MARKOV MODEL CLASSIFIERS

By

SENIHA ESEN YUKSEL

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2011

To my wonderful mother whose strength never ceases to amaze me

ACKNOWLEDGMENTS

I would like to extend my heartiest gratitute to my advisor Dr. Paul Gader for his guidance and support during my PhD. Not only did he have the most interesting projects, a positive attitute, and a great lab; but also he hever left me without funding at the worst economic times. I feel very fortunate to have met him and be one of his students, and I hope to carry his professionalism and his excitement with me for the rest of my career.

I would like to thank my committe members Dr. Joseph Wilson, Dr. Anand Rangarajan, Dr. Jeffrey Ho and Dr. Haldun Aytug for their valuable suggestions and insightful comments. Dr. Ho has been very encouraging throughout my PhD studies. Dr. Wilson has been a major help in the first part of my dissertation. I would like to also thank him for diligently proof reading my papers.

I would like to also express my thanks to Dr. Rolf Hummel and Dr. Thierry Dubroca from the Material Science and Engineering Department. Our collaborative research at the last two semesters of my PhD was truly inspiring. I enjoyed having a major role in their explosive detection project, and doing interdisciplinary research.

Many thanks to Dr. Jeremy Bolton for emailing me one day to collaborate on a project on multiple instance learning. It was a semester well-spent working with him, that has given me many ideas for future research.

It was a pleasure to meet all the former and current PhD students at the CSI:Florida lab; Xuping Zhang, Ganesh Ramachandran, Andres Vasquez Mendez, Alina Zare, Ken Watford, Ryan Close, Brandon Smock, and Taylor Glenn. Also, I would like to thank all my friends at the CISE Department: Jeeyoung Kim, Hale Erten, Gokhan Kaya, Manu Sethi, Venkatakrishnan Ramaswamy, Karthik Gurumoorthy, Oneil Smith, Ritwik Kumar, and Ajit Rajwade. It has been a privilege to meet all these smart, fun and unique individuals.

I would like to thank all my teachers who have supported me to this day. Prof. Gonul Turhan Sayan and Prof. Gozde Bozdagi from my undergraduate studies, and my MSc advisor Prof Aly A. Farag always had confidence in me and supported me.

This PhD was a collaborative effort of all the people who supported me here and overseas. I would like to thank all the doctors and the nurses in Turkey who kept my father alive, gave me a piece of mind, and gave me a chance to show him my diploma.

The last but not the least, I would like to thank my family; my mother Prof. Oznur Yuksel, my father Arch. Huseyin Yuksel, and my soon to be surgeon brother Dr. Mehmet Eren Yuksel. You are my role models with your endurance and your strength. Nothing could have been possible without your support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

CONTEXT–BASED CLASSIFICATION VIA DATA–DEPENDENT MIXTURES OF LOGISTIC AND HIDDEN MARKOV MODEL CLASSIFIERS

By

Seniha Esen Yuksel

July 2011

Chair: Paul D. Gader
Major: Computer and Information Science and Engineering

This research addresses the problems encountered when designing classifiers for classes that contain multiple subclasses whose characteristics are dependent on the context. It is sometimes the case that when the appropriate context is chosen, classification is relatively easy, whereas in the absence of contextual information, classification may be difficult. Therefore, this research focuses on simultaneous learning of context and classification.

One area where such problems are encountered is landmine detection. The signals collected by radar and metal detector systems show a lot of variability depending on a number of factors. Some of these factors can be listed as follows: (i) landmines are diverse in terms of their physical properties (large, small, plastic, metal), (ii) they may be buried at various depths in the ground (shallow, deep), (iii) they may be covered with clutter (stones, bushes, bottles, etc.), and (iv) the signals collected from these mines at different environments show a huge variability based on temperature, humidity, and soil conditions. When all of these factors are combined, one mine may look more similar to a mine of a different type (or to a metallic clutter) than to itself at a different environment. Moreover, the exact reason that caused this confusion may not be always clear nor available. In such cases, a group of data that looks similar based on some unknown factor defines a context, and machine learning algorithms are needed to find

these contexts and to learn classifiers within each context for a mine versus non-mine decision.

Mixture of experts (ME) model is suitable for problems such as the landmine detection that require context-based classification. ME divides the data into multiple contexts and learns expert classifiers within each context all within one probabilistic framework. For landmine detection, the ME model significantly increased the classification rates while providing insightful parameters that can be understood by a system operator. However, two problems were encountered with the ME model. The first problem was over-fitting, which means that the model could be learning noise, and it may not generalize to future data. To prevent over-fitting, the variational mixture of experts model for multi-class classification was developed, and its lower bound was derived. The lower bound provides a measure of correctness as it has to be increasing, and is used as an excellent stopping criterion. The second problem was that the ME model is not suitable for time series or sequential data as it requires fixed-length features and it cannot represent the dependency between a sequence of observations. To overcome these issues, a novel model, mixture of hidden Markov model experts (MHMME) has been introduced. The MHMME model can successfully divide time series data into multiple contexts, and learn an expert hidden Markov model (HMM) for each of these contexts. MHMME is a general method that is not restricted to landmine detection. Our experiments on multiple real and synthetic datasets show that MHMME can perform better than ME and HMMs, and can do well in comparison to state-of-the-art models.

CHAPTER 1
INTRODUCTION

Many datasets in the real world show different patterns based on multiple contexts. For example, electricity usage has seasonal patterns, and within each season electricity consumption of socio-economic groups show different patterns. In this example, both the seasons and the socio-economic groups define the context for electricity consumption. In electrocardiogram (EKG) heartbeat classification, ethnic and social groups define a healthy beat. For example, athletes typically have slower resting heartbeats. In handwriting recognition, a single character may be impossible to read without the context of the word in which it appears.

The aforementioned problems are also encountered in landmine data so it is difficult to represent the class of all mines with a single model. Going one step deeper, one can think of the class of all mines to have multiple subclasses such as high metal anti-tank (HMAT) mines, low metal anti-tank (LMAT) mines, high metal anti-personnel (HMAP) mines, and low metal anti-personnel (LMAP) mines, however, the features of these subclasses are interlaced, so it is also difficult to appoint a model as an expert to identify a particular subclass of mines. Furthermore, the data might be available from multiple sources. For example, in landmine detection, there are two types of sensors that are widely used, ground penetrating radars (GPR) and metal detectors – also known as electromagnetic induction (EMI) sensors. While the HMAP and HMAT mines can appear similar to a metal detector, they may look quite different to a GPR sensor. Similarly, given an object with a low energy EMI signature, GPR will be better at distinguishing between the LMAP mines and clutter.

All of these arguments bring out the fact that for such datasets where there are multiple contexts, a classifier needs to have expert models for each context. Unfortunately, contexts are generally hard to define, they are often interlaced, and do not have sharp boundaries. Moreover, context information might be inherent in the

data, but not be known to the data modeler. Therefore, the classifier should be able to learn both the context and the experts, ideally in a single model. In cases where the context information is unknown, a context is defined as a group of similar signatures.

The mixture of experts (ME) model introduced by Jacobs and Jordan et al. [1, 2] is suitable for such problems as it solves non-linear problems by a probabilistic division of the input space. In the ME model, a gate makes soft partitions of the input space and defines those regions where the individual expert opinions are trustworthy. A simple example to demonstrate the division of the input space is shown in Fig. 1-1, where the gate simplifies a nonlinear classification problem into two linear classification problems, and the experts learn the simple parametrized surfaces in these regions. While the ME has been used to solve both regression and classification problems, the focus of this study is the classification version.

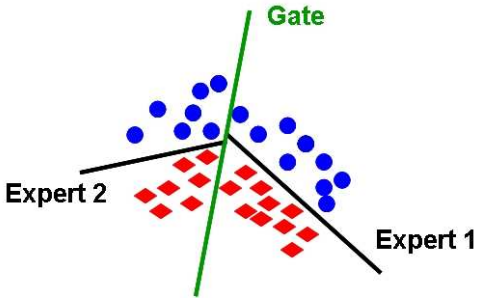


Figure 1-1. Simplified classification example for ME. The gate divides the region in two with a soft decision, then the experts learn the simple surfaces to separate the two classes. Modified with permission [3].

For complicated input examples, however, ME is prone to over-training due to the expectation-maximization training, especially if the number of experts are increased. To regularize the ME, variational mixture of experts (VME), was developed and used by several researchers [4–8] for regression problems. Waterhouse [7] discussed how to adjust the VME to classification, but the steps to train the model were not made clear since the equations were applicable to vector-valued parameters as opposed to

matrices for each expert. Also, the previous algorithms used the number of iterations as a stopping condition, which may lead to the premature termination of the algorithm or to unnecessary training.

In addition, ME model is not suitable for time series data, and the several extensions in the literature [9–14] concern regression and use a one-step-ahead or multi-step-ahead prediction. In multi-step prediction the last d values of the time-series data are used as features in a neural network, but such algorithms cannot handle data with varying length and the use of multilayer network type approaches prevent them from completely describing the temporal properties of time-series data.

1.1 Original Contributions

In the first part of this study, the ME model was used for landmine detection to choose an appropriate context and to learn multiple models for that context. The experts of the ME model give soft decision boundaries for different types of mines (HMAT, LMAT, etc.), and learn specialized models for each type. The ME model significantly increased the classification rates while providing insightful parameters that can be understood by a system operator.

A key problem in the ME model was over-fitting, which means that the model could be learning noise, and it may not generalize to future data. To prevent over-fitting, prior knowledge was incorporated to the system and a variational mixture of experts model for multi-class classification (VMEC) was developed. The VMEC model (i) regularizes the model parameters and resists over-fitting if the number of experts is high or if the training dataset is small, (ii) gives distributions of the parameters instead of point estimates, which is an important property for statistical inference, and (iii) increases the classification rates on the landmine dataset. Furthermore, the derivation of the lower bound of VMEC is presented. The lower bound can be used as an excellent stopping criterion as opposed to the number of iterations. It also provides a measure of

correctness as it has to be increasing. The efficacy of VMEC is presented on synthetic data as well as on landmine data in Chapter 4.

A second problem with the ME model is that it is not suitable for time series or sequential data as it requires fixed-length features and it cannot represent the dependency between observations. In this study, a novel mixture of hidden Markov model experts (MHMME) is developed that can both decompose time-series data into multiple contexts, and learn expert classifiers for each context. In MHMME, a gate of hidden Markov models cooperate with a set of hidden Markov model experts that provide multi-class classification. With this change of functions at the gate and the experts, the learning and testing algorithms are completely modified, but the essence of the ME model is still kept. In doing so, while the HMMs at the gate make soft partitions of all the data from all the classes, the HMMs at the experts look for models that would discriminate among these data. The main advantages of MHMME can be summarized as follows:

- MHMME model is suitable for time-series data and sequential data of varying lengths due to the use of the hidden Markov models.
- MHMME model provides a divide and conquer approach, is probabilistic, and has soft boundaries – all of which make it very suitable for context learning.
- Unlike the traditional mixture models where the mixture coefficient is a scalar, in the MHMME model, the mixture coefficient (i.e. the gate) depends on the input, which supports the context learning.
- The learning of the contexts and the classifiers is accomplished simultaneously, in one model. MHMME is suitable for multi–class classification.
- Experiments on synthetic and real data show that MHMME can perform better than ME and HMMs, and can do well in comparison to state-of-the-art models.

The update equations that lead to the simultaneous training of the experts and the gate are given in Chapter 6. Throughout this study, the following words, temporal, sequential, or time-series data, will be used interchangeably as we are not only interested in time-series data, but with any data that carry sequential information.

CHAPTER 2
DESCRIPTION OF DATASETS

2.1 Landmine Dataset

The landmine dataset consists of mine and non-mine object data collected using robotic systems with ground penetrating radar (GPR) and wide-band electromagnetic induction (WEMI) sensors. At present, there are between $60,000,000$ to $100,000,000$ active buried landmines around world [15], and the detection of these landmines is a hard problem because of the various sizes and different types of mines, as well as the changes in their signatures with changes in temperature, humidity, and soil conditions.

Landmines are mainly categorized into four groups according to their metallic content and according to their targets. These categories are high metal anti-tank (HMAT), high metal anti-personnel (HMAP), low metal anti-tank (LMAT), and low metal anti-personnel (LMAP). To create a controlled environment for algorithm development, clutter and mine objects were placed into grids in two temperate sites, and data was collected using the robotic systems. The properties of these temperate sites are given in Table 2-1 and Table 2-2. Table 2-1 gives the number of objects in each site. It can be observed that the first site has more clutter objects and fewer mine objects. The second site contains more mine objects and more mine types. In Table 2-2, the percentage of landmine and clutter objects is given.

Table 2-1. Landmine data collection from two temperate sites

Data/Site	Temperate Site 1	Temperate Site 2
Total number of data	224	220
Number of mine objects	44	112
Number of clutter objects	148	68
Number of blank grids	32	40
Number of mine types	12	26
Grid area	1.5m x 1.5m	1m x 1m

Table 2-2. Proportion of landmine and clutter objects

Notation	Meaning	Proportion
HMAP	High metal anti-personnel mine	6.7
LMAP	Low metal anti-personnel mine	14.8
HMAT	High metal anti-tank mine	2.4
LMAT	Low metal anti-tank mine	11
HMC	High metal clutter	20.2
MMC	Medium metal clutter	6.3
LMC	Low metal clutter	6.3
NMC	Non-metallic clutter or blank cell	32.1

To detect these landmines, robotic systems with GPR and WEMI sensors have been developed as shown in Fig. 2-1. The WEMI sensors are good at finding the mines with metallic content, whereas the GPR sensors are good at finding the plastic mines. A GPR sensor's operating frequency is between 200 MHz and 7 GHz, and it collects data with 2 cm intervals in 24 channels. WEMI sensor, also called the metal detector, collects complex response in 21 frequencies between 330Hz and $90,030$Hz spaced in log space, at 1 cm intervals in three channels. This results in a $21 \times 3 \times 150$ complex data matrix for each grid. Metal detector is completely described in [16, 17].

Signatures of data collected by GPR and WEMI are shown in Fig. 2-2. In general, landmines show hyperbolic patterns in the GPR data [18], and show smoother patterns in WEMI data when compared to non-metallic clutter. Anti-personnel mines are generally placed close to the ground surface, show some faint metallic signature, and display some GPR that can often be confused with surface clutter. Anti-tank mines are found at deeper levels, are bigger in size, so they yield better GPR signatures, but as they are found at greater depths, their EMI signals might be faint, especially for the ones with low metal content. These properties are summarized in Table 2-3.

In the data collection, adjoining lanes were divided into grids and the object was assumed to be at the center of the grid cell as shown in Fig. 2-3. The data were collected in two opposite directions over the same set of grid locations. The feature extraction algorithms were designed to give a score from the center of the grid.

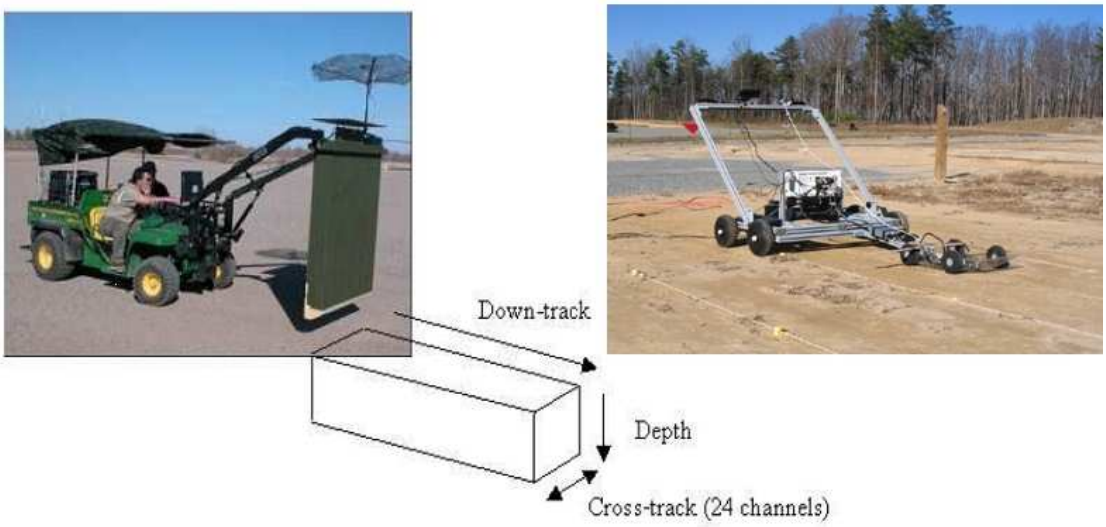


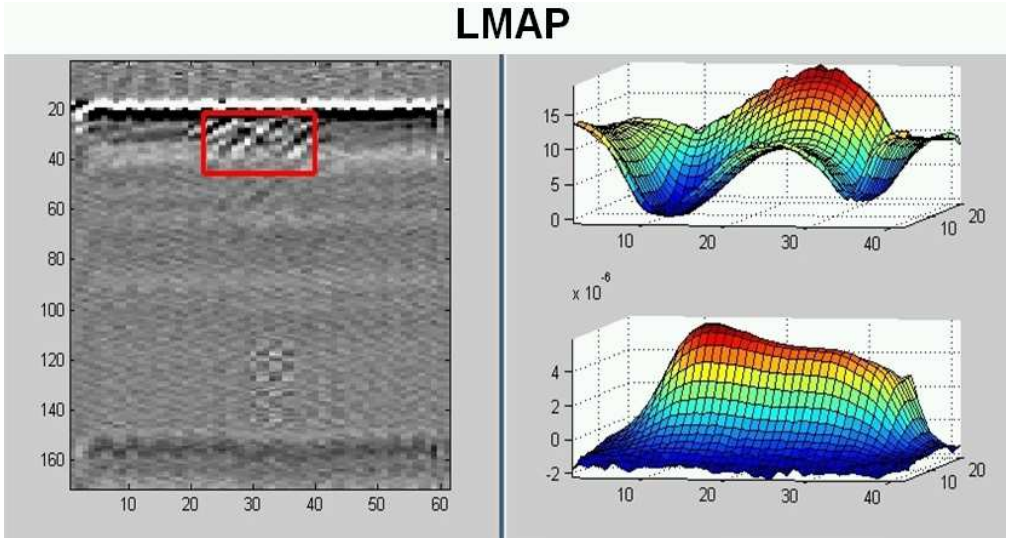
Figure 2-1. The robotic systems for landmine detection. The vehicle on the left has the GPR, while the cart on the right has the metal detector sensor. In the 3-D data collected from the GPR, the path of the moving vehicle is denoted as the down-track, and the channels in the GPR sensor are denoted as the cross-track.

Table 2-3. Properties of landmines

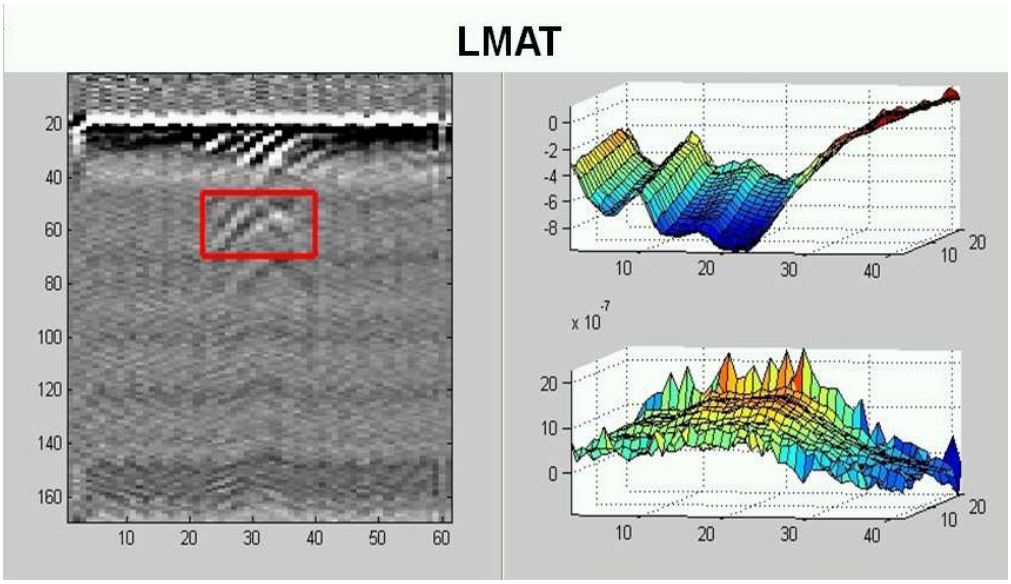
	Low Metal (LM)	High Metal (HM)
Anti-personnel (AP)	Shallow Some GPR Faint metal	Shallow Some GPR Good metal
Anti-tank (AT)	Deep Good GPR Faint metal	Deep Good GPR Good metal

In this study, the angle prototype matching (APM) [19] and gradient angle model (GRANMA) based K-nearest neighborhood (K-NN) [20] feature extraction algorithms were used for EMI data; and linear prediction processing (LPP) [18], spectral feature extraction (SF) [21] and edge histogram descriptor (EHD) [22, 23] methods were used for GPR data.

APM and GRANMA feature extractors use argand diagrams. An argand diagram is a plot of the in-phase component against the quadrature-phase component; and



A LMAP object's GPR data on the left, EMI data on the right



B LMAT object's GPR data on the left, EMI data on the right

Figure 2-2. Signatures collected from the GPR and WEMI sensors for an LMAP and an LMAT. In the GPR data, the x-axis shows the down-track direction, and the y-axis shows the depth. In the WEMI data, for both the real (top) and imaginary response (bottom), the x-axis shows the down-track direction, y-axis shows the energy, and z-axis shows the frequency.

the angles between the points in an argand diagram have been shown to provide discriminative features for different types of landmines [20]. The APM method finds the

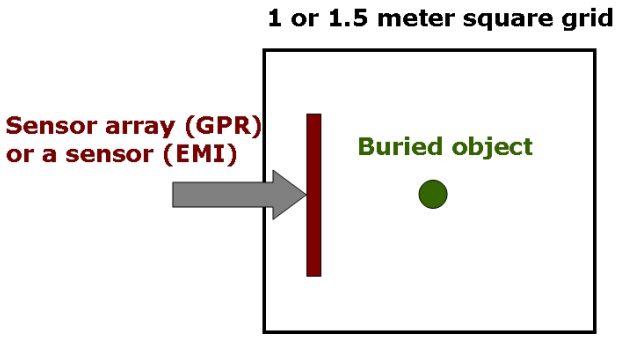


Figure 2-3. The lanes are divided into grid cells of width 1m and 1.5m. Each object is placed at the center of a grid cell.

angle sequences and uses a point-wise median (median among candidates at a given frequency) as the prototype and computes the Euclidean distance as a goodness of fit measure. The GRANMA method uses a parametric model to model the WEMI data, and the model parameters are used for classification in a K-NN framework.

The first of the GPR features, LPP, is the maximum energy value of the background removed data around an alarm location declared by a pre-screener. The background in the GPR is removed using a dynamic linear predictor, where the current background vector sample is approximated by a linear combination of past background vector samples. The linear prediction coefficients are obtained by minimizing the mean-square prediction error, and the LPP output is the difference between the current GPR vector sample and its predicted value generated from the linear prediction model.

The second GPR feature extractor is the spectral algorithm. The spectral feature is created by estimating the energy density spectrum (EDS) of an alarm at the declaration position and matching the estimated EDS with a template that is obtained from a landmine target. The LPP and spectral features were fused by the direct sum of the two (after proper scaling in LPP) divided by the square root of spectral compactness. This is called the SpectralLPP feature.

The third GPR feature, EHD, is a translation invariant feature based on the local edge distribution of the 3D GPR signatures [22, 23]. With EHD, a 3D signature is divided into windows. In each window, local edges are characterized into vertical, horizontal, diagonal, antidiagonal edges and nonedges. Then the local edge distribution for each window is represented by a histogram. Finally, each object is given a confidence value by comparing it to a number of prototypes using a possibilistic K-NN rule.

2.2 Chicken Pieces Dataset for 2D Shape Recognition

The chicken pieces database [24] consists of 446 binary images of chicken pieces from five classes, a sample of which is displayed in Fig. 2-4. The number of samples in each class is given in Table 2-4. This dataset is publicly available at http://algoval.essex.ac.uk/data/sequence/chicken/. Its curvature features were made available by Bicego et al., and fully exlained in [25]. Briefly, the curvature features were obtained as follows: the contours of the binary images were found by a Canny edge detector, and approximated by line segments. The initial point was set to the rightmost point lying on the horizontal line passing through the object centroid. Then the curvature value at each point was computed as the angle between two consecutive segments at that point and recorded in a counterclockwise manner as demonstrated in Fig. 2-5. The resulting sequences have different lengths as summarized in Table 2-5.

Table 2-4. Distribution of samples in the chicken pieces dataset

Wing	Back	Drumstick	Thigh and Back	Breast	Total
117	76	96	61	96	446

Table 2-5. Lengths of the sequences in the chicken pieces dataset

Min. length	Max. length	Mean length	Median length
18	104	54	51

Figure 2-4. A sample from the chicken pieces database. The rows represent the wing, back, drumstick, thigh and back, and breast classes, respectively.

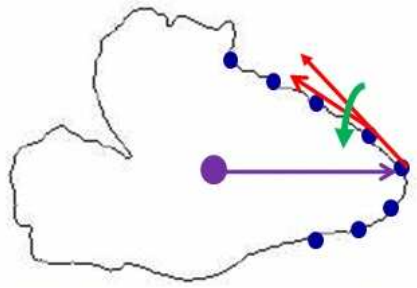


Figure 2-5. Curvature features. Contours of binary images were found by a Canny edge detector, and approximated by line segments. Curvature value at each point was computed as the angle between two consecutive segments, and was recorded in a counterclockwise manner. The initial point is the rightmost point lying on the horizontal line passing through the object centroid.

CHAPTER 3
MIXTURE OF EXPERTS (ME)

The original mixture of experts (ME) model introduced by Jacobs et al. [1] can be viewed as a tree structured architecture that is based on the principle of divide and conquer, and has three main components: several experts that are either regression functions or classifiers; a gate that makes soft partitions of the input space and defines the regions where the expert opinions are trustworthy; and a probabilistic model to combine the gate and the experts. The model is a weighted sum of the experts, where the weights are the input dependent gates. In this simplified form, the original ME model has three important properties, (i) it allows for the individual experts to specialize in smaller parts of a bigger problem, (ii) it uses soft partitions of the data, and (iii) it allows the splits to be formed along hyperplanes at arbitrary orientations in the input space [26]. These properties allow to represent non-stationary or piecewise continuous data in complex regression processes, and to identify the non-linearities in classification problems. Therefore, to understand the systems that produce such non-stationary or piecewise continuous data, ME has been revisited and revived over the years in many publications. In these studies, the key elements have been the three components of the original ME – the gate, the experts and a model to combine the gate and the experts. The linear gate and the experts of the original ME model have been improved with more complicated regression or classification functions, the learning algorithm has been changed, and the mixture model has been modified for density estimation and for time series data representation.

In the past twenty years, there have been solid statistical and experimental analyses on ME, and a considerable number of studies have been published in all areas of regression, classification and fusion. The ME models have been found useful in conjunction with many of the current classification or regression algorithms because

of their modular and flexible structure. In the late 2000's, several ME studies were published, including [12, 14, 27–52] in the last several years.

To see the benefit that ME models have provided, we can look at a 2008 survey that identified the top 10 most influential algorithms in the data mining area. It cited C4.5, k-Means, support vector machines (SVM), Apriori, expectation-maximization (EM), PageRank, AdaBoost, k-nearest neighborhood (kNN), naive Bayes, and classification and regression trees (CART) [53]. ME is closely related to most of these algorithms, and has been compared to, or combined with most of them to improve their performance. Specifically, MEs have often been trained with EM and have been initialized using k-Means [35, 54]. It has been found that decision trees have the potential advantage of computational scalability, handling data of mixed types, handling missing values, and dealing with irrelevant inputs [55]. However, decision trees carry the limitations of low prediction accuracy and high variance [56]. ME can be regarded as a statistical approach to decision tree modeling where the decisions are treated as hidden multinomial random variables. Therefore, ME carries the advantages of decision trees, but improves on them with its soft boundaries, lower variance, and its probabilistic framework to allow for inference procedures, measures of uncertainty, and Bayesian approaches [57]. On the other hand, decision trees have been combined in ensembles, forming random forests, to increase the performance of a single ensemble and increase prediction accuracy while keeping other decision tree advantages [58]. Similarly, ME has been combined with boosting and, with a gating function that is a function of confidence, ME has been shown to provide an effective dynamic combination for the outputs of the experts [59].

ME is flexible enough to be combined with a variety of different models. It has been combined with SVM [33, 34, 52] to partition the input space and to allocate different kernel functions for different input regions, which would not be possible with a single SVM. Recently, ME has been combined with Gaussian processes (GP) to make them

accommodate nonstationary covariance and noise. A single GP has a fixed covariance matrix, and its solution typically requires the inversion of a large matrix. With the mixture of GP experts model [27, 28], the computational complexity of inverting a large matrix can be replaced with several inversions of smaller matrices, providing the ability to solve larger scale datasets. ME has also been used to make a hidden Markov model (HMM) with time-varying transition probabilities that are conditioned on the input [14, 39].

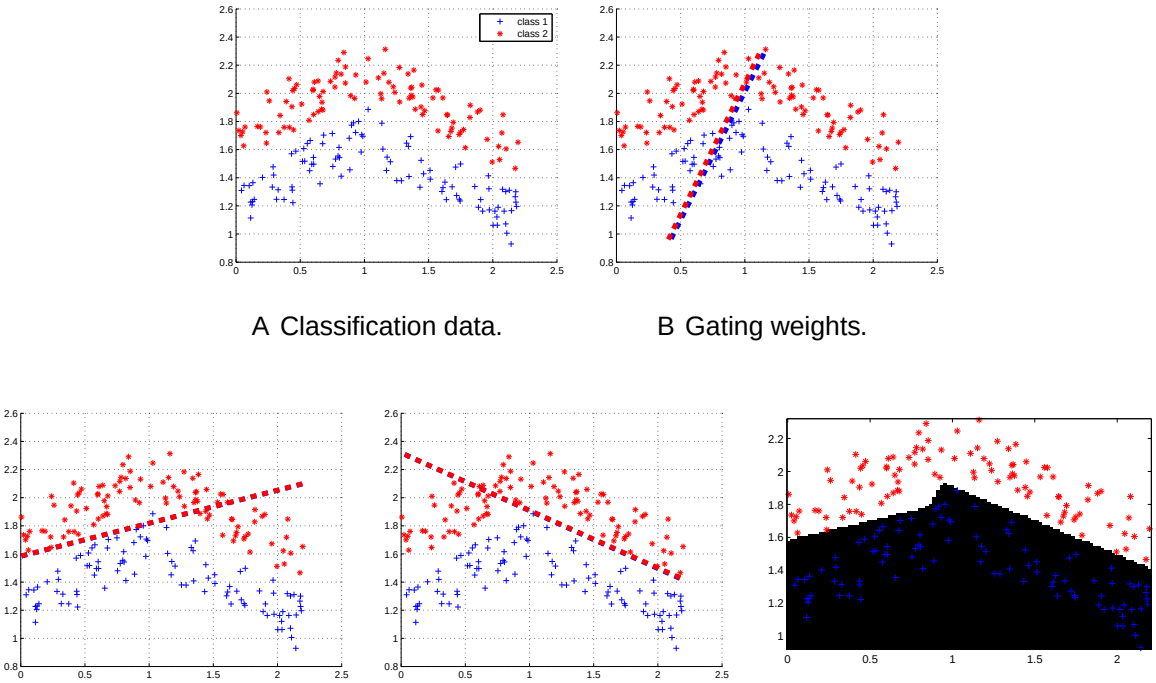
ME has been regarded as a mixture model for estimating conditional probability distributions, and with this interpretation, MEs statistical properties have been investigated during 1994 – 2006. The convergence properties [2], the bias and variance analysis [60], upper bounds on the approximation error [61], rate of approximation to the true mean in regression [62], bounds on the Vapnik-Chervonenkis dimension of ME [63], regularity conditions for consistency [64], identifiability of ME [65], function approximation properties [66], asymptotic normality conditions [67], consistency properties [68, 69] have been analyzed.

These statistical properties have led to the development of various Bayesian training methods between 1996 – 2010, and it has been possible to train ME with variational methods and Markov chain Monte Carlo (MCMC). The Bayesian training methods have introduced prior knowledge into the training, helped avoid over-training, and opened the search for the best model (the number of experts and the depth of the tree) during 1994 – 2007. In the meantime, the model has been used in a very wide range of applications from finance to biology.

3.1 ME Classification Model

In the ME model, a set of expert networks and a gating network cooperate with each other to solve a non-linear supervised learning problem by dividing the input space into a nested set of regions. The gating network makes a soft split of the whole input space, and the experts learn the simple parametrized surfaces in these partitions of the regions. The parameters of these surfaces in both the gate and the expert networks can

be learned using the EM algorithm. A simple example to demonstrate the division of the input space is shown in Fig. 3-1. In Fig. 3-1(A) the blue points from the first class and the red points from the second class are not linearly separable. However, once the data is partitioned into two by the gate shown in Fig. 3-1(B), the data on either side of the gate is linearly separable, and can be classified by linear experts as shown in Fig. 3-1(C) and Fig. 3-1(D). The parameters of the gate and the experts have been plotted as lines. The final decision has been explicitly shown in Fig. 3-1(E). The decisions of the gate and the experts are probabilistic, so these regions are actually soft partitions, and the final decision is made by taking the maximum of the probabilistic outputs.



A Classification data. B Gating weights.

C Expert 1 weights. D Expert 2 weights. E Classification results.

Figure 3-1. Classification results on a simple example. The blue and the red points in (A) belong to class 1 and class 2, respectively. The gate divides the region into two in (B), such that, to the left of the gating line, the first expert is responsible as shown in (C), and to the right of the gating line, the second expert is responsible as shown in (D). In (E), classification results are shown as areas, where the black region corresponds to class 1 (blue points) and the white region corresponds to class 2 (red points).

Let $D = \{X, Y\}$ denote the data where $X = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ is the input, $Y = \{\mathbf{y}^{(n)}\}_{n=1}^{N}$ is the target, and N is the number of training points. Also, let $\Theta = \{\Theta_g, \Theta_e\}$ denote the set of all parameters where Θ_g is set of the gate parameters and Θ_e is the set of the expert parameters. Unless necessary, we will denote an input vector $\mathbf{x}^{(n)}$ with $\mathbf{x}$, and a target vector $\mathbf{y}^{(n)}$ with $\mathbf{y}$ from now on. A superscript (n) will be used to indicate that a variable depends on an input $\mathbf{x}^{(n)}$.

In ME, it is assumed that the experts are mutually exclusive, i.e., each pattern belongs to one and only one expert [10]. Hence, given an input vector $\mathbf{x}$ and a target vector $\mathbf{y}$, the total probability of observing $\mathbf{y}$ can be written in terms of the experts:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}, \Theta) &= \sum_{i=1}^{I} P(\mathbf{y}, i|\mathbf{x}, \Theta) \\ &= \sum_{i=1}^{I} P(i|\mathbf{x}, \Theta_g) P(\mathbf{y}|i, \mathbf{x}, \Theta_e) \\ &= \sum_{i=1}^{I} g_i(\mathbf{x}, \Theta_g) P(\mathbf{y}|i, \mathbf{x}, \Theta_e) \end{aligned} \tag{3–1}$$

where I is the number of experts, the function $g_i(\mathbf{x}, \Theta_g) = P(i|\mathbf{x}, \Theta_g)$ representing the probability of the i^{th} expert given $\mathbf{x}$ is the gate, and $P(\mathbf{y}|i, \mathbf{x}, \Theta_e)$ is the probability of the i^{th} expert generating $\mathbf{y}$ given $\mathbf{x}$. The latter will be denoted by $P_i(\mathbf{y})$ from now on. The ME training algorithm maximizes the log-likelihood of the probability in Eq. 3–1 to learn the parameters of the experts and the gate.

The gate is the scalar defined by the softmax function:

$$g_i(\mathbf{x}, \mathbf{v}) = \frac{e^{\beta_i(\mathrm{x},\mathrm{v})}}{\sum_{j=1}^{I} e^{\beta_j(\mathrm{x},\mathrm{v})}} \tag{3–2}$$

where $\beta_i(\mathbf{x}, \mathbf{v})$ are functions of the gate parameter $\mathbf{v}$, and are linear given by $\beta_i(\mathbf{x}, \mathbf{v}) = \mathbf{v}_i^T[\mathbf{x}, 1]$ in the original ME. The softmax function is a smooth version of the winner-take-all model.

For a K class classification, there are K parameter sets $\{\{\mathbf{w}_{ik}\}_{i=1}^{I}\}_{k=1}^{K}$ to be learned in each expert, corresponding to the parameters of each class, as shown in Fig. 3-2. The desired output $\mathbf{y}^{(n)}$ is of length K and $y_k^{(n)} = 1$ if $\mathbf{x}^{(n)}$ belongs to class k and 0 otherwise. For I the number of experts and $k : 1...K$ the class index, the expert outputs per class are softmax functions:

$$\hat{y}_{ik}^{(n)} = \frac{\exp(\mathbf{w}_{ik}^T[\mathbf{x}^{(n)}, 1])}{\sum_{r=1}^{K}\exp(\mathbf{w}_{ir}^T[\mathbf{x}^{(n)}, 1])} \quad , \tag{3–3}$$

which are the means of the experts' multinomial probability models:

$$P_i(\mathbf{y}) = \prod_k \hat{y}_{ik}^{y_k}. \tag{3–4}$$

To make a single prediction, the outputs are computed per class:

$$\hat{y}_k^{(n)} = \sum_i g_i(\mathbf{x}^{(n)})\hat{y}_{ik}^{(n)},$$

and for practical purposes, the input $\mathbf{x}^{(n)}$ is classified as belonging to the class k that gives the maximum $\hat{y}_k^{(n)}$, $k : 1...K$.

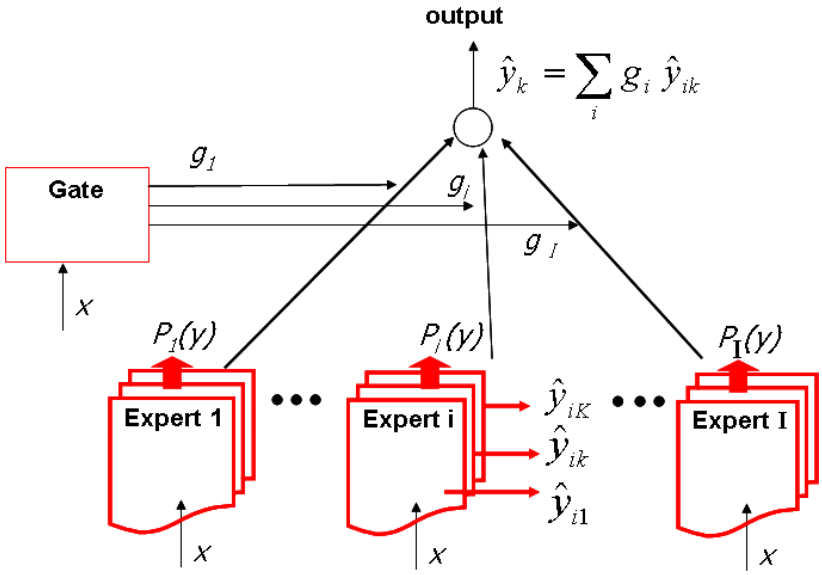


Figure 3-2. ME architecture for classification

In training the ME, indicator variables $Z = \{\{z_i^{(n)}\}_{n=1}^N\}_{i=1}^I$ are introduced such that

$$z_i^{(n)} = \begin{cases} 1 & \text{if } \mathbf{x}^{(n)} \in R_i \ , \\ 0 & \text{otherwise} \end{cases}$$

where R_i is the region specified by expert i. With the indicator variables, Eq. 3–1 can be turned into the complete data distribution

$$P(Y, Z|\mathbf{w}, \mathbf{v}) = \prod_n \prod_i \left(g_i^{(n)} P_i(y^{(n)}) \right)^{z_i^{(n)}} . \tag{3–5}$$

The complete log likelihood can be written as:

$$l(\Theta; D; Z) = \sum_{n=1}^N \sum_{i=1}^I z_i^{(n)} \{\log g_i^{(n)} + \log P_i(\mathbf{y}^{(n)})\} \ , \tag{3–6}$$

which is a random function of the missing random variables z_i. Therefore, the EM algorithm is employed to average out z_i and maximize the expected complete data likelihood $E_Z(\log P(D, Z|\Theta))$. The expectation of the likelihood in Eq. 3–6 results in:

$$\begin{aligned} Q(\Theta, \Theta^{(p)}) &= \sum_{n=1}^N \sum_{i=1}^I h_i^{(n)} \{\log g_i^{(n)} + \log P_i(\mathbf{y}^{(n)})\} \\ &= \sum_{i=1}^I Q_i^g + Q_i^e \ , \end{aligned} \tag{3–7}$$

where p is the iteration index,

$$Q_i^g = \sum_n h_i^{(n)} \log g_i^{(n)} \tag{3–8}$$

$$Q_i^e = \sum_n h_i^{(n)} \log P_i(\mathbf{y}^{(n)}) \tag{3–9}$$

and

$$h_i^{(n)} = E[z_i^{(n)}|D] = \frac{g_i^{(n)} P_i(\mathbf{y}^{(n)})}{\sum_j g_j^{(n)} P_j(\mathbf{y}^{(n)})}. \quad (3\text{–}10)$$

The parameter Θ is estimated by the iterations through the E and M steps given by:

1. E-step: Compute $h_i^{(n)}$, the expectation of the indicator variables.

2. M-step: Find a new estimate for the parameters, such that

$$\mathbf{v}_i^{(p+1)} = \underset{\mathbf{v}_i}{\text{argmax}}\ Q_i^g \ ,$$

and

$$\theta_i^{(p+1)} = \underset{\theta_i}{\text{argmax}}\ Q_i^e \ .$$

There are three important points regarding the training. Firstly, Eq. 3–8 describes the cross-entropy between g_i and h_i. In the M step, h_i is held constant, so g_i learns to approximate h_i. Remembering that $0 \leq h_i \leq 1$ and $0 \leq g_i \leq 1$, the maximum Q_i^g is reached if both g_i and h_i are 1 and the others (g_j , h_j , $i \neq j$) are zero. This is in line with the initial assumption from Eq. 3–1 that each pattern belongs to one and only one expert. If the experts are actually sharing a pattern, they pay an entropy price for it. Because of this property, ME algorithm is also referred to as competitive learning among the experts, as the experts are rewarded or penalized for sharing the data. A more detailed explanation about the effect of entropy on the training is provided in [10, 26] .

The second important point is that, by observing Eq. 3–8 and Eq. 3–9, the gate and the expert parameters are estimated separately owing to the use of the hidden variables. This decoupling gives a modular structure to the ME training, and has led to the development of the hierarchical mixtures of experts (HME) and to the use of other modular networks at the experts and the gates. The probability model for HME [2, 26] is:

$$P(\mathbf{y}|\mathbf{x}, \Theta) = \sum_{i=1}^{I} g_i(\mathbf{x}, \Theta_{g_i}) \sum_{j=1}^{I} g_{j|i}(\mathbf{x}, \Theta_{g_{j/i}}) P_{ij}(\mathbf{y}, \Theta_e) \ , \quad (3\text{–}11)$$

where g_i is the output of the gate in the top layer, $g_{j|i}$ is the output of the j^{th} gate connected to the i^{th} gate of the top layer, and Θ_{g_i} and $\Theta_{g_{j/i}}$ are their parameters, respectively.

The third important point is that, $\max_{v_i} Q_i^g$ can not be solved analytically because of the softmax function. Therefore, one can either use the iterative recursive least square (IRLS) technique for linear gate and expert models [26], the extended IRLS algorithm for non-linear gate and experts [2], or use the generalized EM (GEM) algorithm that increases the Q function but does not necessarily fully maximize the likelihood [70].

3.2 Advances in ME for Classification

The ME model was designed mainly for function approximation rather than classification, however, it has a significant appeal for multi-class classification due to the idea of training the gating network together with the individual classifiers through one protocol. In fact, more recently, the ME model has been getting attention as a means of finding sub-clusters within the data, and learning experts for each of these sub-clusters. In doing so, the ME model can benefit from the existence of common characteristics among data of different classes. This is an advantage compared to the other classifiers that do not consider the other classes when finding the class conditional density estimates from the data of each class.

Waterhouse and Robinson provided a nice overview on the parameter initialization, learning rates and stability issues in [71] for multi-class classification. Since then, there have been reports in the literature [72–74] that networks trained by the IRLS algorithm perform poorly in multi-class classification. Although these arguments have merit, it has also been shown that if the step-size parameters are small enough, then the log-likelihood is monotonic increasing and the IRLS is stable [75].

The IRLS algorithm makes a batch update, and updates all the parameter vectors of an expert $\{\mathbf{w}_{ik}\}_{k=1}^{K}$ at once, and implicitly assumes that these parameters are independent. Chen et al. [74] pointed out that IRLS updates result in an incomplete

Hessian matrix, in which the off-diagonal elements are non-zero, implying the dependence of the parameters. In fact, in multi-class classification, each parameter vector in an expert relate to each other through the softmax function in Eq. 3–3, and therefore, these parameter vectors can not be updated independently. Chen et al. noted that such updates result in unstable log-likelihoods, so they suggested using the exact Hessian matrix in the inner loop of the EM algorithm. However, the use of the exact Hessian matrix results in expensive computations, therefore, they proposed using the generalized Bernoulli density in the experts for multi-class classification as an approximation to the multinomial density. With this approximation, all of the off-diagonal block matrices in the Hessian matrix are zero matrices, and the parameter vectors are separable. This approximation resulted in simplified Newton-Raphson updates and required less time; however, the error rates increased due to the fact that it was merely an approximation.

Following this study, Ng and McLachlan [75] ran several experiments to show that the convergence of the IRLS algorithm is stable if the learning rate is kept small enough, and the log-likelihood is monotonic increasing even though the assumption of independence is incorrect. However, they also suggested using the expectation-conditional maximization (ECM) algorithm with which the parameter vectors can be estimated separately. The ECM algorithm basically suggests to learn the parameter vectors one by one, and to use the updated parameters while learning the next parameter vector. In doing so, the maximizations are over smaller dimensional parameter space and are simpler than a full maximization, and the convergence property of the EM algorithm is kept. In 2007, Ng and McLachlan [76] presented an ME network for binary classification, in which, the interdependency between the hierarchical data was taken into account by incorporating a random effects term into the experts and the gates. The random effects term in an expert provided information as to whether there is a significant difference in local outputs from each expert network, and it was shown to increase the classification rates.

Recently, ME has found interest in classification studies that can benefit from the existence of sub-clusters within the data. In a study by Titsias and Likas [77], the ME classification formulation was written more explicitly, such that, the probability of selecting an expert and the probability of selecting the sub-cluster of a class within an expert were written separately as:

$$I(\Theta; D; Z) = \sum_{k=1}^{K} \sum_{x \in X_k} \sum_{i=1}^{I} z_i(\mathbf{x}) \{\log g_i P(C_k|i) p(\mathbf{x}|C_k, i, \theta_{ki})\} \quad , \tag{3–12}$$

where $p(\mathbf{x}|C_k, i, \theta_{ki})$ is the Gaussian model of a sub-cluster corresponding to class C_k, and θ_{ki} are the corresponding parameters.

In a 2008 study by Xing and Hu [35], unsupervised clustering was used to initialize the ME model. In the first stage, a fuzzy C-means (FCM)-like algorithm was used for carving all the unlabeled data into several clusters, and a small fraction of these samples from the cluster centers were chosen as training data. In the second stage, several parallel two-class MEs were trained with the corresponding two-class training data sets. Finally, given a test data, its class label was determined as the largest vote of the MEs.

In using clustering approaches for the initialization of ME, a good cluster can be a good initialization to the gate and speed-up the training significantly. On the other hand, a strong clustering may lead to an unnecessary number of experts, or lead to the over-training of the data. It might also force the ME to a local optima where it would be hard to get out. Therefore, it would be interesting to see the effect of initialization with clustering on the ME model. Another work that would be interesting would be to see the performance of ME for a K-class problem where $K > 2$ and compare it to the $\binom{K}{2}$ comparisons of 2-class MEs and the decision from their popular vote.

3.3 Modifications to ME to Handle Sequential Data

The original formulation of ME was for static data, and was based on the statistical independence assumption of the training pairs. Hence, it did not have a formulation to handle the causal dependencies. To overcome this problem, several studies extended

ME for time-series data. In the past decade, ME has been applied to regression of time-series data in a variety of applications. It was found to be a good fit in such applications where the time-series data is non-stationary, meaning the time series switch their dynamics in different regions, and it is difficult for a single model to capture the entire dynamics of the data. For example, Weigend et al. [10] used ME to predict the daily electricity demand of France, which switches among regimes, depending on the weather, seasons, holidays, and workdays, establishing daily, seasonally and yearly patterns. Similarly, Lu [37] used ME for climate prediction because the time series dynamics switch between different regions. For such data, ME showed success in finding both the decomposition and the piecewise solution in parallel. In most of the following papers, the main idea is that the gating divides the data into regimes, and the experts learn local models for each regime.

For time-series data, Zeevi et al. [78] and Wong et al. [79] generalized the autoregressive models for time series data by combining them with an ME structure. In the speech-processing community, Chen et al. [9, 72, 80] used HME for text-dependent speaker identification. In [72] the features were calculated from a window of utterances to introduce some temporal information to the solution. In [9] a modified HME structure was introduced. In the modified HME, a new gating network was added to make use of the transitional spectral information while the original HME architecture dealt with instantaneous information. The new gating network, called the S-gating network, was placed at the top of the tree and given a paired observation (X, y) with $X = x_1, ..., x_T$; the probabilistic model was modified as:

$$P(y|X,\Theta) = \sum_{t=1}^{T} \lambda_X(x_t,\Phi) \sum_i g_i(x_t,v_i) \sum_j g_{j/i}(x_t,v_{ij}) P(y|x_t,\theta_{ij}), \qquad (3\text{–}13)$$

with

$$\lambda_X(x_t) = \frac{P(x_t|\Phi)}{\sum_{s=1}^{T} P(x_s|\Phi)}, \tag{3–14}$$

where $P(x_t|\Phi)$ is a Gaussian distribution. Then, for a speaker identification system of population K, they selected the unknown speaker k^* that gives the highest regression probability out of the K models. Using this model, Chen et al. modified the EM-update equations and solved for the parameters of the top-most Gaussian gate analytically, whereas the rest of the parameters for the expert and gating networks were found iteratively. With this work, HME gained the capability to handle sequences of observations, but the experts and gating networks (except the extra gate) were still linear models.

For non-stationary data, Cacciatore and Nowlan [81] suggested using recurrence in the gating net, such that, the input to the gating network was the ratio of the outputs at the two preceding time steps. Weigend et al. [10] developed gated ME to handle time-series data that switches regimes. A gating network in the form of a multilayer perceptron combines the outputs of the neural network experts. Hence, while the gate discovers the hidden regimes, the experts learn to predict the next observed value. This gated ME approach was extended by Coelho et al. [12], where training was accomplished using genetic algorithms instead of gradient descent. A similar idea to detect switching regimes was also visited by Liehr et al. [82] where the gating was a transition matrix, and the experts were Gaussian functions.

Most of these ME models that handle time-series data on regression use a one-step-ahead or multi-step-ahead prediction, in which the last d values of the time-series data are used as a feature of d-dimensions in a neural network. The benefit of using a sliding window type technique is that a sequential supervised learning problem can be converted into a classical supervised learning problem. However, these models cannot handle data with varying length, and the use of multilayer network type approaches prevent them from completely describing the temporal properties of a time-series data. Such problems were discussed by Dietterich in [83], for predicting the

pronunciation of English words from spelling. As an example, Dietterich noted that the only difference between the words “thought” and “though” is the final “t”, yet it influences the initial pronunciation of the initial “th”. Hence, a small window might cause to miss the longer-range interactions, or a large window might consider features that are not really relevant. Therefore, it is of interest to develop models that can fully use the information from time-series or sequential data by using models such as the hidden Markov models, which we show in Chapter 6.

In a number of studies by Wang et al. [14] and by Li [39], the experts are the states of a hidden Markov model (HMM), and the goal is to estimate the HMM parameters. The architecture introduced by Wang et al. [14] was named the time-line hidden Markov experts, and it segments the trajectory into several states and learns each of them using a local expert. The learning of the time-line HMM is to estimate the parameters of the HMM for observing the training samples. In time-line hidden Markov experts, transition probabilities are also time-varying and designed to be conditioned on the input. The local experts can be some sort of a regression model depending on the particular modeling case. The time-line HMM combines the experts using the state probabilities of the HMM, which are determined by the previous state of the HMM and the state transition probabilities simulated by the state transition network.

Similarly, the probability density estimation of an HMM has been mimicked using ME systems in [13, 84–87]. Zhao et al. [84] mimicked an HMM by using an HME system in which, hierarchically organized, relatively small neural networks were trained to perform the probability density estimation. Meila et al. [85] implemented the transition matrix with a bank of gating networks, and the observation probability densities were modified to be centered around the experts.

Bengio et al. [13] extended the ME model in a structure named an input-output HMM (IOHMM) with a linear feedback loop and two sets of experts. The first set of experts compute the output given the current state and input, whereas the second

set of experts make use of the feedback loop and compute the next state distribution given the input and the previous state (i.e. the state transition probabilities). The learning is carried out with an EM algorithm, where the update equations are similar to the forward backward recursions of HMM. A nice feature of this study is that the transition probabilities are time-varying and depend on the input to the system. However, the limitation of both the gated ME and the IOHMM is that the experts are multilayer perceptrons, which may not always be appropriate for time-series modeling. Fritsch et al. [86] developed a neural network based system for estimating emission probabilities in an HMM system such that the neural network has as many output nodes as there are HMM states, and the neural network produces estimates of posterior state probabilities. The outputs of these neural networks are then combined with HME.

To remove the independent and identically distributed (iid.) assumption of the data that was necessary in the original HME model, and to find a model appropriate for time-series data, Jordan et al. [88] described a decision tree with Markov temporal structure named as a hidden Markov decision tree (HMDT), in which each decision in the decision tree is dependent on the decision taken at the node at the previous step. The result was an HMM in which the state at each moment in time was factorized, and the factors are coupled to form a decision tree. This model is an effort to combine adaptive graphical probabilistic models such as the HMM, HME, IOHMM and factorial HMM [89] in a variational study.

The benefits of these approaches of modeling an HMM with HME is that, the restrictions on HMM because of the Markov property can be relaxed, or one can replace the observation distributions with more complex distributions. However, why such approaches did not become as famous as HMMs might be due to the need to learn more parameters and to find elegant training algorithms.

In none of these studies has HME been used for the multi-class classification of sequential data. Therefore, in Chapter 6, we develop the mixture of HMM experts model, provide its derivation and the training algorithm.

3.4 Comparison to Popular Algorithms

A comparison of ME with some of the popular models was given at the beginning of the chapter. In this section, we extend these comparisons to more models. HME can be regarded as a statistical approach to decision tree modeling where the decisions are treated as hidden multinomial random variables. Therefore, in comparison to the decision trees such as CART [90], HME uses soft boundaries, and allows us to develop inference procedures, measures of uncertainty, and Bayesian approaches. Also, it was discussed by Haykin [57] that an HME can recover from a poor decision somewhere further up the tree, whereas a standard decision tree suffers from a greediness problem, and gets stuck once a decision is made.

HME bears a resemblance to the boosting algorithm [91, 92] in that the weak classifiers are combined to create a single strong classifier. ME finds the subsets of patterns that naturally exist within the data, and learns these easier subsets to solve the bigger problem. In boosting on the other hand, each classifier becomes an expert on difficult patterns on which other classifiers make an error. Hence, the mixture coefficient in boosting depends on the classification error and provides a linear combination, whereas the mixture coefficient (the gate) in the HME depends on the input and makes probabilistic combination of experts. This difference in mixing makes the training of these two algorithms significantly different such that, in boosting, the classifiers are trained sequentially based on the data that was filtered by the previously trained networks. Once the training data is specified, the boosting networks are learned independently, and combined with a mixture coefficient that is learned from the classification error. In HME, the experts compete with each other for the right to learn particular patterns; hence, all the experts are updated at each iteration depending

on the gate's selection. In an effort to incorporate boosting into ME, Waterhouse and Cook [93] initialized a split of the training set learned from boosting to different experts. In boosted ME, Avnimelech and Intrator [59] added the experts one by one, and trained the new experts with the data on which the other experts were not confident. The gating function in this case was a function of the confidence. One can think of these boosted ME algorithms as a smart preprocessing the data. Another study on preprocessing was published by Tang et al. [54] where self organizing maps (SOM) were used, in which, as opposed to feeding all the data into the expert networks, only the local regions of the input space found by SOM were assigned to the individual experts.

Multivariate adaptive regression splines (MARS) model partitions the input space into overlapping regions and fits a univariate spline to the training data in each region. The same mixture coefficient argument also applies to the MARS model [94], which has the equational form of a sum of weighted splines. In comparison to the latent variables of HME, MARS defines the states by the proximity of the observed variables. On the other hand, the Bayesian MARS is non-parametric, and requires sampling methods. It was found that HME requires less memory as it is a parametric regression approach, and the variational Bayesian inference for HME converges faster in time [49] .

In comparison to neural networks (NN), Nowlan and Hinton [95] found ME to be better at fitting the training data. When dealing with relatively small training sets, ME was better at generalizing than a comparable single back-propagation network on a vowel recognition task. HME was shown to learn much faster than back-propagation for the same number of parameters by Jordan and Jacobs [96]; however, it is questionable if this was a good comparison since NNs were trained using a gradient-descent algorithm, whereas HME was trained using second order methods. Additionally HME provide insightful and interpretable results that the NNs do not provide. In terms of the degree of approximation bounds, NNs and ME were found equivalent by Zeevi et al. [61].

Other models of ME include the max-min propagation neural network by Estevez et al. [97] where the softmax function was replaced with max(min) units; and the model by Lima et al. [33] where neural networks were used at the gate and support vector machines at the experts.

3.5 Experimental Results on Landmine Data

For landmine detection, it is convenient to present the experimental results using the receiver operating characteristics (ROC) curves. The ROC curve specifies the probability of detection (PD) vs. the probability of false alarm (PFA), that are defined as PD $=$ Hit$/($ Hit + Miss$)$ and PFA $=$ FA$/($FA + Reject$)$, where the hit, miss, false alarm (FA) and reject options are defined in Table 3-1.

Table 3-1. Naming conventions for receiver operating characteristics (ROC) curves

Algorithm / Actual	Mine	Non-mine
Mine	Hit	False alarm
Non-mine	Miss	Reject

The data from the experiments were collected in August 2007 from two geographically separate test sites that also have different soil properties. Combined, both sites had a total of 11 high metal anti-tank mines, 49 low metal anti-tank mines, 30 high metal anti-personnel mines, 66 low metal anti-personnel mines as well as 90 high metallic clutter objects, 28 medium metallic clutter objects, 28 low metallic clutter objects, 143 non-metallic clutter objects, and blanks. At each site, there were two collections; the first collection was performed in the northwest (NW) direction, and the second collection was done in the southeast (SE) direction. The samples from both of these collections were combined with a special requirement: if an object was included in the training, it was included using the data from both collections (hence both NW and SE collections of the same object data went into training), and neither was included in the testing. More on this dataset was given in Sec. 2.1.

The experimental results were trained and tested with a 10-cross-fold validation; i.e., the dataset was divided into 10 subsets, 9 subsets were used for training, and 1 subset was used for testing; and this validation was continued until all the subsets were tested.

Eight classes were trained corresponding to HMAT, HMAP, LMAT, and LMAP mines as well as different kinds of clutter (HMC, LMC, NMC) and blank grids. An HME architecture of depth two with eight experts was used. To obtain a final mine confidence value, the confidences from the first four mine classes were summed up. To have a final non-mine confidence value, the confidences from the four non-mine classes were summed up, then converted to a mine-confidence by subtracting from 1.

The two WEMI metal detector classifiers, namely the angle prototype match and angle model based K-NN gave 90/18 and 90/30 detection rates, respectively as shown in Fig. 3-3. The two GPR classifiers, namely the spectralLPP and EHD, gave detection rates of 90/50 and 90/51, respectively. When the HME algorithm was used, the detection rates increased to 90/10, outperforming the single use of all the algorithms.

When the HME algorithm is run using just the EHD feature from GPR data and the angle model feature from WEMI data, the regions picked by the HME for all classes is displayed in Fig. 3-4. By finding the maximum confidence values of all these regions, the dominant classes are displayed in Fig. 3-4(A). This result is inline with what an expert person would expect for each class of objects. Hence instead of having just a confidence value, such a graph gives the technician the opportunity to visually understand where and why a confidence value was generated. In doing so, (s)he may choose to accept a low confidence value object in the low metal anti-personnel region as a mine, as those are the hardest mines to detect.

In summary, the ME and HME models combine simple expert models with mixing weights, called gating functions, that depend on the input. Learning the ME (or HME) means assigning the experts to various regions, and training each to best adapt to its region. HMEs find the true segmentations when there is one, and they come up with

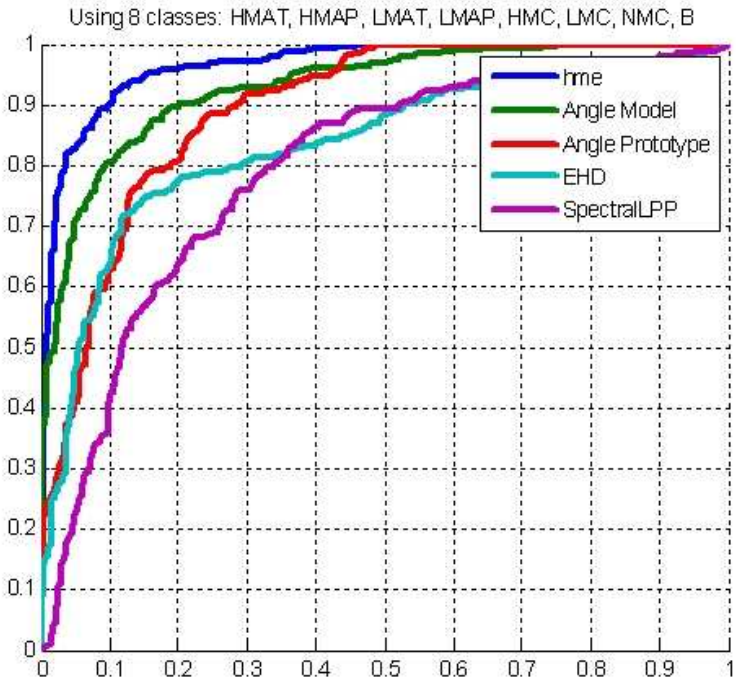


Figure 3-3. ROC curves on landmine data with 10-fold cross validation. The two WEMI metal detector classifiers, namely the angle prototype match and angle model based k-NN gave 90/18 and 90/30 detection rates, respectively. The two GPR classifiers, namely the spectralLPP and EHD, gave detection rates of 90/50 and 90/51, respectively. When the HME algorithm was used, the detection rates increased to 90/10, outperforming any single use of all the algorithms.

good segmentations when there are none. Thus, they were found to be very useful tools by the landmine detection community as published in [43]. With the ME model, a general decision can be made about into which category a landmine falls. Hence, some future work that would be useful to the landmine detection community is to design and run algorithms specifically trained to detect different kinds of mines.

There are two shortcomings to this approach. The first shortcoming is that the EM training may lead to over-fitting. To overcome this problem, a variational ME that regularizes ME and avoids over-fitting is introduced in Chapter 4. The second shortcoming is that the ME works on static features only and cannot handle time-series

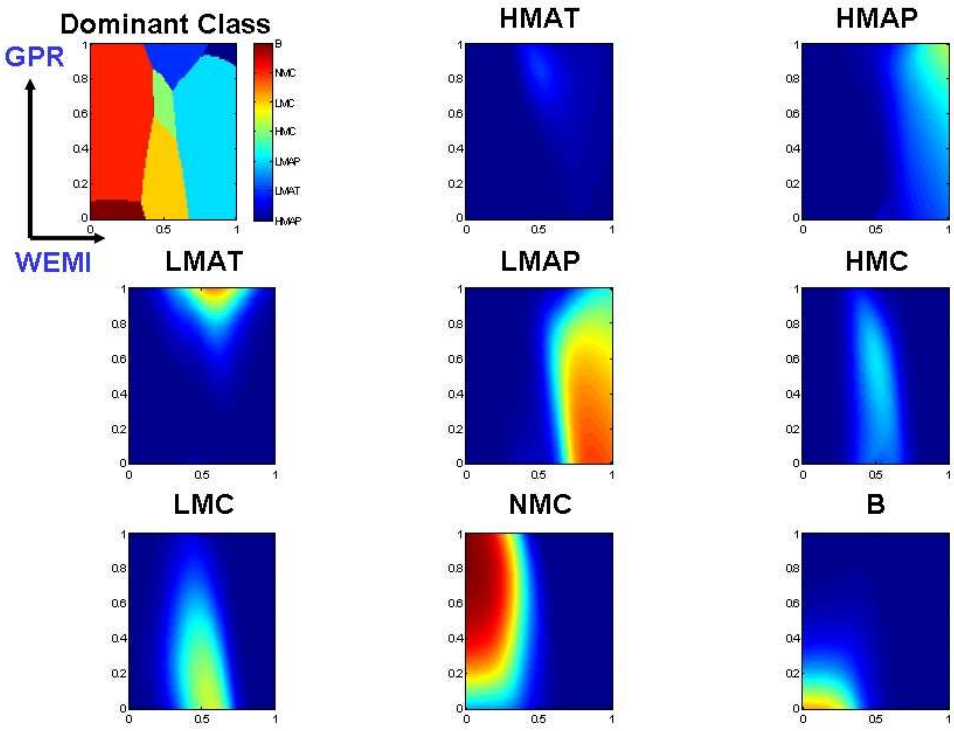


Figure 3-4. The dominant and the soft regions of each class found from HME using EHD and angle model features. The maximum of the soft regions is taken as the dominant class, and displayed in the top-leftmost figure.

data. In the current landmine-detection setting, the object was assumed to be under the radar, so static features were extracted from the region of interest. However, such a setting does not make use of the surrounding information, and it ignores the time-series data collection from moving radar. To overcome these issues, a mixture of HMM experts algorithm is developed in Chapter 6.

CHAPTER 4
VARIATIONAL MIXTURE OF EXPERTS FOR CLASSIFICATION (VMEC)

Variational methods, also called as ensemble methods, variational Bayes, or variational free energy minimization, is a technique for approximating a complicated posterior probability distribution $P(\mathbf{w}|D)$ by a simpler ensemble $Q(\mathbf{w}, \theta)$. The key to this approach is that, as opposed to the traditional approaches where the parameter $\mathbf{w}$ is optimized to find the mode of the distribution, variational methods define an approximating probability distribution over the parameters, $Q(\mathbf{w}, \theta)$, and optimize this distribution by varying θ so that it approximates well the posterior distribution of the parameters $P(\mathbf{w}|D)$ [98]. Hence, instead of point estimates for $\mathbf{w}$ representing the mode of a distribution, variational methods produce an estimate for the whole distribution.

The mixture of experts (ME) training is accomplished through the expectation-maximization (EM), however, EM training is sensitive to initialization and does not regularize the parameters or use any prior information. Thus it is prone to over-training. To address some of these problems, a variational mixture of experts (VME) was proposed in [4–8] for regression and used in [30, 31] for 3D motion tracking. The similarities of the classification and regression algorithms were discussed in [7], although a clear framework was not given for classification. Also, the previous algorithms used the number of iterations as a stopping condition, which may lead to the premature termination of the algorithm or unnecessary training. In this study, a complete learning algorithm for variational mixture of experts classification (VMEC) is introduced. To state more explicitly how our approach builds on the work by Waterhouse [7], in a K class classification, rather than a single weight vector, there are K weight vectors for each expert. Thus, instead of a single hyper-parameter, our VMEC model assumes K hyper-parameters for each expert. Similarly, instead of assuming a single distribution per expert, there are K distributions per expert in VMEC. In addition, a distribution over the hyper-parameters was not assumed in [7], but such an assumption is necessary to

calculate the lower bound which provides an excellent stopping criterion that counters over-training. With these assumptions, the joint distribution was modified to be a product of the aforementioned distributions and the lower bound was derived using this modified joint distribution.

Thus, in this chapter, (1) a complete learning algorithm for VMEC is given, (2) a variational lower bound is derived for VMEC, and (3) the efficacy of the approach is presented on synthetic data as well as on landmine data to compare to the ME trained with EM. In the rest of the chapter, ME indicates the original mixture of experts model trained with the EM algorithm.

4.1 VMEC Model

Following the same notation of the ME algorithm in Chapter 3, we let $\Theta = \{\{\mathbf{v}_i, \mathbf{w}_{ik}\}_{i=1}^{I}\}_{k=1}^{K}$ denote the gate and expert parameters, and place Gaussian priors $\Phi = \{\{\mu_i, \alpha_{ik}\}_{i=1}^{I}\}_{k=1}^{K}$ on the experts and the gates

$$P(\mathbf{w}|\alpha) = \prod_{i,k} P(\mathbf{w}_{ik}|\alpha_{ik}) = \prod_{i,k} N(\mathbf{w}_{ik}|0, \alpha_{ik}^{-1}\mathbf{I}),$$

$$P(\mathbf{v}|\mu) = \prod_{i} P(\mathbf{v}_i|\mu_i) = \prod_{i} N(\mathbf{v}_i|0, \mu_i^{-1}\mathbf{I}).$$

We also assume that the hyperparameters are Gamma distributed:

$$P(\mu) = \prod_{i} P(\mu_i) = \prod_{i} Gam(\mu_i|c_0, d_0),$$
$$P(\alpha) = \prod_{i,k} P(\alpha_{ik}) = \prod_{i,k} Gam(\alpha_{ik}|a_0, b_0).$$

The dependence of the hyperparameters and the parameters is illustrated in Fig. 4-1.

Using the distributions of the hyperparameters, the joint distribution can be written as:

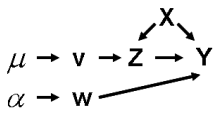


Figure 4-1. Directed acyclic graph representing the VMEC model showing the dependence of the hyperparameters and the parameters.

$$P(\Theta, \Phi, Z, D) = P(Y, Z|\mathbf{w}, \mathbf{v})P(\mathbf{w}|\alpha)P(\alpha)P(\mathbf{v}|\mu)P(\mu). \tag{4–1}$$

In the variational approach, the goal is to find the distribution Q that best approximates the posterior distribution, so the evidence $P(D)$ is decomposed using

$$\log P(D) = L(Q) + KL(Q||P), \tag{4–2}$$

where L is the lower bound

$$L(Q) = \int Q(\Theta, \Phi, Z) \log \frac{P(\Theta, \Phi, Z, D)}{Q(\Theta, \Phi, Z)} d\Theta d\Phi dZ, \tag{4–3}$$

and KL is the Kullback-Leibler divergence

$$KL(Q||P) = -\int Q(\Theta, \Phi, Z) \log \frac{P(\Theta, \Phi, Z|D)}{Q(\Theta, \Phi, Z)} d\Theta d\Phi dZ. \tag{4–4}$$

The Q distribution minimizes the KL-divergence; however, working on the KL-divergence would be intractable, so we maximize the lower bound L [99], as illustrated in Fig. 4-2.

We assume the approximating distribution factorizes as:

$$Q(\Theta, \Phi, Z) = Q(Z) \prod_i Q(\mathbf{v}_i)Q(\mu_i) \prod_k Q(\mathbf{w}_{ik})Q(\alpha_{ik}). \tag{4–5}$$

Plugging the joint distribution (Eq. 4–1) and the Q distribution (Eq. 4–5) into the lower bound equation (Eq. 4–3), and taking the expectations with respect to all the other variables, we obtain the Q distributions as:

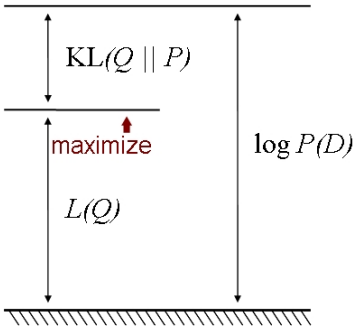


Figure 4-2. Variational methods minimize the KL-divergence and maximize the lower bound.

$$Q^*(\mathbf{w}_{ik}) = N(\mathbf{w}_{ik}|\overline{\mathbf{w}}_{ik}, A_{\mathbf{w}_{ik}}), \tag{4–6}$$

$$Q^*(\alpha_{ik}) = Gam(\alpha_{ik}|a_p, b_p), \tag{4–7}$$

$$Q^*(\mathbf{v}_i) = N(\mathbf{v}_i|\overline{\mathbf{v}}_i, A_{\mathbf{v}_i}), \tag{4–8}$$

$$Q^*(\mu_i) = Gam(\mu_i|c_p, d_p), \tag{4–9}$$

$$Q^*(Z) = \prod_{n=1}^{N}\prod_{i=1}^{I} h_i^{(n)^{z_i^{(n)}}}. \tag{4–10}$$

Here $\overline{\mathbf{w}}_{ik}$ and $\overline{\mathbf{v}}_i$ are the means of the Gaussians and they are found using Newton-Raphson updates. The covariance matrices $A_{\mathbf{w}_{ik}}$ and $A_{\mathbf{v}_i}$ are the inverse of the negative Hessian matrices, $A_{\mathbf{w}_{ik}} = -H_w^{-1}$ and $A_{\mathbf{v}_i} = -H_v^{-1}$, that are explained below.

For a learning rate η, expert parameters are found by

$$\mathbf{w}_{ik}^{(p+1)} = \mathbf{w}_{ik}^{(p)} - \eta H_w^{-1} G_w, \tag{4–11}$$

where

$$G_w = \sum_n \overline{h}_i^{(n)}(y_k^{(n)} - \widehat{y}_{ik}^{(n)})\mathbf{x}^{(n)} - \overline{\alpha}_{ik}\mathbf{w}_{ik}, \tag{4–12}$$

$$H_w = -\sum_n \overline{h}_i^{(n)}\widehat{y}_{ik}^{(n)}(1 - \widehat{y}_{ik}^{(n)})(\mathbf{x}^{(n)})(\mathbf{x}^{(n)})^T - \overline{\alpha}_{ik}.I \quad . \tag{4–13}$$

Similarly, the updates to the gating parameters follow

$$\mathbf{v}_i^{(p+1)} = \mathbf{v}_i^{(p)} - \eta H_v^{-1} G_v, \tag{4–14}$$

where

$$G_v = \sum_n (h_i^{(n)} - g_i^{(n)})\mathbf{x}^{(n)} - \overline{\mu}_i \mathbf{v}_i, \tag{4–15}$$

$$H_v = -\sum_n g_i^{(n)}(1 - g_i^{(n)})(\mathbf{x}^{(n)})(\mathbf{x}^{(n)})^T - \overline{\mu}_i . I \quad . \tag{4–16}$$

Newton-Raphson updates are continued in a loop until the $P(Y, Z|\mathbf{w}, \mathbf{v})$ updates converge; and the parameters found at the last iteration are taken to be $\overline{\mathbf{w}}_{ik}$ and $\overline{\mathbf{v}}_i$, where $A_{\mathrm{w}_{ik}}$ and A_{v_i} are their covariance matrices.

The updates for expert hyper-hyperparameters are

$$a_p = a_0 + \frac{d}{2}, \tag{4–17}$$

$$b_p = b_0 + \frac{1}{2}(\mathbf{w}_{ik}^T \mathbf{w}_{ik} + Trace(A_{\mathrm{w}_{ik}})), \tag{4–18}$$

and similar equations apply for the gate hyper-hyperparameters c_p and d_p. As a result, hyperparameter updates become

$$\alpha_{ik}^{(p+1)} = a_p / b_p, \tag{4–19}$$

and

$$\mu_i^{(p+1)} = c_p / d_p. \tag{4–20}$$

4.1.1 VMEC Lower Bound

Parameter updates are continued until the lower bound converges to a given small number (e^{-5} in our case), and the lower bound provides a test of correctness as it is supposed to be nondecreasing at each parameter re-estimation. Expanding the integral and evaluating the expectations, we arrive at the closed form solution for the lower bound as:

$$
\begin{aligned}
L(Q) &= \int Q(\Theta, \Phi, Z) \log \frac{P(\Theta, \Phi, Z, D)}{Q(\Theta, \Phi, Z)} d\Theta d\Phi dZ, \\
&= \sum_{n,i} E_{Z,\mathrm{v},\mathrm{w}}[\log P(Y, Z|X, \mathbf{v}, \mathbf{w})], \\
&+ \sum_{i} E[\log P(\mathbf{v}_i|\mu_i)] + \sum_{i} E[\log P(\mu_i)] + \sum_{i,k} E[\log P(\mathbf{w}_{ik}|\alpha_{ik}) + \sum_{i,k} E[\log P(\alpha_{ik})], \\
&- \sum_{i} E[\log Q(\mathbf{v}_i)] - \sum_{i} E[\log Q(\mu_i)] - \sum_{i,k} E[\log Q(\mathbf{w}_{ik})] - \sum_{i,k} E[\log Q(\alpha_{ik})] - E[\log Q(Z)],
\end{aligned}
$$

where

$$
\begin{aligned}
E_{\mathrm{w},\alpha}[\log P(\mathbf{w}|\alpha)] &= \frac{d}{2}[\psi(a_p) - \log b_p] - \frac{d}{2}\log(2\pi) - \frac{a_p}{2b_p}\left(\overline{\mathbf{w}}_{ik}^T \overline{\mathbf{w}}_{ik} + Trace(A_{\mathrm{w}_{ik}})\right), && \text{(4–21)} \\
E_{\alpha}[\log P(\alpha)] &= a_0 \log b_0 - b_0(a_p/b_p) - \log\Gamma(a_0) + (a_0 - 1)[\psi(a_p) - \log b_p], && \text{(4–22)} \\
E_{\alpha}[\log Q(\alpha)] &= -\log\Gamma(a_p) + (a_p - 1)\psi(a_p) + \log b_p - a_p, && \text{(4–23)} \\
E_{\mathrm{w}}[\log Q(\mathbf{w}_{ik})] &= \frac{1}{2}\log|A_{\mathrm{w}_{ik}}| + \frac{d}{2}(\log(2\pi) + 1), && \text{(4–24)} \\
E_{Z}[\log Q(Z)] &= \sum_{n} h_i^{(n)} \log h_i^{(n)}, && \text{(4–25)} \\
E_{Z,\mathrm{v}}[\log P(Z|\mathbf{v}_i)] &= \sum_{n} h_i^{(n)} \log \overline{g}_i^{(n)}, && \text{(4–26)} \\
E_{Z,\mathrm{w}}[\log P(Y|Z, \mathbf{w}_{ik})] &= \sum_{n} h_i^{(n)} \log \overline{P}_i(\mathbf{y}^{(n)}). && \text{(4–27)}
\end{aligned}
$$

Expressions for the gate $E_{\mu}[\log P(\mu_i)])$, $E_{\mu}[\log Q(\mu_i)])$, $E_{\mathrm{v},\mu}[\log P(\mathbf{v}_i|\mu_i)])$, $E_{\mathrm{v}}[\log Q(\mathbf{v}_i)]$ are similar to those of the experts. The training algorithm of the VMEC model updates

the parameters through the E and M steps until the change in the lower bound becomes less than a threshold (e^{-5} in our case).

4.1.2 Training the VMEC Model

1. For a 1-of-K class problem, initialize the number of experts I, parameters, and the hyperparameters.
2. E step: Compute the expert and gating outputs $\widehat{y}_{ik}^{(n)}$, $g_i^{(n)}$ as well as the expert probabilities $P_i(\mathbf{y}^{(n)})$ and the posterior probabilities $h_i^{(n)}$.
3. M step: Compute the new expert parameters $\mathbf{w}_{ik}^{(p+1)}$ and the new gating parameters $\mathbf{v}_i^{(p+1)}$ using Newton-Raphson updates.
4. Update the hyperparameters $\alpha_{ik}^{(p+1)}$ and $\mu_i^{(p+1)}$.
5. Check the convergence of the lower bound. Go to Step 2 if $L(p+1)-L(p) > 1e-5$; else terminate.

4.2 Experimental Results on Synthetic Data

Synthetic data was generated by sampling from two Gaussian distributions with standard deviation 0.1 at means $(0.5, 0.7)$ and $(0.7, 0.7)$, as shown in Fig. 4-3. For testing, 200 points were generated from each class. For training, the number of points were increased at each iteration, from 10 points to 60 points per class. At each iteration, a different set of training points were used.

When only one expert is used, VMEC and ME give similar results on both the training and the testing set, as expected. In Fig. 4-4, the classification rates are displayed for one expert. However, if the number of experts is increased, ME tends to over-carve the regions, thus overtrains. This behavior is shown in Fig. 4-5. For five experts, ME does better than VMEC on the training set, but its classification rates on the testing set decreases because of over-training. The consistent behaviour of VMEC is explained by the fact that its gate prefers fewer experts. Ultimately, this characteristic can be used to determine the maximum number of experts required for the best classification results.

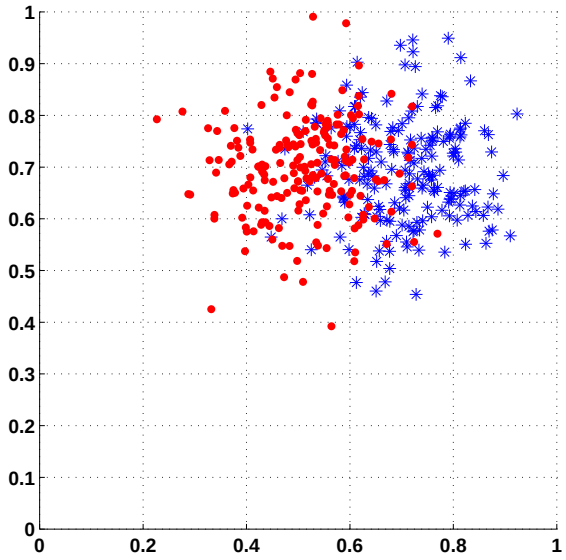
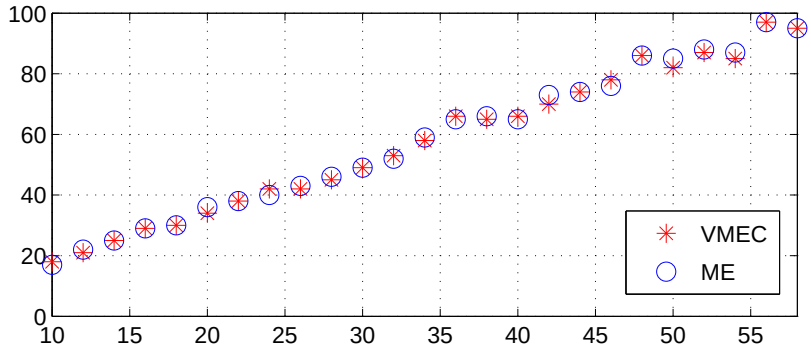


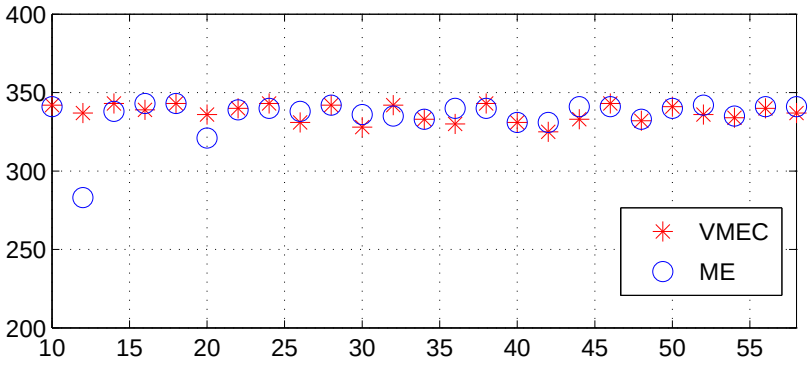
Figure 4-3. Synthetic data for the VMEC experiment. Synthetic data were generated by sampling from two Gaussian distributions with standard deviation 0.1 at means $(0.5, 0.7)$ and $(0.7, 0.7)$.

4.3 Experimental Results on the Landmine Dataset

For the landmine dataset, the two classifiers, GRANMA and EHD were described in Sec. 2.1, and in [19, 23, 43]. Using the confidences obtained from these two classifiers, VMEC and ME models were run five times using five classes, five experts, and ten fold cross-validation. The 5 classes represent the high metal mines (HMAT and HMAP), LMAT, LMAP, metallic and non-metallic clutter. The number of experts was found experimentally. If it is increased, the ME model tends to over-fit, and it either carves the input space unnecessarily, or learns the same expert, also unnecessarily. On the average, the VMEC model decreased the probability of false alarms to 11.4 % from 15.4 % of ME at 90 % PD. The individual performances of the ME and the VMEC model are given in Table 4-1, for each experiment on the mine data. In Fig. 4-6, ROC curves zoomed around 90% PD are displayed for one experiment with five classes, five experts, and ten fold cross-validation. In Fig. 4-7, the increase in the lower bound in one of the



A Number of true classification on training data with one expert.



B Number of true classification on test data with one expert.

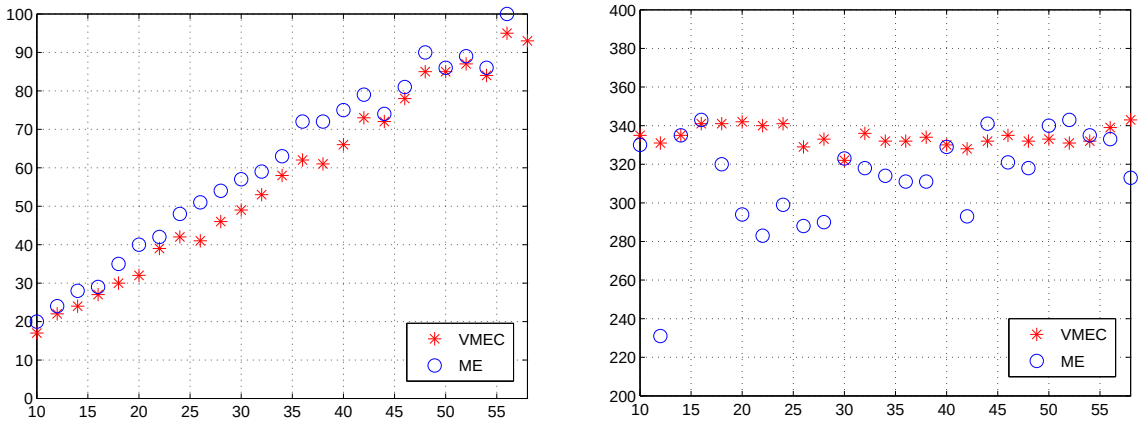
Figure 4-4. VMEC and ME comparison on synthetic data for one expert. The number of training data per class is increased from 10 to 55 with 5 point increments. At each experiment, a different dataset was generated. For varying numbers of training data, when there is only one expert, VMEC and ME give similar results both in the training set and on the testing set, as expected.

Table 4-1. Probability of false alarm (PFA) at 90% probability of detection (PD)

Experiment	1	2	3	4	5	Average
ME	0.13	0.16	0.15	0.18	0.15	0.1540 ∓ 0.0182
VMEC	0.11	0.11	0.10	0.13	0.12	0.1140 ∓ 0.0114

ten folds is displayed. The sharp increase in the lower bound corresponds to one of the hyper-parameter updates.

VMEC improved the performance over ME in these experiments with real-world data, and a part of these results were published in [44]. Since the number of experts is relatively small, the optimal number was found experimentally. For other datasets that



A Number of true classification on training data. B Number of true classification on test data.

Figure 4-5. VMEC results for five experts. The number of training data per class is increased from 10 to 55 with 5 point increments.When the experts are increased, ME does a better job of classification than VMEC on the training set, but does worse on the testing set. VMEC avoids over-training, and keeps giving consistent classification rates.

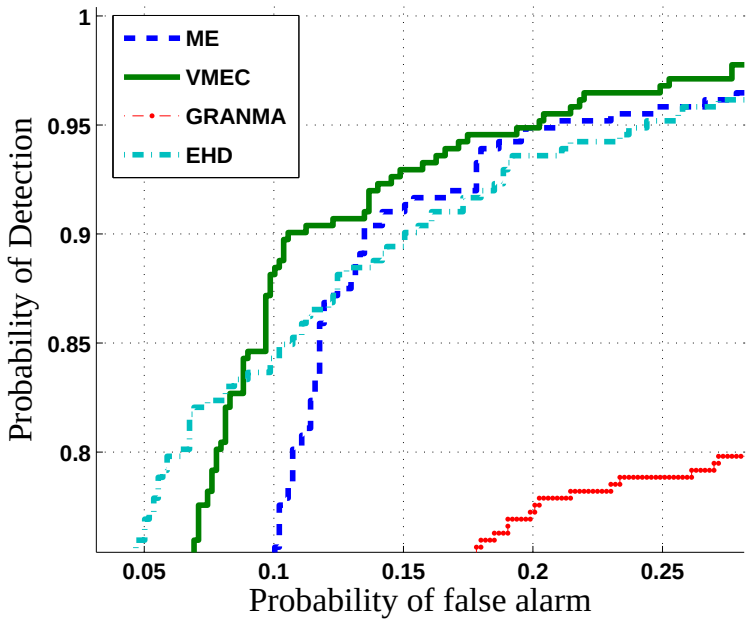


Figure 4-6. Landmine detection results for GRANMA and EHD features. ROC curves of ME, VMEC, GRANMA and EHD. In a setting of 5 classes and 5 experts, for a 10 fold cross-validation, VMEC model consistently increases detection rates to around 90/11.6 percent from 90/16.2 of ME.

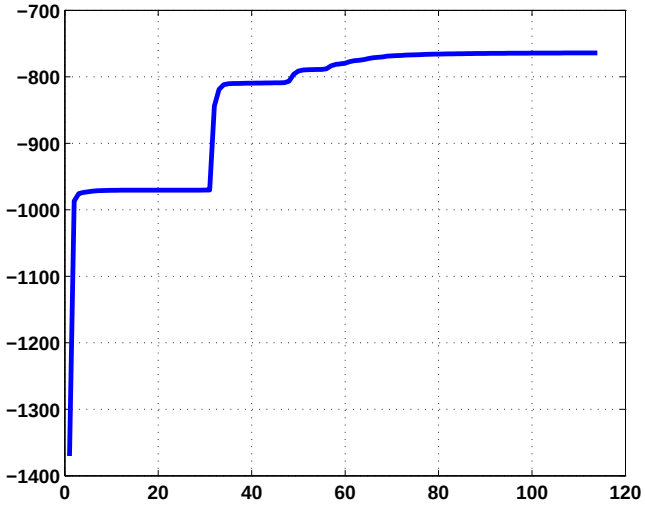


Figure 4-7. The lower bound of the VMEC training. A sudden increase in the lower bound corresponds to an update of the hyper-parameters.

require more experts, automated approaches as in [5, 8, 100, 101] can be investigated. Also, the comparisons were performed on networks of depth one, hence an immediate extension of this work would be to investigate the hierarchical VMEC. On the other hand, both VMEC and ME models work on static features, and cannot handle time-series data as features. For example in landmine detection, the confidence values and the features that were used in the GRANMA and EHD classifiers were obtained with the assumption that the robot was already on the target, and the target was in the center. This assumption may work fine for grid based landmine detection efforts, but it does not use the temporal structure that is found in lane-based landmine detection data. To incorporate this temporal structure into an ME model, a new model, named as the mixture of hidden Markov model experts is introduced in Chapter 6.

CHAPTER 5
FUNDAMENTALS OF HIDDEN MARKOV MODELS (HMM)

Hidden Markov models (HMMs) are major tools to analyse time-series data, and they have been extensively used in a variety of applications, including, but not limited to, speech recognition [102, 103], landmine detection [104–106], handwriting recognition [107], behavior detection [108], face recognition [109], music analysis [42], crash detection [110], and understanding genetic expression [111]. The literature on HMMs is immense, so in this chapter only the fundamentals of HMMs that relate to this study are given. Throughout this chapter, the notation by Rabiner et al. [103] will be used, where an HMM is characterized by the following:

- W = number of states
- M = number of observation symbols
- T = length of observation sequence
- $V = \{v_1, ..., v_M\}$, the discrete set of possible observation symbols
- $O = O_1 O_2 ... O_T$ denotes an observation sequence, where O_t is the observation at time t, and it is one of the observation symbols V.
- $Q = q_1 q_2 ... q_T$, a fixed state sequence
- $S = \{S_1, S_2, ..., S_N\}$, the individual states
- q_t = the state at time t
- The initial state distribution $\pi = \{\pi_i\}_{i=1}^{W}$, where $\pi_i = P(q_1 = S_i)$ is the probability of being in state i at time $t = 1$.
- The state transition probability $A = \{\{a_{ij}\}_{i=1}^{W}\}_{j=1}^{W}$, where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ is the probability of being in state j at time $t + 1$ given that we are in state i at time t.
- The observation symbol probability distribution $B = \{\{b_j(m)\}_{j=1}^{W}\}_{m=1}^{M}$, where $b_j(m) = P(v_m \texttt{ at } t | q_t = j)$ is the probability of observing the symbol v_m given that we are in state j.
- $\lambda = (A, B, \pi)$ is the compact notation for the complete parameter set of an HMM.

There are three main problems related to HMMs:

1. Given a model, what is the probability that the observation sequence was generated from this model $P(O|\lambda)$?
2. Given a model, what is the best state sequence i.e. the best path Q to follow that maximizes $P(Q|O,\lambda)$?
3. How do you find the model that best describes the observation sequences?

The answers to the first two questions are rather standard. The first problem is efficiently solved using a forward-backward algorithm, while the second can be solved using the Viterbi algorithm. The answer to the third question has received much investigation, but the most popular algorithms are the segmental K-means [112] and the Baum-Welch (BW) [103] re-estimation formulas. These two algorithms are generative models, and they are applied in an attempt to increase the probability of a sequence given a model. Additionally, it has been observed that generative models are not powerful enough to distinguish between some sequences, so discriminative models have been developed in [113]. In the rest of this chapter, the fundamentals of BW learning are given in section 5.3.1 and minimum classification error (MCE) based discriminative learning is briefly explained in section 5.3.2. Finally in section 5.4, a literature review explains the studies that use HMMs to find multiple models from time-series data.

5.1 Evaluation

The straightforward way to compute $P(O|\lambda)$ is to find $P(O|Q,\lambda)$ and $P(Q|\lambda)$; and sum their multiplication over all possible state sequences, where

$$P(O|Q,\lambda) = b_{q_1}(O_1)b_{q_2}(O_2)...b_{q_T}(O_T), \tag{5–1}$$

$$P(Q|\lambda) = \pi_{q_1}a_{q_1q_2}a_{q_2q_3}...a_{q_{T-1}q_T}, \tag{5–2}$$

and

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)P(Q|\lambda), \tag{5–3}$$

$$= \sum_{Q} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T), \tag{5–4}$$

$$= \sum_{Q} \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} b_{q_t}(O_t), \tag{5–5}$$

which is the standard HMM model. To reduce the number of multiplications in computing this model, forward-backward algorithms are generally used to inductively solve the summation.

5.2 Decoding

The decoding finds the optimal state sequence associated with the given observation sequence. Of the many criteria that define what is best, the most widely used criterion is to find the single best path that maximizes $P(Q|O,\lambda)$, which is equivalent to maximizing $P(Q,O|\lambda)$ [103]. The Viterbi algorithm [114] inductively keeps the best state sequence for each of the N states as the intermediate state for the observation sequence. Hence, it gives the best paths for the N states and single outs the path with the highest probability.

5.3 Reestimation

Reestimation algorithms can be divided into two groups: the generative and discriminative training approaches. In generative training, Viterbi [114], BW [103], and segmental K [112] are the most commonly used algorithms. In areas of DNA or protein-sequence modelling, where Viterbi paths play an important role, Viterbi learning may be more desirable. If all possible paths are to be considered, BW is preferred. Recently, variational training of HMMs is becoming more common [115–117] which has a Bayesian way of learning an approximate posterior distribution. Conversely, discriminative approaches [18, 113, 118, 119] are based on separating the two (or more) classes via the utilization of such methods as minimization of classification error.

5.3.1 Baum-Welch (BW) Training

Maximum likelihood estimation maximizes $P(O|\lambda)$ over all the parameters λ for a given observation sequence O. BW, also referred to as the expectation maximization (EM) algorithm, iterates over the E and the M steps. In the E step, the EM algorithm calculates the expectation of unknown (hidden) data given a probabilistic model. In the M step, it computes the maximum likelihood estimate of the complete data, where the complete data is both the data and the expectation of the hidden states.

BW is the most widely used algorithm to learn the HMM parameters, provides a good generative solution and it is guaranteed to converge to a local maximum [120], but it is sensitive to initialization, and may not find the global solution.

5.3.2 Minimum Classification Error (MCE) Training

The MCE model was introduced in [113] and later extended to MCE-HMMs in [121], where HMMs are the discriminative classifiers. In MCE-HMM, the total misclassification error is written in terms of an empirical loss function, and the discrete HMM (DHMM) parameters minimizing this loss are learned using a gradient descent approach.

Let $\Lambda = \{\lambda\}_{k=1}^{K}$ be the set of parameters of all classes. Given a set of training observation sequences $\{O^{(n)}\}$, $n = 1, 2, \ldots N$, the empirical loss function is defined as:

$$L(\Lambda) = \sum_{n=1}^{N} \sum_{k=1}^{K} l_k(O^{(n)}; \Lambda) \mathbf{I}(O^{(n)} \in K_k). \tag{5–6}$$

When there are enough training sequences, the empirical loss is an estimate of the expected loss. Minimizing the empirical loss is equivalent to minimizing the total misclassification error, and DHMM parameters are estimated by a gradient descent on $L(\Lambda)$.

$l_k(O; \Lambda)$ is a sigmoid function with misclassification corresponding to the loss values in $(0.5, 1]$, computed as

$$l_k(O; \Lambda) = \frac{1}{1 + \exp(-\zeta d_k(O) + \theta)}, \tag{5–7}$$

where d is a misclassification measure of the sequence O. If d_k is much less than 0, meaning correct classification, no loss is incurred. When d_k is positive, the classification error count becomes the penalty. Generally, θ is set to 0 and ζ is set to greater than 1. The misclassification measure of a sequence for the kth class is computed as

$$d_k(O) = -g_k(O, \Lambda) + \log \left[\frac{1}{1-K} \sum_{j, j \neq k} \exp(\eta g_j(O, \Lambda)) \right]^{\frac{1}{\eta}}, \tag{5–8}$$

where η is a positive number and $g_k(O, \Lambda)$ is the log of the likelihood found by the Viterbi algorithm. When η approaches ∞, the term in the bracket becomes $\max_{j, j \neq k} g_j(X; \Lambda)$. So $d_k(X) > 0$ implies misclassification and $d_k(X) \leq 0$ means correct decision. The log-likelihood of the Viterbi algorithm is found by

$$g_k(O, \Lambda) = \log[\max_Q g_k(O, Q, \Lambda)], \tag{5–9}$$

where

$$g_k(O, Q, \Lambda) = P(O, Q; \Lambda_k) = \pi^{(k)}_{q_0} \prod_{t=1}^{T-1} a^{(k)}_{q_t q_{t+1}} \prod_{t=1}^{T} b^{(k)}_{q_t}(o_t) \tag{5–10}$$

With the empirical loss as defined in 5–6; the parameters are trained with a gradient descent approach.

5.4 Finding Multiple Models from Time-Series Data

Finding multiple models in time-series data dates back to the late 1980s [122] and has found applications in both clustering and classification problems including surveillance [123, 124], fMRI analysis [125], seismology [126], speech synthesis [127], speaker adaptation [128, 129], speaker clustering [130], handwriting recognition [131], and object classification [132]. The survey in [133] lists many of the online datasets and [134] provides a comparison of the algorithms in the data mining area. These approaches learn and generate multiple models within a group of data, and they are often unsupervised. Some of the algorithms listed below, find multiple models from

each class of data and test a given time-series sequence on all the models from all the classes for classification. If the output information is not explicitly used in the algorithm, we will refer to these approaches as clustering algorithms.

Most of the time-series clustering algorithms convert time-series data to static data using HMMs and employ the traditional clustering algorithms, such as K-means or hierarchical agglomerative clustering, on this static data. One of the earliest HMM clustering papers is by Rabiner et al. [122], where an HMM model was trained on a training set, all the sequences with poor likelihood scores were separated out, and new models were learned for the outliers. The recognition rates were significantly enhanced if a structure was found from the outliers, but the performance was very sensitive to the threshold for separating the training set into clusters.

A similar approach was taken by Li et al. [135] and Szamonek et al. [136], where an HMM model was fit to all the data and the number of clusters as well as the number of states were gradually increased based on the objects with the least likelihoods and the Bayesian information criterion (BIC). Since these algorithms considered splitting every model at each iteration, redistributing the objects and relearning the HMMs, they required a lot of training in very small steps.

A faster way than the top-down approach of splitting the models is the bottom-up approach of fitting an HMM model to each and every sequence, and clustering these sequences based on the similarities of their likelihoods or HMM models. Perhaps the most popular of the earlier clustering solutions was developed by Smyth [137] where an HMM model was fit to each training sequence, resulting in N HMM models for N sequences. Then all the N training sequences were tested on all the N HMM models, and the log-likelihoods were stored in a similarity matrix, as shown in Fig. 5-1. Using this similarity matrix, the data sequences with similar likelihoods were clustered together into K groups using hierarchical clustering, as in Fig. 5-2 and a new set of K HMMs were then trained on these clusters using the BW updates [103]. In the same year, a similar

approach was developed by Korkmazskiy et al. [138] with K-means clustering on the log-likelihoods, but the final HMMs were learned with a discriminative HMM model [121] rather than the BW. In 1999, Oates et al. [139] constructed the similarity matrix from the dynamic time warping (DTW) values instead of the log-likelihoods, then clustered it with hierarchical agglomerative clustering and selected a prototype from each cluster. An initial HMM was fit to the prototype sequence, and all the other sequences were introduced one by one into the cluster with the retraining of the HMMs.

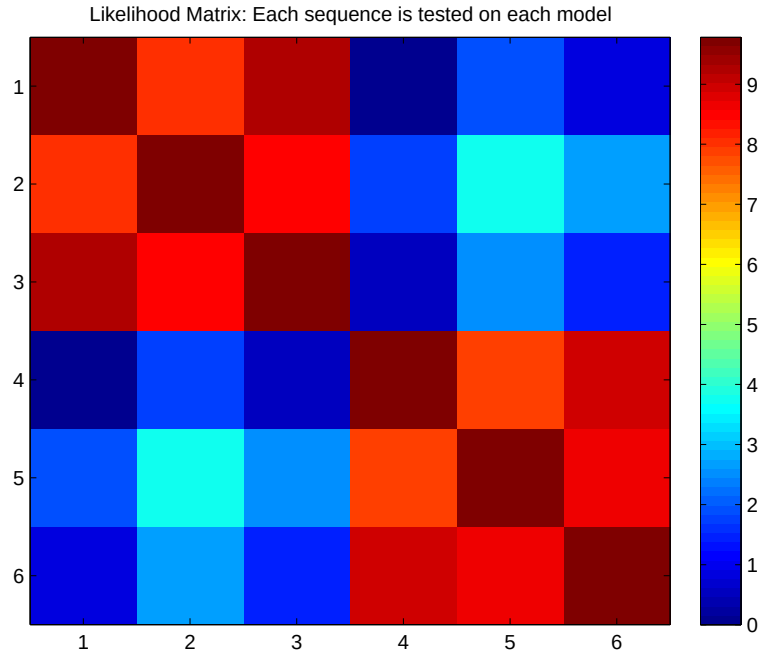


Figure 5-1. Symmetric likelihood matrix. One HMM is trained on one sequence, resulting in as many HMM models as the number of training sequences. Then each sequence is tested against all the HMMs, and the symmetrized likelihoods are stored as a matrix.

The previous approaches in [135, 137, 139] used hard clustering, where each sequence was assigned to a single cluster, and an HMM was trained only on those sequences in a cluster. In 2003, Alon et al. [123] introduced a soft clustering approach, where the expectations of cluster memberships were updated in an EM-type of learning. Although criticized in [140] for making strong assumptions about the shape of the

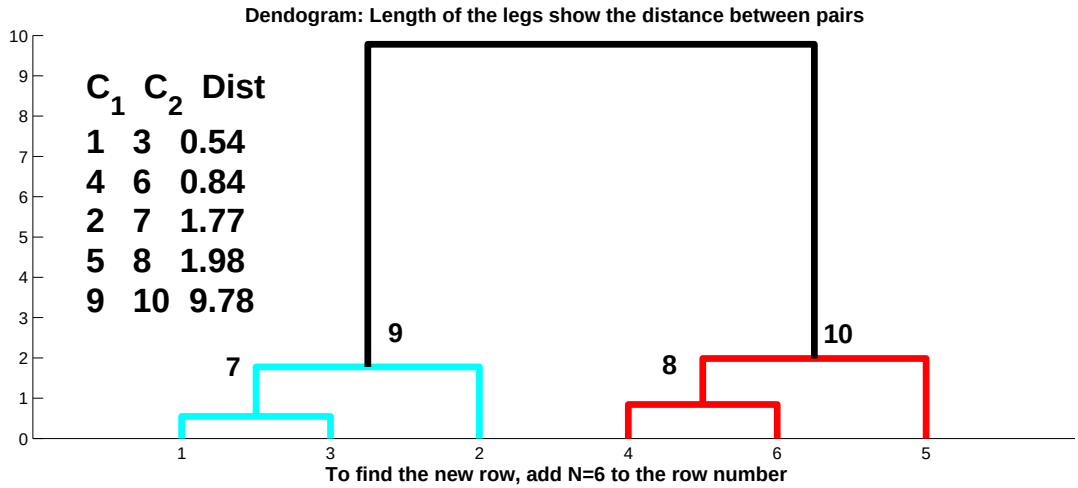


Figure 5-2. Hierarchical clustering of likelihoods.

overall distributions, this approach allowed for the simultaneous learning of the cluster and the mixture parameters. In 2006, for very large amounts of data, Tsoi et al. [141] proposed using Gaussian mixture models to group the data into clusters and trained the HMMs on randomly selected subsets of data from each cluster. More recently, in order not to make assumptions on the shape of the clusters, spectral approaches were used in [140, 142]. In [142], the similarity matrix was obtained by modelling each sequence as a hidden Markov model, just as in [137], but spectral methods were applied to this similarity matrix for dimensionality reduction. With this approach, the significant eigenvectors of the affinity matrix were used to map the original samples into a lower dimensional space. These vectors were then clustered by the K-means clustering algorithm. Similarly Porikli et al. [143] defines an affinity matrix A whose elements a_{ij} are equal to $a_{ij} = e^{-d(s_i,s_j)/2\sigma^2}$, where s_i and s_j are two sequences, and the distance is computed from $d(i,j) = |p_{ij} + p_{ji} - p_{ii} - p_{jj}|$, where p_{ij} refers to testing sequence s_i on the model trained from sequence s_j. Then the affinity matrix is decomposed into its eigenvectors, and clusters are formed using connected component analysis. Instead of the log-likelihoods, in [140], a probability product kernel (PPK) is used to find the Gram matrix which is followed by spectral methods, as in [142]. The PPK integrates

over the entire space of all possible samples given an HMM model, and may capture the properties of the data better.

A study closely related to our algorithm has been published by Bicego et al. [132], who found HMM to be very effective in dealing with object occlusions and used them for object classification in [25, 144, 145]. They also trained one HMM per sequence, but instead of clustering the similarity matrix, they treated each row of the un-symmetrized similarity matrix as a feature vector and used these feature vectors in standard classification techniques as follows:

1. Given a set of sequences $T = \{O_1 ... O_N\}$ to be clustered, let $R = \{P_1, ..., P_R\}$ be a set of R representative objects where $R \subseteq T$.

2. Train one HMM λ_r for each sequence $P_r \in R$.

3. Represent each sequence O_i by a feature vector $D_R(O_i) = \frac{1}{L_i} \begin{bmatrix} \log P(O_i|\lambda_1) \\ \log P(O_i|\lambda_2) \\ \vdots \\ \log P(O_i|\lambda_R) \end{bmatrix}$

   where L_i is the length of the sequence O_i.

4. Perform clustering in Euclidean space using any clustering technique (hierarchical agglomerative complete link, K-means, etc.)

The main drawback of this approach is the high dimensionality of the resulting feature space, which is equal to the cardinality of the data set, so dimensionality reduction was handled in [25] using principal component analysis (PCA) and Fisher discriminant analysis (FDA), and also using matching pursuits to select a set of representatives.

In a very recent study, Garcia et al. [146] normalize the likelihood matrix so that each column adds up to one. Hence, they view these columns as the probability density functions over the approximated model space conditioned on each of the individual sequences. In doing so, they find the distance between two sequences from the KL-divergence between the two discrete probability density functions.

Although it has not been presented as clustering research, one close relative of our study would be relative density nets (RDN) by Brown et al. [147]. RDN is a

discriminative learning by a network of HMM models in which many small HMMs are placed into the input layer of a feed-forward network. The hidden layer of the network performs all pairwise comparisons between the HMMs, and the weighted sum of these pairwise comparisons are passed through a softmax function to make the final decision. Also, an alternative to RDNs was introduced in [148]. It is a generative algorithm formed from the products of HMMs (PoHMM). The algorithm trains the HMMs independently using BW, but uses gradient based minimization of contrastive divergence to calculate the log-likelihood of the PoHMM. Although the training of the HMMs is straightforward, the log-likelihood of the overall architecture requires the use of Gibbs sampling methods and ends up being computationally very costly.

From a different perspective, clustering can be done to find the different segments of a sequence. For a music recommendation system, Qi et al. [42] consider a mixture of HMMs approach, named as the DP-HMM, in which the mixture coefficient is based on a Dirichlet process (DP) prior. The DP-HMM is used to analyse the characteristics of different segments of a given piece of music. The number of HMMs in the mixture is decided by the DP prior, and the learning of the HMM parameters is achieved using variational methods. To compute the similarity between the HMM mixture models, samples are generated from each model using Monte Carlo sampling. For a sample set S_g from the mixture model M_g for music g, and a sample set S_h from the mixture model M_h for music h, the distance between two HMM mixture models is defined as:

$$D(M_g, M_h) = \frac{1}{2}\left[\log p(S_h|M_g) - \log p(S_h|M_h)\right] + \frac{1}{2}\left[\log p(S_g|M_h) - \log p(S_g|M_g)\right]$$

There are also other studies that define the distance between HMM models, based on state-observation probability matrices for discrete HMM [149, 150] and for a continous HMM [151], for ergodic HMMs [152, 153], for left-to-right HMM models [154], based on KL-divergence [155–160], based on the probability product kernel [161], based

on the stationary cumulative distribution [162]. A comparative study was published by Mohammad et al. [163]. These studies use the parameters of the HMM, hence assume the existence of a true HMM model. Due to this assumption, such distances did not find a lot of interest in the clustering of time-sequences, for which the HMM model is unknown. However, it should be possible to incorporate these distances into clustering, in addition to comparing the likelihoods.

Among these models, we initialize our model in the next chapter with [137], however, any of [123, 135, 139, 140, 142, 143] could also be used. We also compare our results with the results reported in [25, 144, 145].

CHAPTER 6
MIXTURE OF HIDDEN MARKOV MODEL EXPERTS (MHMME)

Time series or sequential data often show multiple patterns owing to the different contexts that they appear in. For example, electricity usage has seasonal patterns. In addition, within each season, electricity consumption of socio-economic groups show different patterns. Another example is electrocardiogram (EKG) heartbeat classification. Athletes typically have slow resting heartbeats compared to non-athletes, and without this information an athlete might be considered to have an unhealthy beat. Therefore, the socio-economic group and even the ethnicity of people define a context in classifying a healthy vs an unhealthy beat. Unfortunately, unlike these examples, contexts are generally hard to define, they are often interlaced, and do not have sharp boundaries. Moreover, context information might be inherent in the data, but not be known to the data modeler. In such cases, we define a context as a group of similar signatures.

In this chapter, we are introducing a novel model, mixture of hidden Markov model experts (MHMME), that can both decompose time series data into multiple contexts, and learn expert classifiers for each context. In this model, a gate of hidden Markov models defines the contexts, and cooperate with a set of hidden Markov model experts which provide multi-class classification.

The main advantages of our model can be summarized as follows:

- MHMME model is suitable for time series data and sequential data of varying lengths due to the use of the hidden Markov models.
- MHMME model provides a divide and conquer approach, is probabilistic, and has soft boundaries - all of which make it very suitable for context learning.
- There is no hard clustering of the data so sequences can freely move between contexts and classifiers during training. Moreover, the final decision is a mixture of the decisions of all the classifiers.
- Learning of the contexts and classifiers is accomplished simultaneously, in one model.
- MHMME is suitable for multi-class classification.

The MHHME model extends the ME model to be suitable for time series data by the use of HMMs. Thus the learning algorithm is completely modified, but the essence of MEs is retained. While the HMMs at the gates make soft partitions of the input space and define the regions where the expert opinions are trustworthy, the HMMs at the experts specialize in finding models that would discriminate among these data. Hence, multiple models are learned from the data and the detection rates are increased by the competition of the experts to claim regions of the input space. There is no restriction on the dimension of the data and varying length sequences can be handled.

In this study, our goal is to increase classification rates by training discriminative models for the sequences of different classes that would be in the same context because of their highly similar structures. Our MHMME algorithm can learn these contexts and classification parameters in one framework, and adjusts itself naturally during the iterations. Although the idea is similar to the ME, the gate and experts are now HMMs, and the learning requires the learning of the HMM parameters instead of the linear weights. The HMM models at each expert can be of different structures (number of states, etc.) and the algorithm can easily be extended to be a hierarchical mixture of HMM experts.

In Sec. 6.1, we compare the MHMME model to some of the related models that have been reviewed in Sec. 3.3 and Sec. 5.4. Then, in Sec. 6.2, we derive the update equations that combine the ME architecture with hidden Markov model updates, and provide the training algorithm for the simultaneous probabilistic learning of the contexts from multi-class data and the discriminative classification of the data in these contexts.

6.1 How ME Compares to Other Models

ME finds multiple sub-regions within the data and trains the experts to specialize in these regions. Unfortunately, ME models cannot operate on time series data. In the literature review in Sec. 3.3, a number of models [9, 10, 12, 164] were described that extend the ME architecture to make it suitable for time series data. These models

are only applicable to regression and they use a one-step-ahead or multi-step-ahead prediction in which the last d values of the time series data are used as features in a neural network. Such models cannot handle data of varying length and the use of multilayer network-type approaches prevent them from completely describing the temporal properties of a time series dataset. Additionally, the studies in the ME literature that combine HMMs and HMEs [13, 84–87] are handling quite different problems despite the similar names. These models were covered in Sec. 3.3. Briefly, in [84, 86], each state of the HMM is an ME, whereas in our model, each HMM is a part of an expert. In [34] one can think of the Hidden Conditional Random Fields as the gate and the SVMs as the experts, but temporal sequences were not of interest.

Due to the soft-partitioning at the gates, the MHMME model might be compared to time series clustering algorithms. Most of the time series clustering approaches reviewed in Sec. 5.4 learn one HMM per sequence to construct a similarity matrix. However, such approaches start with the assumption that fitting one HMM to one sequence would give reliable results. This may not always be the case if the sequence is not sufficiently long enough or if it does not have a repeating pattern. For example, in 2D shape recognition from boundaries, the sequences are of varying lengths, shorter, and not self repeating. Moreover, those algorithms learn HMM models that are of the same HMM topology (same number of states, number of symbols, and the like).

Our MHMME model does not require to fit a single HMM model to a single sequence. Instead, it uses all the data available and gives a weight to each sequence by the use of the HMMs at the gate. Then the experts make a classification decision and learn which HMMs are experts for this weighted data. In addition, because of its modular structure, MHMME can handle experts with different HMM topologies. Moreover, our model does not do a hard clustering at the gate, therefore, during learning, the data can change its probability and start contributing to a different gate or expert.

In comparison to neural network type studies such as [147], the differences are similar to the main differences between neural networks and ME, such as (1) our gate is obtained from the HMMs, as opposed to being weight vectors; (2) our gate is input-dependent, whereas the weights are fixed in [147] once they are learned; (3) our experts provide a discriminative learning between the HMMs and (4) the experts provide visually and statistically interpretable results. The contexts found at the gates produce sensible outputs.

6.2 Mixture of Hidden Markov Model Experts

For all the hidden Markov models, we define:

- W = number of states.
- M = number of symbols in the codebook.
- T = length of observation sequence.
- $V = \{v_1, ..., v_M\}$ the discrete set of observation symbols.
- $O = O_1 O_2 ... O_T$ denotes an observation sequence, where O_t is the observation at time t.
- $Q = q_1 q_2 ... q_T$ is a fixed state sequence, where q_t is the state at time t .
- $S = \{S_1, S_2, ..., S_W\}$ are the individual states.
- λ_{ik} = HMM model for the k^{th} class at the i^{th} expert.
- ψ_i = i^{th} HMM model at the gate.
- I = number of experts, and $i : 1...I$ is the expert index.
- K = number of classes, and $k : 1...K$ is the class index.

The MHMME architecture is illustrated in Fig. 6-1. In this architecture, the gate has I HMM models, and its function is to make soft partitions in the HMM space, and define the contexts where the individual expert opinions are trustworthy. Each branch of the gate is connected to an expert, and an expert has K HMMs, one for each class. We denote the HMM models at the gate with $\Psi = \{\psi\}_{i=1}^{I}$, the HMM models at the experts

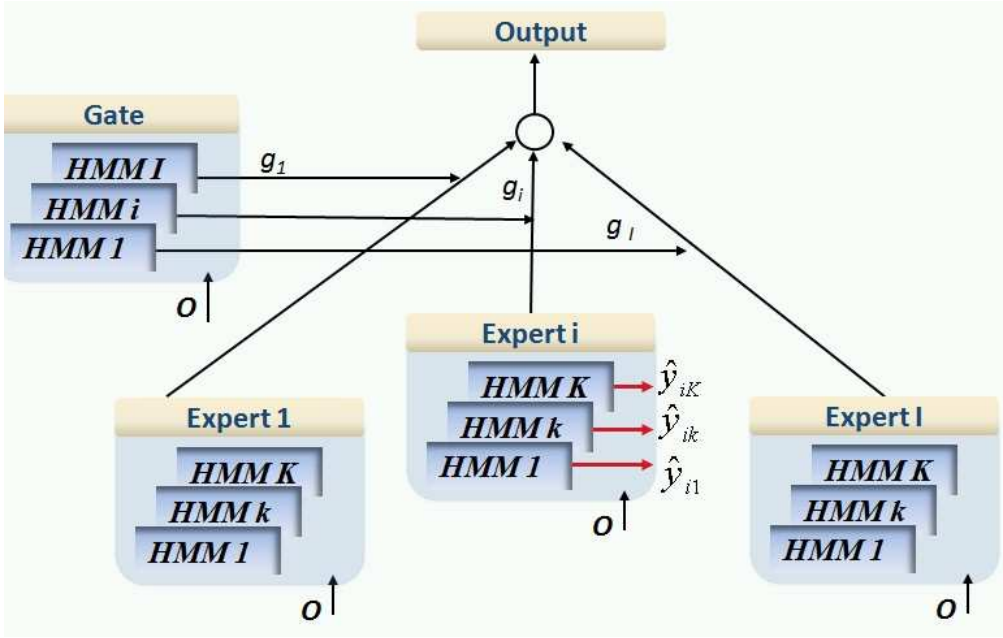


Figure 6-1. MHMME architecture. A gate makes partitions in the HMM-space, and the experts learn to discriminate the classes in these partitions.

with $\Lambda_i = \{\lambda_{ik}\}_{k=1}^{K}$, and finally, we denote the set of all the gate and expert parameters as $\Theta = \{\Psi, \Lambda\}$.

Let the data be denoted by $D = \{\mathbf{O}, Y\}$, where $\mathbf{O} = \{O^{(n)}\}_{n=1}^{N}$ represents the input time series sequences, and $Y = \{\mathbf{y}^{(n)}\}_{n=1}^{N}$ represents class coded true outputs of training data such that

$$y_k^{(n)} = \begin{cases} 1 & x^{(n)} \text{ belongs to class } k \text{ ;} \\ 0 & \text{otherwise.} \end{cases}$$

The gate and experts make a decision for a multi-class classification problem following the probability model:

$$P(D;\Theta) = \prod_{n=1}^{N} \sum_{i=1}^{I} g_i(O^{(n)}, \Psi_i) \;\; P_i(\mathbf{y}^{(n)}, \Lambda_i). \tag{6–1}$$

where $g_i(O^{(n)}, \Psi_i)$ is the gate's probabilistic estimate that the sequence $O^{(n)}$ belongs to the HMM-space that is defined by expert i. In other words, $g_i(O^{(n)}, \Psi_i) = P(i|O^{(n)}, \Psi_i)$, the probability of selecting the i^{th} expert given the sequence $O^{(n)}$. The second term in Eq. 6–1, $P_i(\mathbf{y}^{(n)}, \Lambda_i)$ is the probability that the i^{th} expert has generated $\mathbf{y}^{(n)}$ given the sequence $O^{(n)}$. In the rest of this paper, we will denote $g_i(O^{(n)}, \Psi_i)$ with $g_i^{(n)}$, and $P_i(\mathbf{y}^{(n)}, \Lambda_i)$ with $P_i(\mathbf{y}^{(n)})$.

The gate's probabilistic estimate is obtained by a softmax function that considers the confidences of all the HMM models at the gate, given as:

$$g_i^{(n)} = \frac{\exp f(O^{(n)}|\psi_i)}{\sum_{m=1}^{I} \exp f(O^{(n)}|\psi_m)} \tag{6–2}$$

where $f(O^{(n)}|\psi_i)$ is the Viterbi log-likelihood of observation $O^{(n)}$ for an HMM model ψ_i.

Similar to the gate, the HMMs at the experts compute the Viterbi log-likelihood

$$f(O^{(n)}|\lambda_{ik}) = \log P_{HMM}(O^{(n)}, Q, \lambda_{ik}), \tag{6–3}$$

where the Viterbi likelihood $P_{HMM}(O, Q, \lambda_{ik})$ is

$$P_{HMM}(O, Q, \lambda_{ik}) = \pi_{q_0}^{(ik)} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}^{(ik)} \prod_{t=1}^{T} b_{q_t}^{(ik)}(o_t). \tag{6–4}$$

These log-likelihoods are converted to probabilities by a softmax function, and the output of expert i for class k is $\hat{y}_{ik}^{(n)}$ computed as:

$$\hat{y}_{ik}^{(n)} = \frac{\exp f(O^{(n)}|\lambda_{ik})}{\sum_{r=1}^{K} \exp f(O^{(n)}|\lambda_{ir})}. \tag{6–5}$$

which is also the mean of its multinomial probability model, i.e., for a given sequence $O^{(n)}$, expert i produces a prediction with probability $P_i(\mathbf{y}^{(n)})$ following a multinomial distribution with mean $\hat{y}_{ik}$ such that:

$$P_i(\mathbf{y}^{(n)}) = \prod_k \hat{y}_{ik}^{y_k}. \tag{6–6}$$

Finally, the output of the MHMME architecture, $\{\hat{y}_k\}_{k=1}^{K}$, is a weighted sum of the expert outputs:

$$\hat{y}_k^{(n)} = \sum_i g_i^{(n)} \hat{y}_{ik}^{(n)}. \tag{6–7}$$

It is worthy to notice the relationship between the mixture model Eq. 6–7 and its probabilistic counterpart in Eq. 6–1. The parameters of the MHMME model are learned using the probabilistic model in Eq. 6–1, which will be explained in the next section. Typically, the k^{th} class that gives the maximum $\{\hat{y}_k\}_{k=1}^{K}$ is selected as the final decision; i.e., the class of the sequence $O^{(n)}$.

6.2.1 Training of the MHMME model

The marginal distribution $P(D;\Theta)$ in Eq. 6–1 can be solved much more simply by introducing latent variables Z, and by solving the equivalent formulation $P(D,Z;\Theta)$ with the expectation-maximization (EM) algorithm. These latent variables are $Z = \{\{z_i^{(n)}\}_{n=1}^{N}\}_{i=1}^{I}$ such that

$$z_i^{(n)} = \begin{cases} 1 & \text{if } O^{(n)} \in R_i; \\ 0 & \text{otherwise.} \end{cases}$$

where R_i is the region specified by expert i. Hence, the complete data distribution becomes:

$$P(D,Z;\Theta) = \prod_n \prod_i \left(g_i^{(n)} P_i(y^{(n)}) \right)^{z_i^{(n)}}, \tag{6–8}$$

So now, in the E step, we find the expectations of the hidden variables [2, 26] as:

$$h_i^{(n)} = g_i^{(n)} P_i(\mathbf{y}^{(n)}) / (\sum_j g_j^{(n)} P_j(\mathbf{y}^{(n)})). \tag{6–9}$$

In the M step, we maximize the expected complete data likelihood $E_Z(\log P(D,Z;\Theta))$, which gives the objective function

$$Q(\Theta, \Theta^{(p)}) = E_Z(l(\Theta, D, Z))$$
$$= \sum_{n=1}^{N}\sum_{i=1}^{I} h_i^{(n)} \log g_i^{(n)} + \sum_{n=1}^{N}\sum_{i=1}^{I} h_i^{(n)} \log P_i(\mathbf{y}^{(n)}) \ . \quad (6\text{–}10)$$

In the M step, h_is are kept fixed, so the two terms on the right side of the equation are decoupled, and can be computed independently for the experts and the gates. We refer to these objective functions as Q_g for the gate, and as Q_e for the experts, given as:

$$Q_g = \sum_{n=1}^{N}\sum_{i=1}^{I} h_i^{(n)} \log g_i^{(n)} \quad (6\text{–}11)$$

$$Q_e = \sum_{n=1}^{N}\sum_{i=1}^{I} h_i^{(n)} \log P_i(\mathbf{y}^{(n)}) \quad (6\text{–}12)$$

In the M step, we search for the HMM parameters that maximize these objective functions:

$$\lambda_{ik}^{(p+1)} = \underset{\lambda_{ik}}{\text{argmax}} \ Q_e \ , \quad (6\text{–}13)$$
$$\psi_i^{(p+1)} = \underset{\psi_i}{\text{argmax}} \ Q_g \ . \quad (6\text{–}14)$$

Explicitly, the HMM parameters to be estimated in the experts are $\lambda_{ik} = \{A^{(ik)}, B^{(ik)}\}$. We will denote each element of the A matrix as $a_{rj}^{(ik)}$ with $r = 1...W$, $j = 1...W$, and each element of the B matrix as $b_{mj}^{(ik)}$ with $m = 1...M, j = 1...W$. To ensure that the estimated parameters satisfy the constraints $a_{rj} \geq 0$, $\sum_{j=1}^{W} a_{rj} = 1$, $b_{mj} \geq 0$, and $\sum_{m=1}^{M} b_{mj} = 1$, we map these parameters using $\log$, and map them back with softmax functions:

$$a_{rj} \rightarrow \tilde{a}_{rj} = \log a_{rj} \ , \tag{6–15}$$

$$a_{rj} = \frac{\exp \tilde{a}_{rj}}{\sum_{j'=1}^{W} \exp \tilde{a}_{rj'}} \ , \tag{6–16}$$

$$b_{mj} \rightarrow \tilde{b}_{mj} = \log b_{mj} \ , \tag{6–17}$$

$$b_{mj} = \frac{\exp \tilde{b}_{mj}}{\sum_{m'=1}^{M} \exp \tilde{b}_{m'j}} \ . \tag{6–18}$$

Such mappings are common in gradient based training models such as [121, 165].

Let p denote the iteration number. The HMM parameters that maximize the objective functions are found by gradient ascent updates as:

$$\tilde{a}_{rj}^{(ik)}(p+1) = \tilde{a}_{rj}^{(ik)}(p) + \epsilon \frac{\partial Q_e(\Lambda(p))}{\partial \tilde{a}_{rj}^{(ik)}(p)}, \tag{6–19}$$

$$\tilde{b}_{mj}^{(ik)}(p+1) = \tilde{b}_{mj}^{(ik)}(p) + \epsilon \frac{\partial Q_e(\Lambda(p))}{\partial \tilde{b}_{mj}^{(ik)}(p)}, \tag{6–20}$$

where

$$\frac{\partial Q_e(\Lambda)}{\partial \tilde{a}_{rj}^{(ik)}} = \sum_{n=1}^{N} \sum_{m=1}^{K} h_i^{(n)} \left(y_k^{(n)} - \hat{y}_{ik}^{(n)} \right) \frac{\partial f(O^{(n)}, \Lambda_{ik})}{\partial a_{rj}^{(ik)}} \frac{\partial a_{rj}^{(ik)}}{\partial \tilde{a}_{rj}^{(ik)}}, \tag{6–21}$$

$$\frac{\partial Q_e(\Lambda)}{\partial \tilde{b}_{mj}^{(ik)}} = \sum_{n=1}^{N} \sum_{m=1}^{K} h_i^{(n)} \left(y_k^{(n)} - \hat{y}_{ik}^{(n)} \right) \frac{\partial f(O^{(n)}, \Lambda_{ik})}{\partial b_{mj}^{(ik)}} \frac{\partial b_{mj}^{(ik)}}{\partial \tilde{b}_{mj}^{(ik)}}, \tag{6–22}$$

and the gradients are

$$\frac{\partial f(O^{(n)}, \lambda_{ik})}{\partial a_{rj}^{(ik)}} = \frac{1}{a_{rj}^{(ik)}} \sum_{t=1}^{T} \delta(q_t^{(n)} = m, q_{t+1}^{(n)} = j), \tag{6–23}$$

$$\frac{\partial b_{ij}^{(ik)}}{\partial \tilde{b}_{mj}^{(ik)}} = b_{mj}^{(ik)}[1 - b_{mj}^{(ik)}], \tag{6–24}$$

$$\frac{\partial f(O^{(n)}, \lambda_{ik})}{\partial b_{mj}^{(ik)}} = \frac{1}{b_{mj}^{(ik)}} \sum_{t=1}^{T} \delta(q_t^{(n)} = m, Q_V(O_t^{(n)}) = j). \tag{6–25}$$

Similarly, the updates for the gates are:

$$\frac{\partial Q_g(\Psi)}{\partial \tilde{a}_{rj}^{(i)}} = \sum_{n=1}^{N} \sum_{m=1}^{K} (h_i^{(n)} - g_i^{(n)}) \frac{\partial f(O^{(n)}, \psi_i)}{\partial a_{rj}^{(i)}} \frac{\partial a_{rj}^{(i)}}{\partial \tilde{a}_{rj}^{(i)}}, \tag{6–26}$$

$$\frac{\partial Q_g(\Psi)}{\partial \tilde{b}_{mj}^{(i)}} = \sum_{n=1}^{N} \sum_{m=1}^{K} (h_i^{(n)} - g_i^{(n)}) \frac{\partial f(O^{(n)}, \psi_i)}{\partial b_{mj}^{(i)}} \frac{\partial b_{mj}^{(i)}}{\partial \tilde{b}_{mj}^{(i)}}. \tag{6–27}$$

Hence, the algorithm computes the expectation of the hidden variables (h_i) in the E step, and learns the HMMs of both the gates and the experts in the M step. The complete algorithm is given below.

Algorithm 6.2.1: MHMME TRAINING(K, X, Y)

- Initialize the number of experts I
- Initialize the gating HMM parameters $\{\psi_i\}_{i=1}^{I}$
- Initialize the expert HMM parameters $\{\{\lambda_{ik}\}_{i=1}^{I}\}_{k=1}^{K}$

while $|(Q(\Theta, \Theta^{(p-1)}) - Q(\Theta, \Theta^{(p)}))|/Q(\Theta, \Theta^{(p)} > 1e-5$

do
- **comment: E STEP**
- **do** Compute
  - Viterbi log likelihoods $f(O|\lambda_{ik})$ from Eq. 6–3
  - Expert outputs $\hat{y}_{ik}^{(n)}$ from Eq. 6–5
  - Expert probabilities $P_i(\mathbf{y})$ from Eq. 6–6
  - Gating outputs $g_i^{(n)}$ from Eq. 6–2
  - Posterior probabilities $h_i^{(n)}$ from Eq. 6–9
- **end**
- **comment: M STEP**
- **comment: Expert Updates**
- **while** Q_e in Eq. 6–12 is increasing (i.e. $\Delta Q_e < 1e-5$), for each expert
- **do**
  - Map $a_{rj} \rightarrow \widetilde{a}_{rj}$ and $b_{mj} \rightarrow \widetilde{b}_{mj}$ (Eqs. 6–15 & 6–17)
  - Update A from Eq. 6–19
  - Update B from Eq. 6–20
  - Map $\widetilde{a}_{rj} \rightarrow a_{rj}$ and $\widetilde{b}_{mj} \rightarrow b_{mj}$ (Eqs. 6–16 & 6–18)
- **comment: Gate Updates**
- **while** Q_g in Eq. 6–11 is increasing (i.e. $\Delta Q_e < 1e-5$)
- **do**
  - Map $a_{rj} \rightarrow \widetilde{a}_{rj}$ and $b_{mj} \rightarrow \widetilde{b}_{mj}$ (Eqs. 6–15 & 6–17)
  - Update A using the gradients in Eq. 6–26
  - Update B using the gradients in Eq. 6–27
  - Map $\widetilde{a}_{rj} \rightarrow a_{rj}$ and $\widetilde{b}_{mj} \rightarrow b_{mj}$ (Eqs. 6–16 & 6–18)
- Compute $Q(\Theta, \Theta^{(p)})$ from Eq. 6–10

- Compute $\hat{y}_k^{(n)}$ from Eq. 6–7
- Make a decision $k^* = \underset{k}{\text{argmax}} \ \{\hat{y}_k^{(n)}\}_{k=1}^{K}$.

return (k^*)

A very important point is that, HMMs at each expert are affected by all the data points, but the effect of each data point is weighted by the gate. Therefore, even if a sequence does not have a high enough weight to guide the training of a classifier, it may still affect the training, albeit slightly. Therefore, HMMs can avoid over-training while they specialize for each context.

6.2.2 Two-Class Case

In this section, we show the learning for a single expert in a two-class problem. Remember that at each expert, we are trying to maximize Q_e:

$$Q_e = \sum_{n=1}^{N} \sum_{i=1}^{I} h_i^{(n)} \log P_i(\mathbf{y}^{(n)}).$$

At the M step, the h_is are fixed, so we assume them to be 1. The training for each expert is decoupled, so assume one of the experts is already picked (hence delete summation over i). Also we assume just two classes.

$$\begin{aligned}
Q &= \sum_n \log P_i(\mathbf{y}^{(n)}), \\
&= \sum_n \log \left(\prod_k \hat{y}_{ik}^{y_k} \right), \\
&= \sum_n \sum_k y_k \log \left(\hat{y}_{ik} \right), \\
&= \sum_n \sum_k y_k \log \left(\frac{\exp f(O^{(n)}|\lambda_{ik})}{\sum_{r=1}^{K} \exp f(O^{(n)}|\lambda_{ir})} \right), \\
&= \sum_n \sum_k y_k \log \left(\frac{1}{1 + \exp \left(f(O^{(n)}|\lambda_{i,r\neq k}) - f(O^{(n)}|\lambda_{i,r=k}) \right)} \right).
\end{aligned}$$

Remembering that the $f()$ terms are the log-likelihoods of an HMM and are negative numbers, the only way for Q to increase is if the first log-likelihood term is decreased, and the second log-likelihood term is increased. Thus, at each expert, the HMMs are adjusted such that the correct hypothesis is more probable, while the incorrect hypotheses is less probable.

6.3 Results on Synthetic Data

For an illustrative example, we generated 80 sequences from two classes as follows: the $[0, 1]$ domain was divided into 10 intervals, and the x values were uniformly sampled from each of these intervals. Then, for the first class, 20 sequences were generated from $y = x + N(0, \sigma^2)$ where $\sigma = 0.04$, and 20 sequences were generated from $y = -x + 1 + N(0, \sigma^2)$. This results in 40 training sequences (each of length 10) for the first class. Similarly, 40 training sequences were generated for the second class, 20 of which were generated from $y = x^2 + N(0, \sigma^2)$, and the remaining 20 were generated from $y = -x^2 + 1 + N(0, \sigma^2)$. These sequences are displayed in Fig. 6-2. Using the same parameters and the same protocol, a test set was generated that also had 80 sequences with 40 from each class.

The y values were used as the data, and the x values were ignored. Therefore, the features were just the 1D sequences as if the data are projected onto the y-axis. Then these values were were discretized to five symbols which are linearly spaced between $[0, 1]$. Hence, the classification is actually not as simple as it seems because the data were discretized to only five symbols, and to learn models that discriminate $y = x$ and $y = x^2$ without the time information is not trivial.

To discriminate these sequences, an MHMME model was trained with two experts. All HMMs in the MHMME model were fixed to have three states and five symbols. To initialize the gate, the data were clustered by k-means into two, and a Baum-Welch (BW) HMM was fit to each of these clusters. To initialize the experts, the sequences that were highly weighted by the gates were clustered using the HMM clustering approach in [137]. With this initialization, the first HMM in the gate immediately gave more weight to the $y = x$ and $y = x^2$ sequences, and the second HMM in the gate gave more weight to the $y = -x + 1$ and $y = -x^2 + 1$ sequences. Upon the MHMME training, the first expert learned HMM models that discriminate among $y = x$ and $y = x^2$, and the second

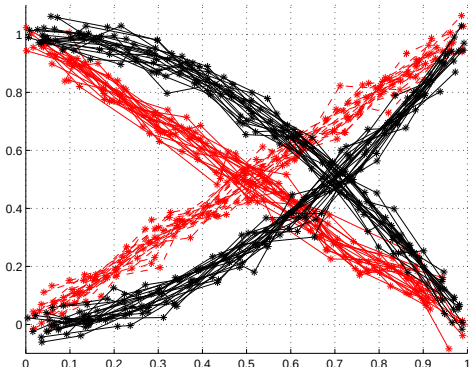


Figure 6-2. Synthetic data for two classes. The sequences that belong to the first class are displayed in red, and the sequences that belong to the second class are displayed in black. The first class has 20 sequences generated from the function $y = x + N(0, \sigma^2)$ overlayed on each other, and 20 sequences generated from $y = -x + 1 + N(0, \sigma^2)$, also overlayed on each other. Similarly, the second class has 20 sequences generated from the function $y = x^2 + N(0, \sigma^2)$, and 20 sequences generated from $y = -x^2 + 1 + N(0, \sigma^2)$.

expert learned HMM models that discriminate among $y = -x + 1$ and $y = -x^2 + 1$. The patterns learned by each HMM are shown in Fig. 6-3.

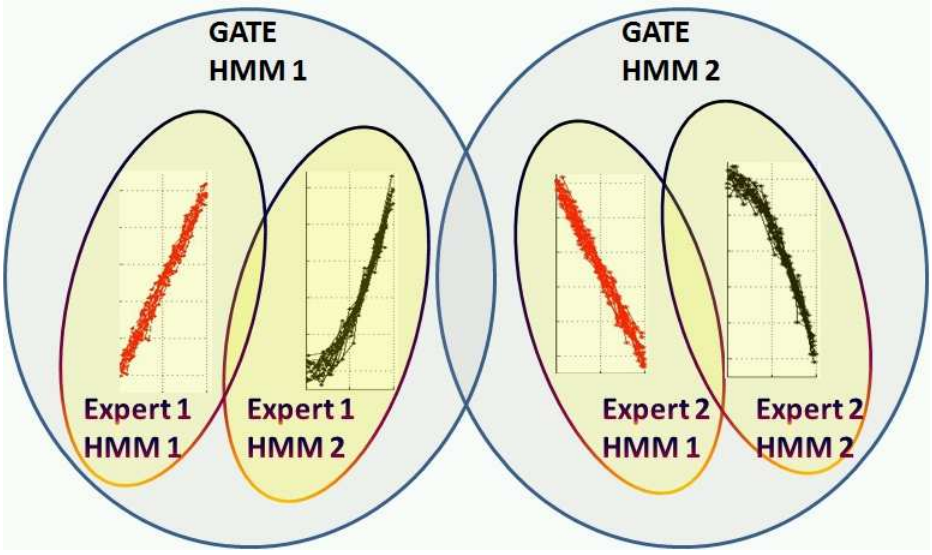


Figure 6-3. MHMME results on synthetic data. The first HMM in the gate learns a model for $y = x$ and $y = x^2$, and the second HMM in the gate learns a model for $y = -x$ and $y = -x^2$. Then, the first expert learns to disriminate between $y = x$ and $y = x^2$, and the second expert learns HMM models to distinguish between $y = -x$ and $y = -x^2$.

The initial classification rates were 65% on the training data and 60% on the test data. After MHMME learning, the classification rates reached 98% on the training data, and 94% on the test data. The improvement in the objective function with respect to the outer iterations is displayed in Fig. 6-4. With one outer iteration, we mean a complete E-M step where all the parameters in the experts and the gate are updated. Note that at each update of an HMM, there are several inner iterations that are run until convergence, as given in Algorithm 6.2.1.

The red curve is the objective function of the gates, Q_g, in Eq. 6–11, and the green curve is the objective function of the experts, Q_e, in Eq. 6–12. Q_g and Q_e are summed to get the total objective function Q in Eq. 6–10, which is displayed as the solid blue curve. The patterns in the iterations point to an interesting observation. In the first iteration, the gate stays the same, and the experts get a quick update. Then the update of the expert attains a smaller incline, while the gate shows a significant increase for the next two iterations. Finally, when the gate updates become almost constant, the experts keep adjusting themselves until both the experts and the gate reach a steady solution. With these adjustments at each iteration, the experts strive to best represent the sequences that are highly weighted by their corresponding gates.

It should be noted that with a different initialization or with different number of experts, the gate could partition the space in a different way as there are many ways to solve this problem. In that case, one expert/gate combination learns whatever the other expert/gate combinations are not learning, and finds meaningful patterns for the data that received low confidences from the other experts. With this interpretation, one can compare it to AdaBoost, however, there is at least one major difference: the experts and the gates are learned in conjuction with each other, and updated simultanously; whereas in AdaBoost, the experts are learned successively, and there is no going back when an expert is learned.

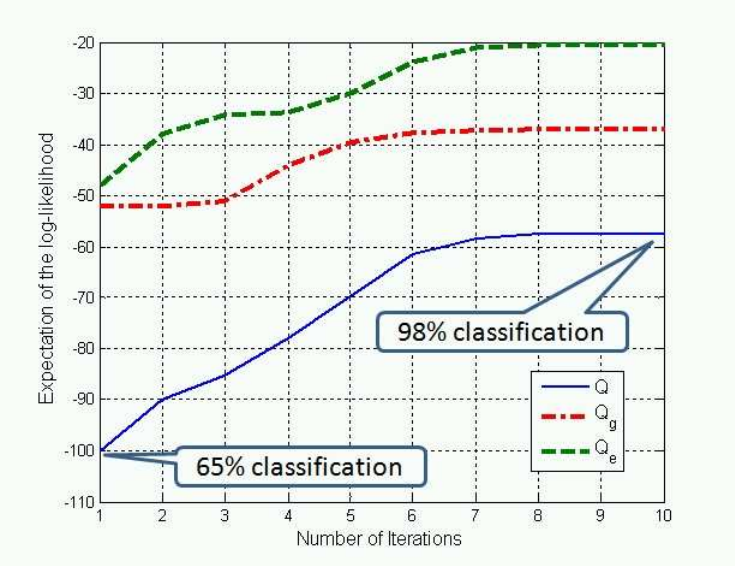


Figure 6-4. MHMME objective function. The objective functions for the gate (Q_g)and the experts (Q_e) versus the number of iterations. The total objective function Q is displayed in blue, which is the sum of Q_g and Q_e.

6.4 Results on the Landmine Dataset

In landmine detection, landmines and metallic clutter show smoother EMI response with respect to response of nonmetallic clutter. Moreover, each type of landmine has a unique EMI response depending on the distribution of metal in the target. However, this unique response cannot always be observed in its perfect shape, could be cluttered with noise, and could be scaled depending on the depth the mine is buried at. Therefore, MHMME model can find multiple models from the data that represent each of these contexts, and do a better classification than those ignoring the context.

The metal detector's EMI response can be modeled as

$$S(w) = A[I(w) + iQ(w)] \tag{6–28}$$

where w is the frequency, A is the magnitude, $I(w)$ is the real (in-phase) response and $Q(w)$ is the imaginary (quadrature) response of the complex system. The term $I(w) + iQ(w)$ describes the shape of the response as a function of the 21 frequencies [17]. The real EMI response $I(w)$ is plotted against the imaginary EMI response $Q(w)$

where w is the 21 frequencies of w. The plot of the real response with respect to the imaginary response is called an argand diagram. The shape of an argand diagram is indicative of the type and distribution of metal in a target [16], and mines of the same type show similar argand curves that are scaled replicas of each other depending on the depth. To eliminate the variation in magnitude due to depth, typically, the magnitude is normalized between $[0, 1]$ [166]. Examples of argand diagrams are shown in Fig. 6-5, for two different types of mines in Fig. 6-5(A-B), for a metallic object in Fig. 6-5(C), and a for nonmetallic objects in Fig. 6-5(D).

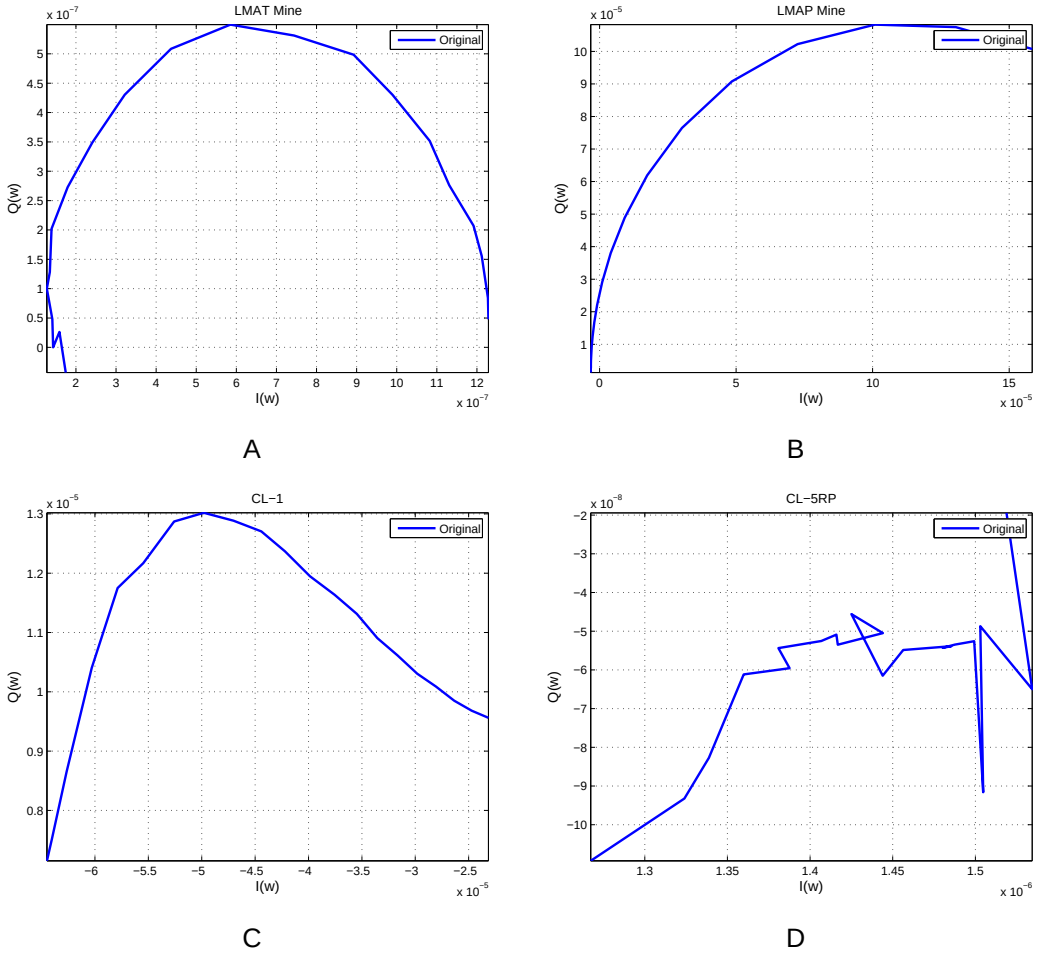


Figure 6-5. Argand diagrams for mine and clutter objects. An argand diagram is the plot of the real EMI response $I(w)$ vs. the imaginary EMI response $Q(w)$. Metallic objects show smoother and distinct shapes such as the LMAT mine in (A), LMAP mine in (B), metallic clutter object in (C). Nonmetallic objects have noisy argand plots such as the nonmetallic clutter object in (D).

Using the shape information from the argand diagrams, it is possible to discriminate between some landmines and clutter. The WEMI sensor's response is a $21 \times 3 \times 150$ complex data matrix for each grid. Similar to the existing systems [20], the data vector from the middle channel at the center of the cell was used for analysis. The argand sequences were discretized to the 50 cluster centers found by fuzzy C-means (FCM) clustering. These cluster centers are displayed in Fig. 6-6. All the HMMs were set to have 3 states, found experimentally. In the rest of this section, we will visualize the properties of MHMME on one fold of a two-fold training. Then we will report our classification results on a 10-fold cross-validation training/testing.

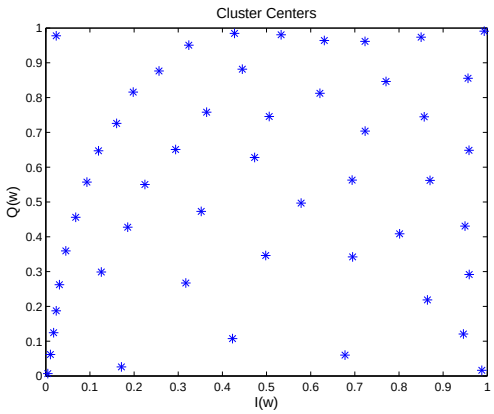


Figure 6-6. Clusters centers of the landmine data. The cluster centers are found using FCM-clustering, and used to quantize the landmine data.

For visualization purposes, the results are displayed on one fold of a 2-fold training. Of the 888 sequences of mines and non-mines, 444 of them were used in the training. The MHMME architecture was set to have 8 experts, which corresponds to 8 HMMs at the gate, and 2 HMMs at each expert. To initialize the gate, the first 4 HMMs were computed from the sequences in the mine class (class 1) using the clustering approach in [137], which was described in the previous chapter. Similarly, the last 4 HMMs were initialized from the nonmine sequences, using the same clustering approach. This results in 8 HMMs, which are the initial parameters of the gate. Then, all the training sequences were tested with all the HMMs, and those that received high likelihoods were

used to train an HMM for each expert. For example, if the gate HMM 1 gave higher likelihoods to 20 sequences from the first class and 10 sequences from the second class, the first HMM of the first expert was initialized with BW-HMM training on the 20 sequences from the first class; and the second HMM of the first expert was initialized with BW-HMM training on the 10 sequences from the second class.

After the MHMME learning, the distribution of the sequences to the gate is shown in Fig. 6-7. On the x-axis, objects from the mine class and the nonmine class are separated by the yellow line, such that the first 156 objects belong to the mine class, and the next 288 objects belong to the nonmine class. The colors represent the true labels of all these objects. On the y-axis, each object is placed at the gate that has the highest weight for that object. For example, the 7th gate successfully models metallic clutter (MC) and nonmetallic clutter (NMC) objects, and does not get confused with any mines. On the other hand, the second gate predominantly models the HMAP and HMAT mines, however, does get confused with some metallic clutter. Similarly, the 4th gate picks some LMAP and LMAT objects, and gets confused with MC mostly. Considering the metal content of the targets, such a division makes a lot of sense.

Actually, there is no such hard assignment, and all of the assignments in Fig. 6-7 have a probability attached to them. If we find the sequences with the highest probabilities at each gate, we find the sequences that represent the context that a gate has learned. These sequences are displayed in Fig. 6-8 and in Fig. 6-11. In these figure, the type of the mine or nonmine object is given in the y-axis. For example, the first context is defined by the LMAT mines that have a particular shape. Similarly, LMAT and HMAP objects of a particular shape share the second context.

The Viterbi paths corresponding to these sequences in Fig. 6-8 and in Fig. 6-11 are given in Fig. 6-10 and Fig. 6-11 to show the difference in the paths for each context.

The classification results are shown in Fig. 6-16. In (A), the sequences that are highly weighted are displayed. This is the same figure as Fig. 6-7, only without the

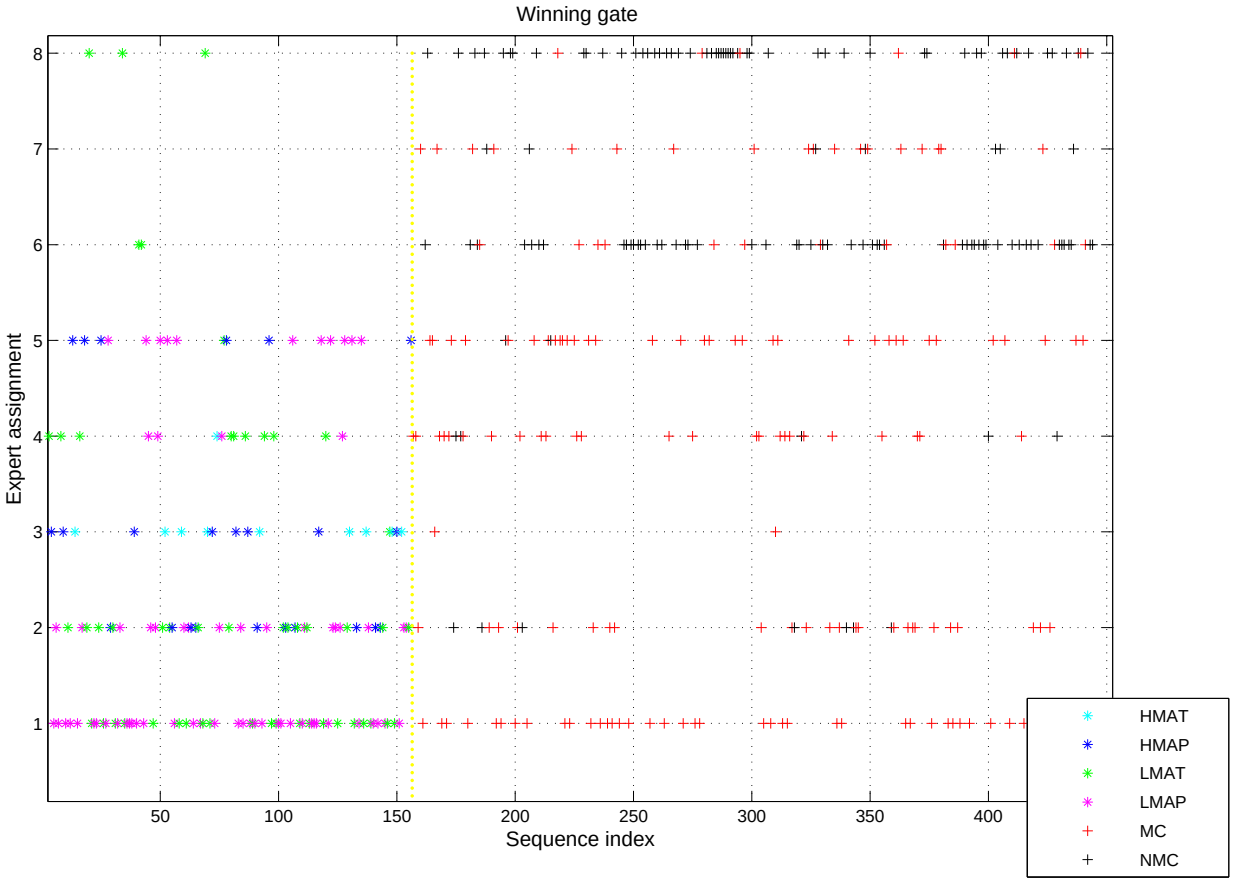
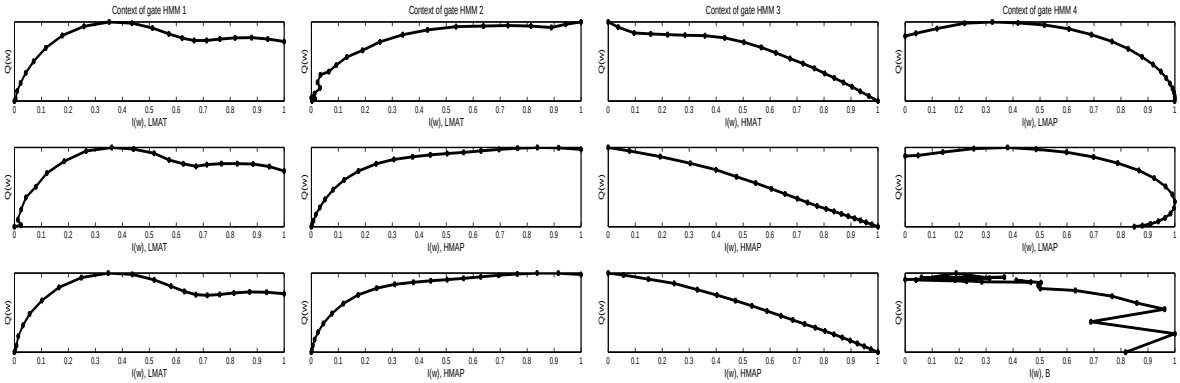


Figure 6-7. Distribution of the sequences by the gate to the experts. On the x-axis, objects from the mine class and the nonmine class are separated by the yellow line, such that the first 156 objects belong to the mine class, and the next 288 objects belong to the nonmine class. The colors represent the true labels of all these objects. On the y-axis, each object is placed at the gate that has the highest weight for that object.

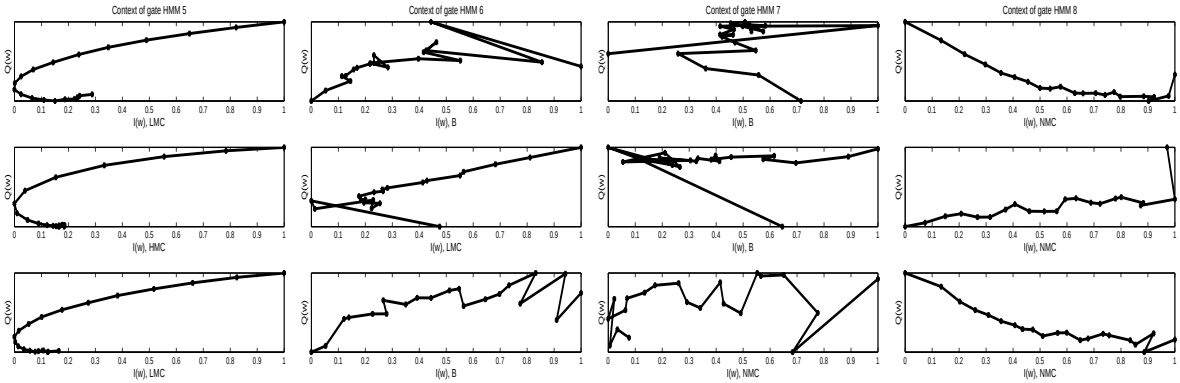
colors. The experts help correctly classify the sequences that create a confusion, resulting in the final classification in Fig. 6-16(B) which has less objects that are misclassified.

To report our classification results, we ran twenty experiments of 10-fold cross-validation. The ROC curve for the 10-fold cross-validation experiment is displayed in Fig. 6-13. 90% true positive rate is attained at 30% false alarm rate. The classification rates are given in Table 6-1. Each entry is described below.



A Gate HMM 1 B Gate HMM 2 C Gate HMM 3 D Gate HMM 4

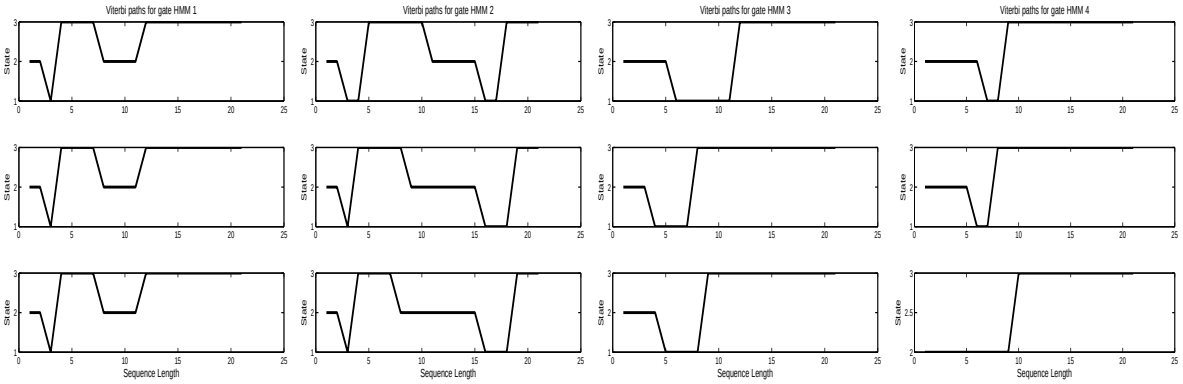
Figure 6-8. Contexts defined by the gate HMMs 1-4. Each column shows the top three argand sequences among all the training sequences that are assigned the highest weight by the gate HMM. On the y-axis, the type of the mine or nonmine object is given.



A Gate HMM 5 B Gate HMM 6 C Gate HMM 7 D Gate HMM 8

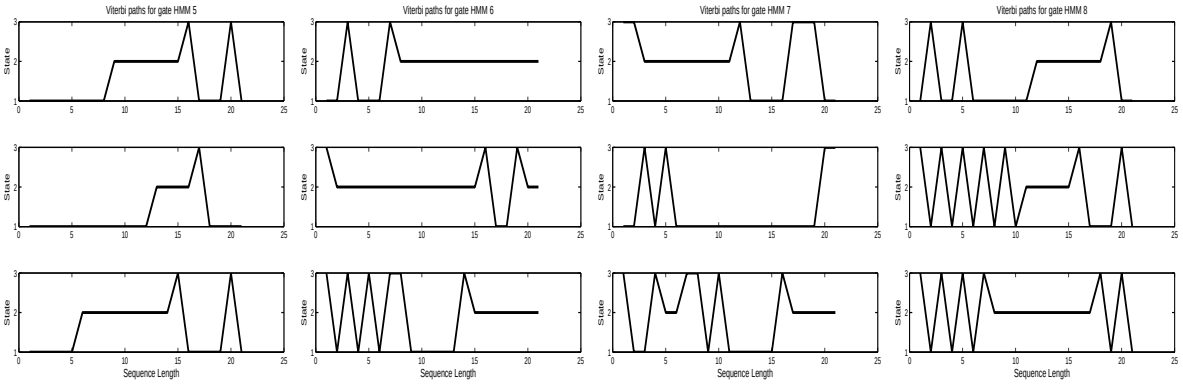
Figure 6-9. Contexts defined by the gate HMMs 5-8. Each column shows the top three sequences among all the training sequences that are assigned the highest weight by the gate HMM. On the y-axis, the type of the mine or nonmine object is given.

CI-HMM: Sequences from each class are clustered into 4 using [137], and an HMM is learned for each cluster, resulting in 8 HMMs. A test sequence is assigned to the class whose HMM yields the highest log-likelihood. This method is also used in initializing the gate of the MHMME model.



A Gate HMM 1 B Gate HMM 2 C Gate HMM 3 D Gate HMM 4

Figure 6-10. Viterbi paths of the sequences highly weighted by gate HMMs 1-4, corresponding to the original sequences in Fig. 6-8. Each column shows the Viterbi paths of the top three sequences among all the training sequences that are assigned the highest weight by the gate HMM.

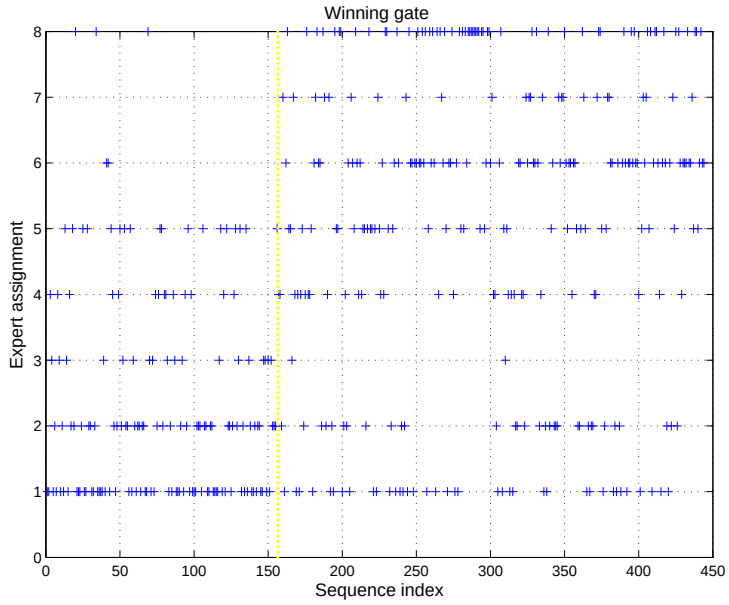


A Gate HMM 5 B Gate HMM 6 C Gate HMM 7 D Gate HMM 8

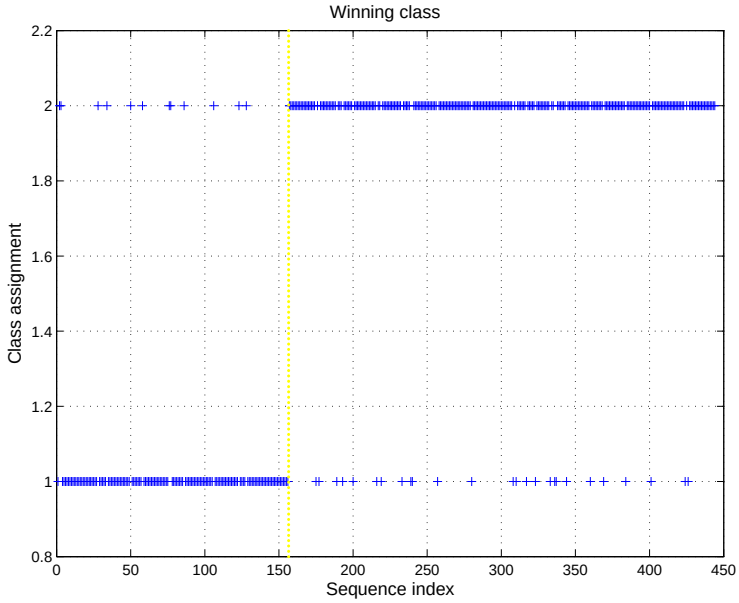
Figure 6-11. Viterbi paths of the sequences highly weighted by gate HMMs 5-8, corresponding to the original sequences in Fig. 6-8. Each column shows the Viterbi paths of the top three sequences among all the training sequences that are assigned the highest weight by the gate HMM.

Gate: The HMMs at the gate of the final MHMME model are used as classifiers to test their individual performance. The first four HMMs are assumed to represent the first class, and the next four HMMs are assumed to represent the second class.

Experts: The HMMs at the experts of the final MHMME model are used as classifiers to test their individual performance. A test sequence is assigned to the class for which it has the highest log-likelihood for any of the 16 HMMs at the experts.



A



B

Figure 6-12. Sequences highly weighted by the gate are shown in (A). The final classification result is shown in (B). The objects to the left of the yellow line should be assigned to class 1, and the objects to the right of the yellow line should be assigned to class 2. On the top left, the objects from class 1 are classified as class 2; and on the bottom right, the objects from class 2 are classified as class 1. These objects are the misclassified ones, but are much less when compared to the ones the gate misclassified.

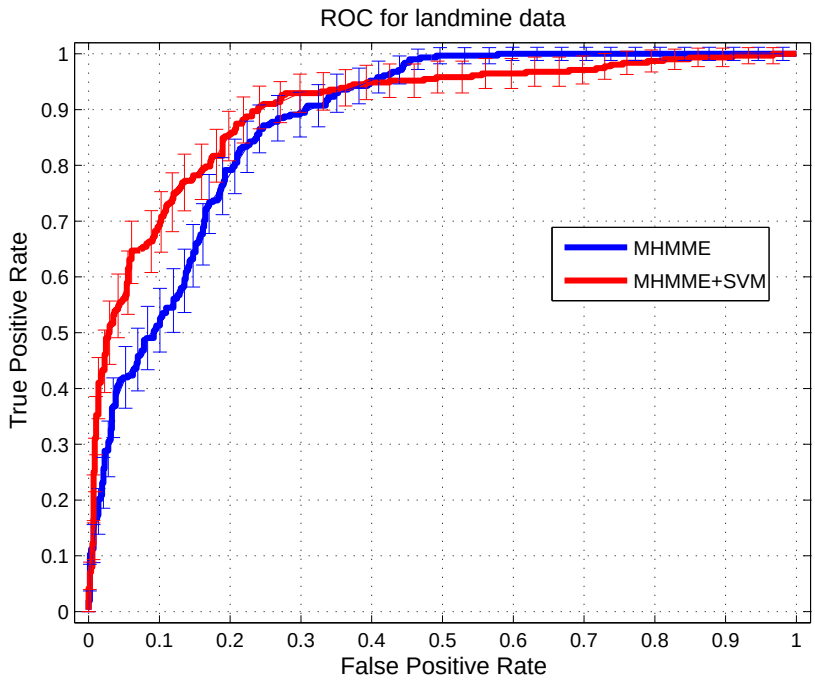


Figure 6-13. ROC curve of the MHMME and the MHMME+SVM for the landmine dataset for a 10-fold cross validation experiment. 90% true positive rate is attained at 30% false alarm rate.

MHMME + SVM: For each sequence, the log-likelihoods obtained from all the HMMs in the MHMME model are concatenated to form a 30-D feature vector, which is then used in training an rbsvm.

These rates are the average of 20 independent training runs, each of which employs a 10-fold cross-validation.

Table 6-1. Classification rates on the landmine dataset for 10-fold training

Model	Mean	Standard Deviation
Cl-HMM	0.70	0.02
Gate	0.71	0.05
Experts	0.61	0.02
MHMME	0.80	0.05
MHMME + SVM	0.83	0.04

6.5 Results on 2D Shape Recognition with the Chicken Pieces Database

The chicken pieces sequences were quantized to 20 symbols which were found using fuzzy c-means clustering of the data. All the HMMs were set to 3 states and 20 symbols. The MHMME model was initialized to have 10 experts, which corresponds to 2 experts per class. The HMMs at the gate were initialized using the clustering approach in [137]. Therefore, by initialization, the context was defined at each gate to be a special cluster of a given class. Within this context, it is the expert's duty to learn models that would discriminate two classes. To be able to compare our results to that in the literature, a two-fold cross-validation was used.

The contexts (i.e. the similar sequences in the HMM sense) have been shown in Fig. 6-14. Each column represents the contexts learned by each HMM by the gate. The three sequences in each column are the ones that attain the highest log-likelihood among all the training sequences when tested against with the corresponding HMMs at the gate.

The log-likelihoods of the gate and the experts for the training data of a given class are shown in Fig. 6-15. Each point corresponds to a data sequence from the training set of a given class. The x-axis shows the difference of the log-likelihoods between two experts. The y-axis shows the log-likelihoods of the gate HMMs. The HMMs at the gate define a context, and the HMM experts specialize within these contexts. As a result, the experts and the gate complement each other resulting in the butterfly effect: if a gate/expert pair is doing bad in a region of the space, the other gate/expert pair performs better and dominates the classification decision.

The sequences for which each gate HMM has the maximum probability is shown in Fig. 6-16(A). Note that by initialization, every two HMMs at the gate is supposed to specialize in one class, however, there is a great overlap between these classes. The experts are able to distinguish between these sequences, resulting in better classification rates in Fig. 6-16(B). These class assignments are probabilistic, and these

probabilities are shown in Fig. 6-16(C). A perfect classification would have had red, green, blue, magenta, and black colors, respectively, in each region between the yellow lines. Any deviation from this color arrangement is a misclassification. For example, a blue point in the leftmost region (1st class) indicates a misclassification, and means that the sequence from the first class has been labeled to belong to the 3rd class (blue) instead of its true label (red). Note that most of the misclassified objects have lower probabilities in Fig. 6-16(C).

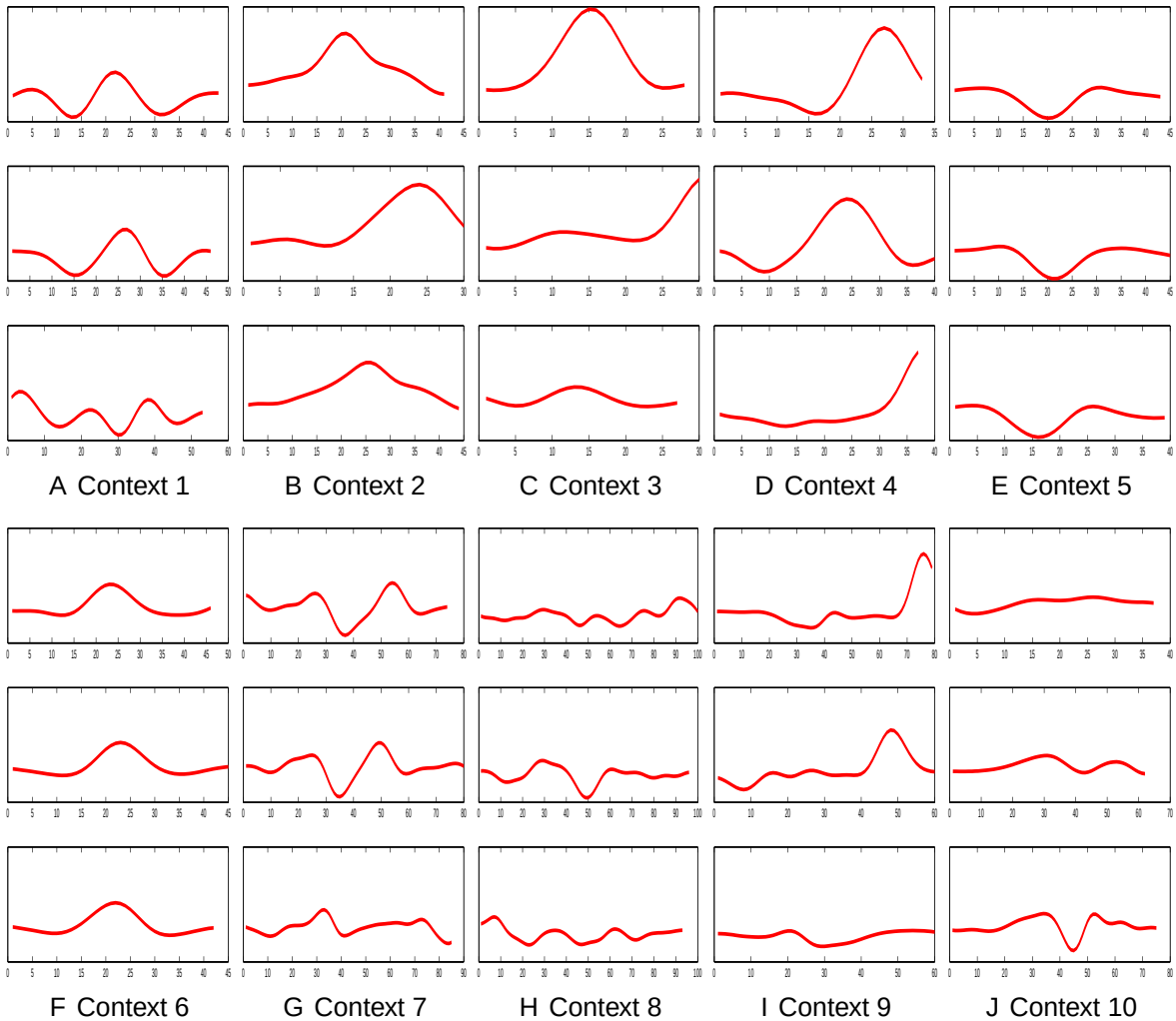


Figure 6-14. Contexts learned by each HMM at the gate. Three sequences that a corresponding HMM is most confident is given in (A)-(J).

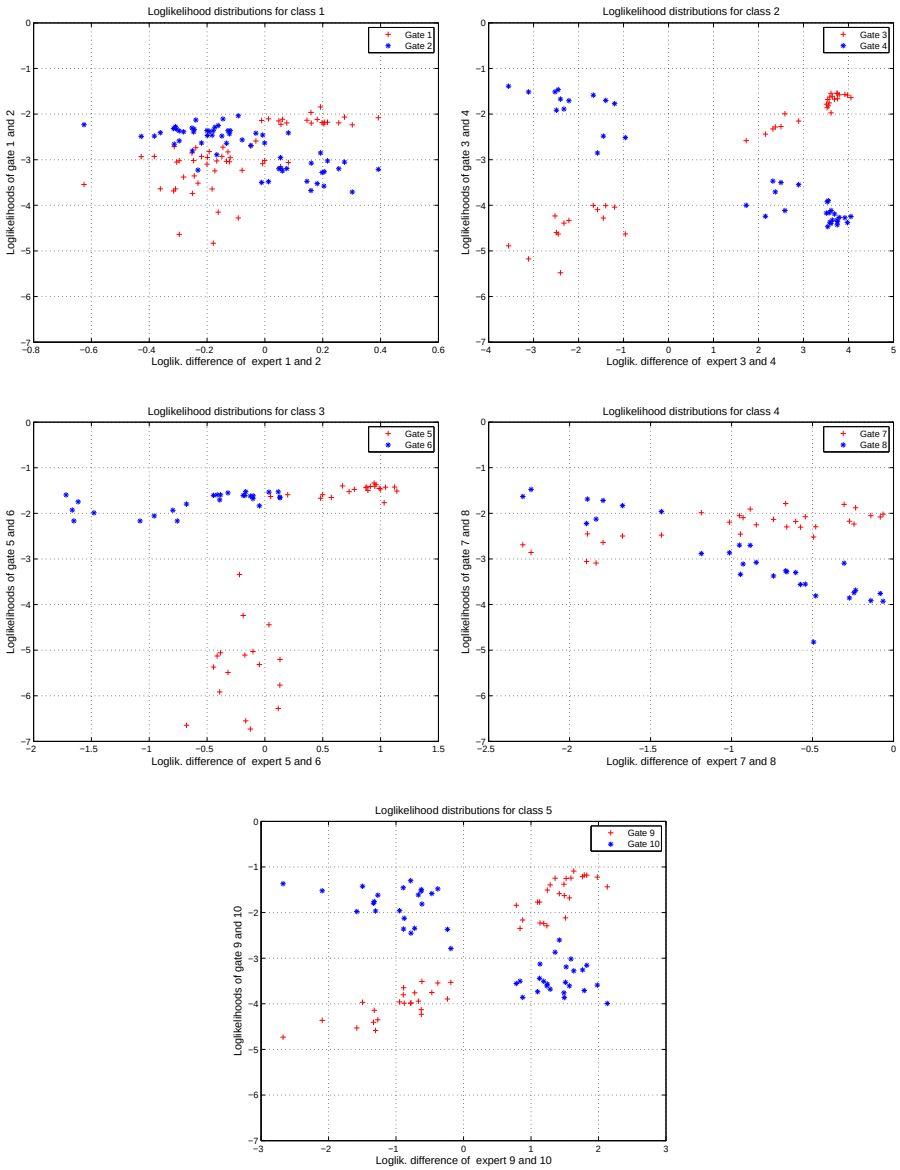


Figure 6-15. The log-likelihoods of the gate and the experts for data of a given class. Each point corresponds to a data sequence from the training set of a given class. The x-axis shows the difference of the log-likelihoods between two experts. The y-axis shows the log-likelihoods of the gate HMMs.

When compared to the ME-only and HMM-only models, our results show that an the combination of ME and HMM models in an MHMME model increases the performance of any single classifier. In addition, we compare MHMME to its individual components, i.e. the HMMs at the experts and at the gate. Our results show that the

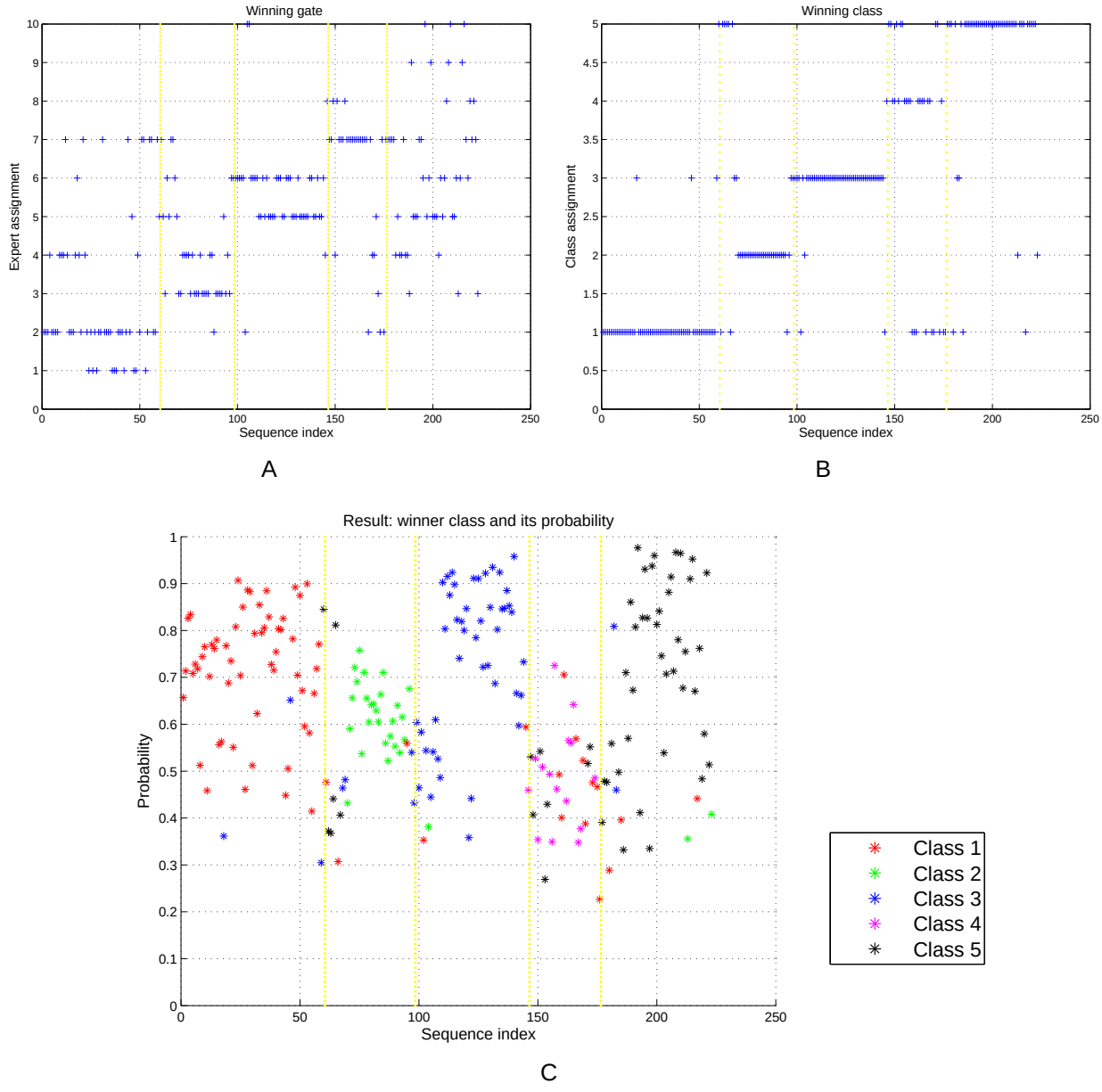


Figure 6-16. Classification results of MHMME on the chicken pieces dataset. Sequences highly weighted by the gate are shown in (A). The yellow lines indicate the true labels of the objects on the x-axis, and the classification results are given on the y-axis. The final classification result is shown in (B), and the probabilities corresponding to each decision is shown in (C). The legends on the right side show the color of the classifier decision. A perfect classification would have had red, green, blue, magenta, and black colors, respectively, in each region between the yellow lines. Any deviation from this color arrangement is a misclassification.

MHMME combination of the individual components increase the classification rates. Each classifier is briefly described below.

PCA+ME: To bring all the sequences to a manageable fixed length for ME, all the sequences are interpolated to length 110, and principal component analysis (PCA) is applied. Sequences are transformed using the 10 most significant eigenvectors, and feature vectors of length 10 are then used to train an ME model.

PCA+VMEC: The same procedure of ME+PCA is applied, but using variational ME for classification (VMEC) instead of ME [44]. The gamma hyperparameters are set to 1.

HMM: A single HMM is trained per class, using Baum-Welch training [[103]]. A test sequence is assigned to the class whose HMM (either of the two HMMs) yields the highest log-likelihood.

Cl-HMM: Sequences from each class are clustered into 2 using [137], and an HMM is learned for each cluster, resulting in 10 HMMs. A test sequence is assigned to the class whose HMM (either of the ten HMMs) yields the highest log-likelihood. This method is also used in initializing the gate of the MHMME model.

Gate: The HMMs at the gate of the final MHMME model are used as classifiers to test their individual performance.

Experts: The HMMs at the experts of the final MHMME model are used as classifiers to test their individual performance.

MHMME + SVM: For each sequence, the log-likelihoods obtained from all the HMMs in the MHMME model are concatenated to form a 15-D feature vector, which is then used in training an rbsvm.

The chicken pieces dataset has been previously investigated in the literature. The study by [145] learns multiple HMMs and inputs the outputs of HMMs as features to an SVM. These classifiers are briefly described below, and the classification rate comparisons of these classifiers are given in Table 6-2.

rbsvm: Radial basis support vector machine [167]. The classification rates were taken from [145].

rbsvm + log-space: One HMM is learned per class. Then, each sequence is represented by a 5-D feature vector, which are the log-likelihoods obtained from the 5 HMMs. These feature vectors are then used in training an rbsvm. The procedure is detailed in [145].

rbsvm + emit-space: Similar to the previous approach, a 3 state HMM is learned per class. For each sequence and each HMM, the sum of the emission probabilities at a given state are recorded. These emission probabilities are then concatenated to form a 15-D feature vector, which is then used in training an rbsvm as detailed in [145].

The classification rates of these classifiers are given in Table 6-2. These rates are the average of 20 independent training runs, each of which employs a 2-fold cross-validation. Note that there are 5 classes, so a random classification would be 20%.

Table 6-2. Classification rates on the chicken pieces dataset for 2-fold training

Model	Mean	Standard Deviation
PCA + ME	0.43	0.01
PCA + VMEC	0.44	0.01
HMM	0.41	0.05
Cl-HMM	0.61	0.03
Gate	0.59	0.08
Experts	0.53	0.04
MHMME	0.73	0.02
MHMME + SVM	0.87	0.01
rbsvm	0.57	0.8
rbsvm + log-space	0.75	0.5
rbsvm + emit-space	0.80	0.5

Among these classifiers, PCA+ME and PCA+VMEC are the two ME-only methods, whereas the HMM and the Cl-HMM are the HMM-only methods. When compared to these, MHMME performs better than an HMM-only and an ME-only method. In addition, the final output is a combination of the experts with a gate, and the combination is

better than the individual experts and the gate. MHMME results are better than an SVM, and are comparable to an SVM trained on log-likelihoods. Moreover, MHMME+SVM surpasses any of the other algorithms.

6.6 Discussion and Future Work

The MHMME algorithm allows for competitive learning at the experts. If one expert makes a better prediction than the others, it receives a higher weighted data point. On the other hand, it also learns to ignore the sequences that are handled better by its competitors. Also through the competition of the clusters, one expert tries to model all the data. In such cases, the other experts act like trash collectors, and collect whichever sequence does not fit well with the model. In such cases, classification rates increase if the expert can find a pattern in this outlier data.

Although we only used a single layer model, it is mathematically possible to easily extend it into a hierarchical model. In the future, it would be interesting to see whether or not another level of hierarchy would increase the classification rates. It is also possible to use variational learning, and investigate the optimum number of clusters, as well as the number of states in each HMM.

CHAPTER 7
CONCLUSION

This research addresses the problems encountered when designing classifiers for classes that contain multiple subclasses whose characteristics are dependent on the context. It is sometimes the case that when the appropriate context is chosen, classification is relatively easy, whereas in the absence of contextual information, classification may be difficult. Therefore, this research focuses on simultaneous learning of context and classification.

The mixture of experts model has been applied to the landmine detection problem and has been shown to increase the classification rates of two other classifiers. The ME model provides a probabilistic approach for dividing the data into regions and for learning experts in each region. To prevent the ME from over-training, a variational ME classification (VMEC) model has been introduced, and a lower bound for VMEC training has been derived. The VMEC learning has been shown to further increase the classification rates of ME on the landmine dataset while providing regularized parameters and distributions for the parameters instead of point estimates.

Thirdly, a novel method, mixture of hidden Markov model experts (MHMME) has been developed. The updates of HMM parameters in an ME framework has been derived, and the benefits of ME has been extended to time-series data. The MHMME model allows for the simultaneous probabilistic learning of the sub-regions from multi-class sequential data and the discrimination of the classes in these sub-regions. The output is a mixture of the HMM decisions, but the mixture coefficient is not fixed once it is learned, rather it depends on the input data. The MHMME model has been tested on a synthetic dataset, the landmine dataset and the chicken pieces dataset. It has been shown to perform better than ME-only and HMM-only models, and do well in comparison to state-of-the-art models.

REFERENCES

[1] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[2] M. I. Jordan and L. Xu, "Convergence results for the EM approach to mixtures of experts architectures," *Neural Networks*, vol. 8, pp. 1409–1431, 1995.

[3] G. Hinton, "Introduction to machine learning, decision trees and mixtures of experts," Lecture notes, 2007.

[4] C. M. Bishop and M. Svensen, "Bayesian hierarchical mixtures of experts," in *Proceedings Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003, pp. 57 – 64.

[5] N. Ueda and Z. Ghahramani, "Optimal model inference for Bayesian mixture of experts," in *Proc. IEEE Workshop on Neural Networks for Signal Processing*, 2000, vol. 1, pp. 145–154.

[6] S. Waterhouse, D. Mackay, and T. Robinson, "Bayesian methods for mixtures of experts," in *Adv. Neur. Inf. Proc. Sys. (NIPS) 7*, 1996, pp. 351–357.

[7] S. R. Waterhouse, *Classification and Regression using Mixtures of Experts*, Ph.D. dissertation, Department of Engineering, University of Cambridge, 1997.

[8] F. Peng, R. A. Jacobs, and M. A. Tanner, "Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition," *J. Amer. Statist. Assoc.*, , no. 91, pp. 953–960, 1996.

[9] K. Chen, D. Xie, and H. Chi, "A modified HME architecture for text-dependent speaker identification," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1309–1313, 1996.

[10] A. S. Weigend, M. Mangeas, and A. N. Srivastava, "Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting," *International Journal of Neural Systems*, , no. 6, pp. 373–399, 1995.

[11] A. S. Weigend and S. Shi, "Predicting daily probability distributions of sp500 returns," *Journal of Forecasting*, vol. 19, no. 4, pp. 375 – 392, 2000.

[12] A. Coelho, C. Lima, and F. Von Zuben, "Hybrid genetic training of gated mixtures of experts for nonlinear time series forecasting," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, vol. 5, pp. 4625 – 4630.

[13] Y. Bengio and P. Frasconi, "Input output HMMs for sequence processing," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1231–1249, 1995.

[14] X. Wang, P. Whigham, D. Deng, and M. Purvis, "Time-line hidden Markov experts for time series prediction," *Neural Information Processing - Letters and Reviews*, vol. 3, no. 2, pp. 39–48, May 2004.

[15] "Hidden killers: The global landmine crisis," report 10575, United States Department of State, Sep 1998.

[16] E. B. Fails, P. A. Torrione, J. Waymond R. Scott, and L. M. Collins, "Performance of a four parameter model for modeling landmine signatures in frequency domain wideband electromagnetic induction detection systems," 2007, vol. 6553, pp. 65530D1–7, SPIE.

[17] W. Scott, "Broadband array of electromagnetic induction sensors for detecting buried landmines," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2008, vol. 2, pp. II–375 –II–378.

[18] H. Frigui, P. Gader, and D. Ho, "Real time landmine detection with ground penetrating radar using discriminative and adaptive hidden Markov models," *EURASIP Journal on Applied Signal Processing*, vol. 1, pp. 1867–1885, January 2005.

[19] G. Ramachandran, P. Gader, and J. Wilson, "Fast physics-based mine detection algorihtms for wide-band electromagnetic induction sensors," in *SPIE Defense, Security and Sensing*, April 2009, pp. 7303–77.

[20] G. Ramachandran, P. Gader, and J. Wilson, "Granma: Gradient angle model algorithm on wideband EMI data for land-mine detection," *Geoscience and Remote Sensing Letters, IEEE*, vol. 7, no. 3, pp. 535–539, July 2010.

[21] K. Ho, L. Carin, P. Gader, and J. Wilson, "An investigation of using the spectral characteristics from ground penetrating radar for landmine/clutter discrimination," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 4-2, pp. 1177–1191, April 2008.

[22] H. Frigui and P. Gader, "Detection and discrimination of land mines based on edge histogram descriptors and fuzzy k-nearest neighbors," in *IEEE International Conference on Fuzzy Systems*, 2006, pp. 1494–1499.

[23] H. Frigui and P. Gader, "Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic K-nearest neighbor classifier," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 1, pp. 185–199, Feb. 2009.

[24] J. V. G. Andreu, A. Crespo, "Selecting the toroidal self-organizing feature maps (TSOFM) best organized to object recognition,," in *International Conference on Neural Networks*, 1997, vol. 2, p. 13411346.

[25] M. Bicego, V. Murino, and M. A. T. Figueiredo, "Similarity-based classification of sequences using hidden Markov models," *Pattern Recogn.*, vol. 37, no. 12, pp. 2281–2291, 2004.

[26] M. I. Jordan, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

[27] E. Meeds and S. Osindero, "An alternative infinite mixture of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 883–890.

[28] C. Yuan and C. Neubauer, "Variational mixture of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS) 21*, pp. 1897–1904. 2009.

[29] K.-A. L. Cao, E. Meugnier, and G. J. McLachlan, "Integrative mixture of experts to combine clinical factors and gene markers," *Bioinformatics*, vol. 26, no. 9, pp. 1192–1198, 2010.

[30] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Learning to reconstruct 3D human motion from Bayesian mixture of experts, a probabilistic discriminative approach," Tech. Rep. Technical Report CSRG-502, University of Toronto, October 2004.

[31] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "BM^3E: Discriminative density propagation for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 2030 –2044, nov. 2007.

[32] J. Saragih, S. Lucey, and J. Cohn, "Deformable model fitting with a mixture of local experts," in *12th International Conference on Computer Vision (ICCV)*, Sept. 2009, pp. 2248–2255.

[33] C. A. M. Lima, A. L. V. Coelho, and F. J. Von Zuben, "Hybridizing mixtures of experts with support vector machines: Investigation into nonlinear dynamic systems identification," *Information Sciences*, vol. 177, no. 10, pp. 2049–2074, 2007.

[34] B. Yao, D. Walther, D. Beck, and L. Fei-Fei, "Hierarchical mixture of classification experts uncovers interactions between brain regions," in *Advances in Neural Information Processing Systems (NIPS) 22*, pp. 2178–2186. 2009.

[35] H.-J. Xing and B.-G. Hu, "An adaptive fuzzy c-means clustering-based mixtures of experts model for unlabeled data classification," *Neurocomput.*, vol. 71, pp. 1008–1021, January 2008.

[36] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Fast Algorithms for Large Scale Conditional 3D Prediction," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[37] Z. Lu, "A regularized minimum cross-entropy algorithm on mixtures of experts for time series prediction and curve detection," *Pattern Recognition Letters*, vol. 27, no. 9, pp. 947 – 955, 2006.

[38] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," in *Advances in Neural Information Processing Systems (NIPS) 14*, 2002, pp. 881–888.

[39] Y. Li, "Hidden Markov models with states depending on observations," *Pattern Recogn. Lett.*, vol. 26, no. 7, pp. 977–984, 2005.

[40] R. Ebrahimpour, M. R. Moradian, A. Esmkhani, and F. M. Jafarlou, "Recognition of Persian handwritten digits using characterization loci and mixture of experts," *Journal of Digital Content Technology and its Applications*, vol. 3, no. 3, pp. 42 – 46, 2009.

[41] R. Ebrahimpour, E. Kabir, H. Esteky, and M. R. Yousefi, "View-independent face recognition with mixture of experts," *Neurocomputing*, vol. 71, no. 4–6, pp. 1103 – 1107, 2008.

[42] Y. Qi, J. Klein-Seetharaman, and Z. Bar-Joseph, "A mixture of feature experts approach for protein-protein interaction prediction," *BMC Bioinformatics*, vol. 8, no. (S10)- S6, 2007.

[43] S. E. Yuksel, G. Ramachandran, P. Gader, J. Wilson, D. Ho, and G. Heo, "Hierarchical methods for landmine detection with wideband electro-magnetic induction and ground penetrating radar multi-sensor systems," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2008, vol. 2, pp. II–177–II–180.

[44] S. E. Yuksel and P. Gader, "Variational mixture of experts for classification with applications to landmine detection," in *International Conference on Pattern Recognition (ICPR)*, 2010, pp. 2981–2984.

[45] M. S. Yumlu, F. S. Gurgen, , and N. Okay, "Financial time series prediction using mixture of experts," in *18th International Symposium on Computer and information sciences (ISCIS)*, 2003, vol. 2869, pp. 553 – 560.

[46] M. Versace, R. Bhatt, O. Hinds, and M. Shiffer, "Predicting the exchange traded fund dia with a combination of genetic algorithms and neural networks," *Expert Systems with Applications*, vol. 27, no. 3, pp. 417 – 425, 2004.

[47] I. Guler, E. Derya Ubeyli, and N. Fatma Guler, "A mixture of experts network structure for EEG signals classification," in *27th Annual International Conference of the Engineering in Medicine and Biology Society (IEEE-EMBS)*, 17-18 2005, pp. 2707 –2710.

[48] Y. Jiang and P. Guo, "Mixture of experts for stellar data classification," in *International Symposium on Neural Networks (ISNN)*, 2005, vol. 2, pp. 310–315.

[49] S. Mossavat, O. Amft, B. de Vries, P. Petkov, and W. Kleijn, "A Bayesian hierarchical mixture of experts approach to estimate speech quality," in *Second International Workshop on Quality of Multimedia Experience (QoMEX)*, jun. 2010, pp. 200 –205.

[50] H. Harb, L. Chen, and J.-Y. Auloge, "Mixture of experts for audio classification: an application to male female classification and musical genre recognition," in *IEEE International Conference on Multimedia and Expo (ICME)*, jun. 2004, vol. 2, pp. 1351 –1354.

[51] F. Michel and N. Paragios, "Image transport regression using mixture of experts and discrete Markov random fields," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, Apr. 2010, pp. 1229 –1232.

[52] L. Cao, "Support vector machines experts for time series forecasting," *Neurocomputing*, pp. 321–339, 2003.

[53] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, pp. 1–37, 2008.

[54] B. Tang, M. Heywood, and M. Shepherd, "Input partitioning to mixture of experts," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, 2002, vol. 1, pp. 227–232.

[55] A. A. Montillo, "Random forests," Lecture in Statistical Foundations of Data Analysis, April 2009.

[56] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, New York: Springer-Verlag, 2001.

[57] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, 1999, 2nd edition.

[58] T.-K. Kim, "Boosting and random forest for visual recognition," International Conference on Computer Vision (ICCV) Tutorial, 2009.

[59] R. Avnimelech and N. Intrator, "Boosted mixture of experts: an ensemble learning scheme," *Neural Comput.*, vol. 11, no. 2, pp. 483–497, 1999.

[60] R. A. Jacobs, "Bias/variance analyses of mixtures-of-experts architectures," *Neural Computation*, vol. 9, pp. 369–383, 1997.

[61] A. Zeevi, R. Meir, and V. Maiorov, "Error bounds for functional approximation and estimation using mixtures of experts," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 1010 –1025, May 1998.

[62] W. Jiang and M. Tanner, "On the approximation rate of hierarchical mixtures-of-experts for generalized linear models," *Neural Comput.*, vol. 11, no. 5, pp. 1183–1198, 1999.

[63] W. Jiang, "The VC dimension for mixtures of binary classifiers," *Neural Computation*, vol. 12, no. 6, pp. 1293–1301, June 2000.

[64] W. Jiang and M. Tanner, "On the asymptotic normality of hierarchical mixtures-of-experts for generalized linear models," *IEEE Transactions on Information Theory*, vol. 46, no. 3, pp. 1005 –1013, May 2000.

[65] W. Jiang and M. A. Tanner, "On the identifiability of mixtures-of-experts," *Neural Networks*, vol. 12, pp. 1253–1258, 1999.

[66] W. Jiang and M. A. Tanner, "Hierarchical mixtures-of-experts for exponential family regression models: Approximation and maximum likelihood estimation," *Annals of Statistics*, vol. 27, pp. 987–1011, 1999.

[67] A. Carvalho and M. Tanner, "Mixtures-of-experts of autoregressive time series: asymptotic normality and model specification," *IEEE Trans Neural Netw.*, vol. 16, no. 1, pp. 39–56, Jan 2005.

[68] Y. Ge and W. Jiang, "A note on mixtures of experts for multiclass responses: approximation rate and consistent Bayesian inference," in *Proceedings of the 23rd international conference on machine learning (ICML)*, 2006, pp. 329–335.

[69] Y. Ge and W. Jiang, "On consistency of Bayesian inference with mixtures of logistic regression," *Neural Comput.*, vol. 18, pp. 224–243, January 2006.

[70] L. Xu, M. I. Jordan, and G. E. Hinton, "An alternative model for mixtures of experts," in *Advances in Neural Information Processing Systems (NIPS) 7*. 1995, pp. 633 – 640, MIT press.

[71] S. Waterhouse and A. Robinson, "Classification using hierarchical mixtures of experts," in *Proc. IEEE Workshop on Neural Networks for Signal Processing IV*, 1994, pp. 177 – 186.

[72] K. Chen, D. Xie, and H. Chi, "Speaker identification based on the time-delay hierarchical mixture of experts," in *Proc. IEEE International Conference on Neural Networks*, Nov/Dec 1995, vol. 4, pp. 2062 –2066.

[73] V. Ramamurti and J. Ghosh, "Advances in using hierarchical mixture of experts for signal classification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1996, pp. 3569–3572.

[74] K. Chen, L. Xu, and H. Chi, "Improved learning algorithms for mixture of experts in multiclass classification," *Neural Netw.*, vol. 12, pp. 1229–1252, November 1999.

[75] S.-K. Ng and G. McLachlan, "Using the EM algorithm to train neural networks: misconceptions and a new algorithm for multiclass classification," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 738 –749, May. 2004.

[76] S. Ng and G. McLachlan, "Extension of mixture-of-experts networks for binary classification of hierarchical data," *Artificial Intelligence in Medicine*, vol. 41, no. 1, pp. 57–67, Sept. 2007.

[77] M. K. Titsias and A. Likas, "Mixture of experts classification using a hierarchical mixture model," *Neural Computation*, vol. 14, pp. 2221–2244, 2002.

[78] A. Zeevi, R. Meir, and R. Adler, "Time series prediction using mixtures of experts," in *Advances in Neural Information Processing Systems (NIPS) 9*, 1997, pp. 309–315.

[79] C. Wong and W. Li, "On a logistic mixture autoregressive model," *Biometrika*, vol. 88, no. 3, pp. 833 –846, 2001.

[80] K. Chen, D. Xie, and H. Chi, "Combine multiple time-delay HMEs for speaker identification," in *Proc. IEEE International Conference on Neural Networks*, Jun 1996, vol. 4, pp. 2015–2020.

[81] T. W. Cacciatore and S. J. Nowlan, "Mixtures of controllers for jump linear and non-linear plants," in *Advances in Neural Information Processing Systems (NIPS) 6*, 1993, pp. 719–726.

[82] S. Liehr, K. Pawelzik, J. Kohlmorgen, S. Lemm, and K.-R. Mller, "Hidden Markov mixtures of experts for prediction of non-stationary dynamics," in *In NNSP'99 Workshop on Neural Networks for Signal Processing*, 1999, pp. 195–204.

[83] T. G. Dietterich, "Machine learning for sequential data: A review," in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, 2002, pp. 15–30.

[84] Y. Zhao, R. Schwartz, J. Sroka, and J. Makhoul, "Hierarchical mixtures of experts methodology applied to continuous speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1995, vol. 5, pp. 3443 –3446.

[85] M. Meila and M. I. Jordan, "Learning fine motion by Markov mixtures of experts," in *Advances in Neural Information Processing Systems (NIPS) 8*. 1996, pp. 1003–1009, MIT Press.

[86] J. Fritsch, M. Finke, and A. Waibel, "Context dependent hybrid HME HMM speech recognition using polyphone clustering decision trees," in *Proceedings of the IEEE*

International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1997, pp. 1759–1762.

[87] J. Fritsch and M. Finke, "Improving performance on switchboard by combining hybrid HME/HMM and mixture of Gaussians acoustic models," in *Proceedings of Eurospeech*, 1997, pp. 1963–1966.

[88] M. I. Jordan, Z. Ghahramani, and L. K. Saul, "Hidden Markov decision trees," in *Proc. Conf. Advances in Neural Information Processing Systems (NIPS)*, 1996, pp. 501–507.

[89] Z. Ghahramani and M. I. Jordan, "Factorial hidden Markov models," *Mach. Learn.*, vol. 29, pp. 245–273, November 1997.

[90] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[91] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197–227, 1990.

[92] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *In Advances in Neural Information Processing Systems (NIPS) 12*, 2000, pp. 512–518.

[93] S. Waterhouse and G. Cook, "Ensemble methods for phoneme classification," in *Advances in Neural Information Processing Systems (NIPS)*. 1997, number 9, Cambridge, MA: MIT Press.

[94] J. H. Friedman, "Multivariate adaptive regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[95] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," in *Advances in Neural Information Processing Systems (NIPS) 3*, 1991.

[96] M. Jordan and R. Jacobs, "Hierarchical mixtures of experts and the em algorithm," in *Proceedings of 1993 International Joint Conference on Neural Networks (IJCNN)*, 1993, vol. 2, pp. 1339 – 1344.

[97] P. Estevez and R. Nakano, "Hierarchical mixture of experts and max min propagation neural networks," in *Proc. IEEE International Conference on Neural Networks*, Nov/Dec 1995, vol. 1, pp. 651–656 vol.1.

[98] D. MacKay, *Information theory, inference, and learning algorithms*, Cambridge Univ Press, 2003.

[99] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[100] D. M. Titterington and a. U. M. A.F.M Smith, *Statistical Analysis of Finite Mixture Distributions*, John Wiley, New York, 1985.

[101] R. A. Jacobs, F. Peng, and M. A. Tanner, "A Bayesian approach to model selection in hierarchical mixtures-of-experts architectures," *Neural Netw.*, vol. 10, no. 2, pp. 231–241, 1997.

[102] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4–15, January 1986.

[103] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257–286.

[104] Y. Zhao, P. Gader, P. Chen, and Y. Zhang, "Training dhmms of mine and clutter to minimize landmine detection errors," *IEEE Trans. Geosciences and Remote Sensing*, vol. 41, no. 5, pp. 1016–1024, May 2003.

[105] P. Gader, M. Mystkowski, and Y. Zhao, "Landmine detection with ground penetrating radar using hidden Markov models," *IEEE Trans. Geosciences and Remote Sensing*, vol. 39, no. 6, pp. 1231–1244, 2001.

[106] X. Zhang, S. E. Yuksel, P. Gader, and J. Wilson, "Simultaneous feature and HMM model learning for landmine detection using ground penetrating radar," in *6th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS)*, Aug. 2010, pp. 1–4.

[107] M. Mohamed and P. Gader, "Generalized hidden Markov models. ii. application to handwritten word recognition," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 1, pp. 82 –94, Feb 2000.

[108] R. Brooks, J. Schwier, and C. Griffin, "Behavior detection using confidence intervals of hidden Markov models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 6, pp. 1484 –1492, Dec. 2009.

[109] Y. Nankaku and K. Tokuda, "Face recognition using hidden Markov eigenface models," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 15-20 2007, vol. 2, pp. II–469 –II–472.

[110] G. Singh and H. Song, "Using hidden Markov models in vehicular crash detection," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1119 –1128, march 2009.

[111] S. Eddy, "Profile hidden Markov models," *Bioinformatics*, vol. 14, no. 9, pp. 755–763, 1998.

[112] B.-H. Juang and L. Rabiner, "The segmental k-means algorithm for estimating parameters of hidden Markov models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 9, pp. 1639 – 1641, Sept. 1990.

[113] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 3043 –3054, Dec 1992.

[114] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[115] D. J. C. MacKay, "Ensemble learning for hidden Markov models," 1997.

[116] S. Ji, B. Krishnapuram, and L. Carin, "Variational bayes for continuous hidden Markov models and its application to active learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 522, 2006.

[117] C. A. McGrory and M. Titterington, "Variational Bayesian analysis for hidden Markov models," *Australian and New Zealand Journal of Statistics*.

[118] S. Kapadia, *Discriminative Training of Hidden Markov Models*, Ph.D. dissertation, University of Cambridge, March 1998.

[119] H. Kuo, E. Fosler-Lussier, H. Jiang, and C. Lee, "Discriminative training of language models for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[120] J. A. Bilmes, "A gentle tutorial on the em algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," Technical report, University of California, Berkeley, 1997.

[121] B.-H. Juang, W. Hou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 3, pp. 257 – 265, May 1997.

[122] L. R. Rabiner, C. H. Lee, B. H. Juang, and J. G. Wilpon, "HMM clustering for connected word recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1989, pp. 405–408.

[123] 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 (CVPR)*, 2003, pp. 375–381.

[124] E. Swears, A. Hoogs, and A. Perera, "Learning motion patterns in surveillance video using HMM clustering," in *IEEE Workshop on Motion and video Computing*, 2008, pp. 1–8.

[125] C. Goutte, P. Toft, E. Rostrup, F. A. Nielsen, and L. K. Hansen, "On clustering fMRI time series," *NeuroImage*, vol. 9, no. 3, pp. 298–310, 1999.

[126] R. H. Shumway, "Time-frequency clustering and discriminant analysis," *Statistics and Probability Letters*, vol. 63, no. 3, pp. 307–314, 2003.

[127] A. W. Black and P. Taylor, "Automatically clustering similar units for unit selection in speech synthesis.," in *Proc. of Eurospeech*, 1997, pp. 601–604.

[128] K. Yu and M. Gales, "Discriminative cluster adaptive training," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1694–1703, Sept. 2006.

[129] M. Gales, "Cluster adaptive training of hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 4, pp. 417–428, 2000.

[130] M. Naito, L. Deng, and Y. Sagisaka, "Speaker clustering for speech recognition using the parameters characterizing vocal-tract dimensions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 12-15 1998, vol. 2, pp. 981 –984.

[131] G. Mayraz and G. E. Hinton, "Recognizing handwritten digits using hierarchical products of experts," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 24, no. 2, pp. 189–197, 2002.

[132] M. Bicego and V. Murino, "Investigating hidden Markov models' capabilities in 2d shape classification," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 26, no. 2, pp. 281–286, 2004.

[133] T. W. Liao, "Clustering of time series data–a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[134] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, pp. 349–371, 2003.

[135] C. Li and G. Biswas, "A bayesian approach to temporal data clustering using hidden Markov models," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 2000, pp. 543–550.

[136] Z. Szamonek and C. Szepesvari, "X-mHMM: an efficient algorithm for training mixtures of HMMs when the number of mixtures is unknown," in *IEEE International Conference on Data Mining*, 2005, pp. 27–30.

[137] P. Smyth, "Clustering sequences with hidden Markov models," in *Advances in Neural Information Processing Systems (NIPS)*, 1997, pp. 648–654.

[138] F. Korkmazskiy, B.-H. Juang, and F. Soong, "Generalized mixture of HMMs for continuous speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 21-24 1997, vol. 2, pp. 1443 –1446.

[139] T. Oates, L. Firoiu, and P. R. Cohen, "Clustering time series with hidden Markov models and dynamic time warping," in *Proceedings of the IJCAI Workshop on*

Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning, 1999.

[140] T. Jebara, Y. Song, and K. Thadani, "Spectral clustering and embedding with hidden Markov models," in *Proceedings of the 18th European Conference on Machine Learning (ECML)*, September 2007.

[141] A. C. Tsoi, S. Zhang, and M. Hagenbuchner, *Data Mining*, vol. 3755, chapter Hierarchical Hidden Markov Models: An Application to Health Insurance Data, pp. 244–259, Lecture Notes in Computer Science, 2006.

[142] J. Yin and Q. Yang, "Integrating hidden Markov models and spectral analysis for sensory time series clustering," in *IEEE International Conference on Data Mining*, 2005, pp. 27–30.

[143] F. Porikli, "Clustering variable length sequences by eigenvector decomposition using hmm," in *International Workshop on Structural and Syntactic Pattern Recognition*. 2004, p. 352, Lecture Notes in Computer Science, Springer-Verlag.

[144] M. Bicego, V. Murino, and M. A. Figueiredo, "Similarity-based clustering of sequences using hidden Markov models," in *Machine Learning and Data Mining in Pattern Recognition, vol. LNAI 2734*, 2003, pp. 86–95.

[145] M. Bicego, E. Pekalska, D. M. J. Tax, and R. P. W. Duin, "Component-based discriminative classification for hidden Markov models," *Pattern Recogn.*, vol. 42, no. 11, pp. 2637–2648, 2009.

[146] D. Garcia-Garcia, E. Hernandez, and F. Diaz de Maria, "A new distance measure for model-based sequence clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1325 –1331, July 2009.

[147] A. D. Brown and G. E. Hinton, "Relative density nets: A new way to combine backpropagation with HMM's.," in *Advances in Neural Information Processing Systems (NIPS)*, 2001, pp. 1149–1156.

[148] G. Taylor and G. Hinton, "Products of hidden Markov models: It takes $n > 1$ to tango," in *Proc. of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[149] L. R. SE Levinson and M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst Tech J*, vol. 62, no. 4, pp. 1035–74, April 1983.

[150] R. B. Lyngs, C. N. S. Pedersen, and H. Nielsen, "Metrics and similarity measures for hidden Markov models," in *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 1999, pp. 178–186.

[151] T. Kosaka and S. Sagayama, "Tree-structured speaker clustering for fast speaker adaptation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 19-22 1994, vol. 1, pp. 245–248.

[152] B. H. Juang and L. R. Rabiner, "A probabilistic distance measure for hidden Markov models," *AT&T Technical Journal*, vol. 64, no. 2, pp. 391–408, 1985.

[153] L. Xie, V. Ugrinovskii, and I. Petersen, "Probabilistic distances between finite-state finite-alphabet hidden Markov models," *IEEE Transactions on Automatic Control*, vol. 50, no. 4, pp. 505–511, April 2005.

[154] M. Vihola, M. Harju, and P. Salmela, "Two dissimilarity measures for HMMs and their application in phoneme model clustering," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, vol. 1, pp. 933–936.

[155] M. Falkhausen, H. Reininger, and D. Wolf, "Calculation of distance measures between hidden Markov models," in *Proc. of Eurospeech*, 1995, pp. 1487–1490.

[156] M. Do, "Fast approximation of kullback-leibler distance for dependence trees and hidden Markov models," *IEEE Signal Processing Letters*, vol. 10, no. 4, pp. 115–118, Apr 2003.

[157] Z. Rached, F. Alajaji, and L. Campbell, "The kullback-leibler divergence rate between Markov sources," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 917–921, May 2004.

[158] M. Mohammad and W. Tranter, "A novel divergence measure for hidden Markov models," in *Proceedings of the IEEE SoutheastCon*, April 2005, pp. 240–243.

[159] M. Vidyasagar, "Stochastic modelling over a finite alphabet and algorithms for finding genes from genomes," in *Lecture notes in control and information sciences, Conference on Control of uncertain systems*, 2006, vol. 329, pp. 345–369.

[160] J.-Y. Chen, P. A. Olsen, and J. R. Hershey, "Word confusability – measuring hidden Markov model similarity," in *Interspeech*, 2007, pp. 2089–2092.

[161] L. Chen and H. Man, "Fast schemes for computing similarities between Gaussian HMMs and their applications in texture image classification," *EURASIP J. Appl. Signal Process.*, pp. 1984–1993, 2005.

[162] J. Zeng, J. Duan, and C. Wu, "A new distance measure for hidden Markov models," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1550–1555, 2010.

[163] M. Mohammad and W. Tranter, "Comparing distance measures for hidden Markov models," in *Proceedings of the IEEE SoutheastCon*, April 2006, pp. 256–260.

[164] A. Weigend and N. Gershenfeld, Eds., *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley, MA, 1994.

[165] O. Missaoui, H. Frigui, and P. Gader, "Land-mine detection with ground-penetrating radar using multistream discrete hidden markov models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 6, pp. 2080–2099, June 2011.

[166] H. Frigui, L. Zhang, and P. Gader, "Context-dependent multisensor fusion and its application to land mine detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 6, pp. 2528–2543, June 2010.

[167] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA, 2001.

BIOGRAPHICAL SKETCH

Seniha Esen Yuksel has been pursuing her PhD degree in the Department of Computer and Information Science and Engineering at the University of Florida since 2006. Previously, she received her MSc degree from the Department of Electrical and Computer Engineering at the University of Louisville, KY, USA, in 2005, and her BSc degree from the Department of Electrical and Electronics Engineering at the Middle East Technical University, Ankara, Turkey, in 2003. She was the recipient of the UF College of Engineering Outstanding International Student Award in 2010, the winner of the Innovation through Institutional Integration NSF Writing Contest in 2010, and the recipient of the Phyllis M. Meek Spirit of Susan B. Anthony Award at the University of Florida in 2008. She also received travel awards from Association for Computing Machinery (ACM), Computing Research Association (CRA), and Google. Her research interests include machine learning, pattern recognition, and computer vision.