Sign Language Recognition

6.867 Machine Learning Term Paper

Yale Song, Ying Yin
{yalesong, yingyin}@csail.mit.edu

December, 2008

1 Introduction

Our goal here is to recognize a sign language measured from wearable sensor gloves. A sign language is expressed as a sequence of gestural patterns to convey a meaning. Hidden Markov models (HMMs) have been shown to be successful in temporal pattern recognition, such as speech, handwriting, and gesture recognition [4]. In this project, we investigate how well HMMs can perform when applied to sign language recognition. We also investigate how different initialization methods and model selections affect the overall performance on classification.

2 Data Description

We used two different types of datasets for the sign language recognition; one from a single signer and another one from five signers, herein called X_S and X_M, respectively. We obtained the datasets from the UCI Machine Learning Repository [1] [2]. Although the datasets are for Australian Sign Language (Auslan), we believe that the underlying classification method should be generalizable to other sign languages.

The two datasets are multivariate time series of real-valued data. X_S has 27 samples for each 95 signs captured from a native singer using two high-quality position trackers and two Fifth Dimension Technologies (5DT) gloves for both hands [1]. X_M has 70 samples for each 95 signs captured from five signers using only a single low-quality position tracker, the Nintendo Power Glove. The 95 signs were selected to cover all the typical hand shapes and movements [4]. In some cases, signs that were difficult to distinguish were deliberately chosen [4].

Each data sample in X_S consists of a sequence of 22-dimensional feature vectors. The features include x, y, z, roll, pitch, yaw of the hand orientation and five bend measures of the five fingers. Same features are captured for both hands. The data sampling rate of the capturing system is about 100 frames/sec. The average length of each sign is approximately 57 frames [1].

Each data sample in X_M has rather simple data structure; each sample is consisted of a sequence of 8-dimensional feature vectors, measured from right hand, including x, y, z, and roll of the hand orientation, and thumb, forefinger, middle and ring finger bend measures. There are 6650 samples in total (70 samples $\times$ 95 signs). The number of samples measuerd over five signers are not evenly distributed. Each session was taken at a different time, after a break, etc. The data was recorded every 40 ms (25 fps), and the average number of frames per instance is 58, but varies from 30 to 102 [2].

All the samples are labeled with corresponding sign words. Each sample sequence only contains a single sign word.

3 Recognition Approach

Our task is to recognize a sign based on the input sequence of gestural data, so this is a multiclass classification problem (95 classes). We need to identify the basic characteristics of sign language gestures in order to choose a good classification method. Each sign varies in time and space. Also, the signing speed can differ significantly. Even if one person performs the same sign, the speed and position can differ [3]. The ability of HMMs to compensate time and amplitude variations has been shown for speech and character recognition [6]. Due to these characteristics, HMMs appear to be a good approach for sign language recognition. Many literatures on sign language recog-

[1] http://archive.ics.uci.edu/ml/

[2] The datasets are donated by M. W. Kadous who used it for his PhD dissertation (http://www.cse.unsw.edu.au/ waleed/tml/data/) [4].

nition report the successful usage of HMMs for various settings (continous recognition, user independent recognition, camera based data capture [3][7][8]).

3.1 Hidden Markov Models

The positions and orientations of a signer's hand through time can be assumed to follow the first order Markov process [7]. This means the current hands' positions and orientations depend only on the most recent positions and orientations. There exist many kinds of HMMs [6]. One that can model time-series signals whose characteristics change successively over time is called the Bakis model [3] or the Left-Right model [5]. This model is often used in speech recognition systems [3]. The Bakis model allows transitions to the same state, the next state, and the one after the next state. Figure 1 shows an example of the Bakis model with 4 states. With the Bakis model topology, different signing speed can be compensated [3].

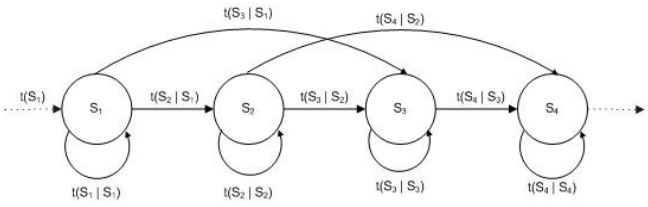

Figure 1: An example of 4-state Bakis model with corresponding transition probabilities

There will be one HMM ($\underline{\theta}_k, k \in \{1...95\}$) trained for each sign k. Two problems of HMMs, namely evaluation and estimation, need to be solved for the recognition system. The probability of an observed sequence $P(\underline{x}_1, ...\underline{x}_m; \underline{\theta}_k)$ will be evaluated for all competing models, and the classification is based on the model that gives the highest log-likelihood. More formally, the classification for an observation sequence $\underline{x}_1, ..., \underline{x}_m$ is:

$$\hat{k} = arg \max_k logP(\underline{x}_1, ...\underline{x}_m; \underline{\theta}_k). \quad (1)$$

3.2 Model Selection

Emission probabilities For the simpler data set X_S involving one signer, a simple Gaussian distribution whose parameters depend on its underlying states is used for defining emission probabilities. More specifically,

$$e(\underline{x} \mid s) = N(\underline{x}; \underline{\mu}_s, \Sigma_s) \quad (2)$$

For the multi-signer dataset, a Gaussian mixture model is used for each state to account for the variance among different people. Hence,

$$e(\underline{x} \mid s) = \sum_{z=1}^{k} q(z \mid s) N(\underline{x}; \underline{\mu}_{s,z}, \Sigma_{s,z}). \quad (3)$$

It is assumed that each of the m states has its own set of mixtures, and k is equal for all states (so there are $k \times m$ mixture components). Equation 2 is just a special case of Equation 3 when k=1.

Model size The choice of the size of a model (the number of states) is an important issue in implementing HMMs. Literature review shows that people have used different number of states in the Bakis model for the gesture recognition. Starner and Pentl [7] used four-state HMMs for all signs, while von Agris et al. [8] used an average of 41 states. The discrepency may lie in the differences of the data sampling rates. Starner and Pentl [7] used a 5 frame/sec rate for data recording, while von Agris et al. [8] used a 25 frame/sec rate.

Intuitively, an underlying state in HMMs represents a particular position, orientation and shape of the hands of the signer in the continuous movement. Using more states means discretizing the whole movement further, which should lead to better accuracy. However, the number of states should be related to the sampling rate (100 frame/sec for X_S dataset and 25 frame/sec for X_M dataset) which affects the number of frames per each sign. Different signs may have different number of states because of the difference in their complexities. Cross validations are used to determine the choice of model size. Due to the time limitation, the same number of states are used for all signs.

Bayesian Information Criterion (BIC) may also give some information about the trade off between fitting the dataset and the complexity of model. The number of independent parameters in the the model G (simple Gaussian distribution with full covariance matrix) is evaluated as

$$dim(G) = 2m + 1 + m \times d + \frac{(1+d) \times d \times m}{2} \quad (4)$$

where m is the number of states, and d is the dimension of a feature vector. Because of the Bakis topology, there are no independent parameters for the initial state parameters, and most of the transition parameters are 0. The results are shown in Table 2 in Section 4.1. However, BIC can not be used as a definite criterion especially when the training data size is small.

3.3 The EM algorithm for HMMs

EM motivation The sign language dataset X_S and X_M contain partially observed data, $\underline{\mathbf{x}}^{(i)} = (\underline{x}_1^{(i)}, ..., \underline{x}_m^{(i)})$ for $i = 1, ..., n$ where n is the number of samples in each dataset ($n = 2565$ for X_S and $n = 6500$ for X_M). The log-likelihood function for the partially observed case is then:

$$L(\underline{\theta}) = \sum_{i=1}^{n} log \sum_{s_1,...,s_m} P(\underline{x}_1^{(i)}, ..., \underline{x}_m^{(i)}, s_1, ..., s_m; \underline{\theta}) \tag{5}$$

If we just take a partial derivatives with respect to its parameters to be estimated, and set them to zero, then we have a coupled nonlinear system of equation. This is known to be very hard to estimate. If we knew the underlying state sequence, $s_1^{(i)}, ..., s_m^{(i)}$, then we could simplify the log-likelihood function as:

$$L(\underline{\theta}) = \sum_{i=1}^{n} logP(\underline{x}_1^{(i)}, ..., \underline{x}_m^{(i)}, s_1^{(i)}, ..., s_m^{(i)}; \underline{\theta}) \tag{6}$$

and we could have directly estimated the maximum-likelihood transition and emission probabilities, $\hat{t}(s'|s)$ and $\hat{e}(x|s)$ as well as initial state probabilities $\hat{t}(s)$.

In a partially observed case, we can efficiently solve the estimation problem using the EM algorithm. The first step in the EM algorithm finds the expected value of $P(\underline{x}_1^{(i)}, ..., \underline{x}_m^{(i)}, s_1^{(i)}, ..., s_m^{(i)}; \underline{\theta})$ given the current parameter values called $\underline{\theta}^{t-1}$ and the observed data. Then the second step in the EM algorithm is to maximize the $Q(\underline{\theta}, \underline{\theta}^{t-1})$ function. We express this as:

$$\underline{\theta}^t = arg \max_{\underline{\theta}} Q(\underline{\theta}, \underline{\theta}^{t-1}) \tag{7}$$

Estimates for simple Gaussian distribution The updates for the t^{th} iteration in the EM algorithm are the following:

$$t^t(s' \mid s) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m-1} p(S_j = s, S_{j+1} = s' \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})}{\sum_{i=1}^{n}\sum_{j=1}^{m-1} p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})} \tag{8}$$

$$t^t(s) = \frac{\sum_{i=1}^{n} p(S_1 = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})}{n} \tag{9}$$

$$\underline{\mu}_s^t = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1}) \underline{x}_j^{(i)}}{\sum_{i=1}^{n}\sum_{j=1}^{m} p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})} \tag{10}$$

$$\Sigma_s^t = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})(\underline{x}_j^{(i)} - \underline{\mu}_s^t)(\underline{x}_j^{(i)} - \underline{\mu}_s^t)^T}{\sum_{i=1}^{n}\sum_{j=1}^{m} p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta}^{t-1})} \tag{11}$$

The posterior probabilities $p(S_j = s \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta})$ and $p(S_j = s, S_{j+1} = s' \mid \underline{\mathbf{x}}^{(i)}; \underline{\theta})$ can be computed using the forward-backward algorithm. Since $\alpha[j, s]$ and $\beta[j, s]$ can quickly get too small, a scaling procedure suggested by Rabiner [5] is used. The procedure is to multiply $\alpha[j, s]$ by a scaling coefficient that is independent of s (i.e. it depends only on j). A similar scaling is done to the $\beta[j, s]$. At the end of the computation, the scaling coefficients are canceled out exactly [5]. The scaled $\hat{\alpha}[j, s]$ is computed as

$$\hat{\alpha}[j, s] = \frac{\sum_{s'} \hat{\alpha}[j-1, s'] \times t(s \mid s') \times e(\underline{x}_j \mid s)}{\sum_{s'}\sum_{s''} \hat{\alpha}[j-1, s'] \times t(s'' \mid s') \times e(\underline{x}_j \mid s'')}. \tag{12}$$

The scaling factor for α is then

$$c[j] = \frac{1}{\sum_{s'}\sum_{s''} \hat{\alpha}[j-1, s'] \times t(s'' \mid s') \times e(\underline{x}_j \mid s'')}. \tag{13}$$

Each $\hat{\alpha}[j, s]$ can then be written as

$$\hat{\alpha}[j, s] = \left[\prod_{l=1}^{j} c[l]\right] \alpha[j, s] = C[j]\alpha[j, s]. \tag{14}$$

Effectively, $\alpha[j, s]$ is normalized at each stage j so that the sum of $\hat{\alpha}[j, s]$ over all s for a particular j is always 1. The same procedure is done for β. The posterior probabilities can be calculated in the same way as before using the scaled $\hat{\alpha}$ and $\hat{\beta}$ because the scaling factors cancel out. For instance,

$$p(S_j = s \mid \underline{x}_1 ... \underline{x}_m; \underline{\theta}) = \frac{\hat{\alpha}[j, s]\hat{\beta}[j, s]}{\sum_{s'} \hat{\alpha}[j, s']\hat{\beta}[j, s']}. \tag{15}$$

The log-likelihood of the observed data can then be computed as

$$logP(\underline{x}_1, ...\underline{x}_m; \underline{\theta}) = -\sum_{j=1}^{m} logc[j]. \tag{16}$$

Parameter updates for the Gaussian mixture model are not presented here (they are derived in Problem Set 5). A MATLAB HMM toolbox[3] which provides implementations for the above algorithms is used for the training and the recognition tasks.

Pre-scaling of data The values for different features in the data have different ranges. In order for the covariance matrices to be meaningful, the data are standardized across each feature to have a mean value of 0 and a standard deviation of 1. This ensures that the components of each feature vector have comparable magnitute.

Initialization Even though the EM algorithm is guaranteed to converge, it can stuck at the local maxima. Therefore, an appropriate way of initializing parameter values is crucial in the EM algorithm.

Additional contstraints on the initial state and transition parameters are applied during the initializaiton so that the model follows the Bakis topology shown in Figure1. The initialization for t(s) is

$$t^0(s) = \begin{cases} 1, s = 1 \\ 0, s \neq 1. \end{cases} \tag{17}$$

The initialization for $t(s' \mid s)$ (m is number of states) is

$$t^0(s'\,|\,s) = \begin{cases} 0, & s' < s \vee s' > s+2 \\ 1/2, & s = m-1 \wedge (s' = m-1 \vee s' = m) \\ 1, & s = m \wedge s' = m \\ 1/3, & \text{otherwise.} \end{cases} \tag{18}$$

The initialization means that each state can transit to itself, the next state, and the one after the next state with equal probability (the last two states are a little bit different). The parameters that are initialized to be 0 will stay at 0 during the EM algorithm.

One of the easiest way to initialize Gaussian mixture models would be randomly assigning all the parameter values. However, this can easily lead to poor performance. Another way would be using k-means clustering algorithm. The k-means clustering is an unsupervised classification method where each point gets assigned to the cluster with the closest center value.

We use k-means clustering to initialize parameters $\underline{\mu}^0$ and Σ^0. For training a particular sign, we first divide each sample sequence equally among the number of states of the model, and group the data from the same state together. Then the k-means algorithm is used to get the $\underline{\mu}^0$ and Σ^0 values for each state. For the covariance matrix, we initialize it mainly with diagonal matrices, but also test with full matrices. The mixture paramters $q(z \mid s)$ are initialized to random values with the constraint that

$$\sum_{z=1}^{k} q(z \mid s) = 1. \tag{19}$$

[3] http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

4 Experiments and Analysis

4.1 Single Signer Dataset

The dataset X_S was collected over a period of 9 weeks. The data are divided into 9 subfolders according to different weeks. Each folder contains 3 samples for each of the 95 signs. One subfolder (285 samples) is set aside for testing, and the remaining 8 subfolders (2280 samples) are used for training. All experiments use 10 iterations in the EM algorithm with full covariance matrix.

A 4-fold cross-validation on the training data is performed to evaluate the effect of the number of states on performance (Table 1). The performance is inversely related to the classification error rate (number of errors / number of test exampls).

Number of States	Error Rate
4	0.2026
30	0.1632
40	0.1610
50	0.1570

Table 1: Cross-validation with different number of states

The error rate decreases with increasing number of states as we expected. However, a complex model may not generalize well for unseen data, and it also significantly slows down the computational speed.

Training is done with all 8 subfolders, and the resutlted models are tested on the 9th subfolder. Table 2 shows the results with different numbers of states.

The result shows that there is a great improvement in the performance when we increase the number of training examples (1 error out of 285 examples for 40 states). Although the test error rate is the lowest for 40 states, the difference is

States	Test Error	Training Error	BIC
4	0.0105	0.0061	-7336
10	0.0281	0.0026	-4361
20	0.0105	0.0004	-6302
30	0.0105	0	-10078
40	0.0035	0	-14107
50	0.0070	0	-18359

Table 2: Test error rates, training error rates, and BIC scores

not very significant due to the limited size of the the testing data. The BIC scores show preference for smaller number of states. However, with the small number of training examples (24 examples per sign), it is hard to make a definite conclusion.

To have a better sense of how the underlying states relate to the observed data, the Viterbi algorithm was run for an observed sequence using the trained HMMs to find the underlying state sequence. Figure 2 in Appendix shows an example for the sign "God". Two features - X and Y coordinates of the right hand - are plotted together with state sequences. The purple line shows the state sequence obtained with 4-state HMMs, and the blue line shows the state sequence obtained with 40-state HMMs which follows the changes in X and Y values more closely.

4.2 Five Signer Dataset

For the second part of our experiment, using X_M, we tried different settings and checked their performances. We tested how different initialization methods and different number of states/mixture-components affect the classification performance. For the initialization methods, we first varied the type of covariance matrix, and then we tested both random (choose centers randomly from data) and k-means clustering for initializing parameter estimates for a mixture of Gaussians. We also varied the numner of states/mixture-components in the test.

4.2.1 Covariance Matrix

We tried with full, and diagonal covariance matrices. $\underline{\mu}^0$ and Σ^0 are chosen randomly from data. Single Gaussian model is used. The result is shown in the Table 3. The result shows that the error rates for a full covariance matrix with HMMs is higher than a diagonal covariance matrix with HMMs. In general, a diagonal covariance matrix is used when we make an assumption

	Full	Diagonal
S=5	0.8256	0.8000
S=8	0.8391	0.8361
S=15	0.7774	0.7774
S=20	0.8286	0.6842
Average	0.8177	0.7744

Table 3: Test error rates on different covariance matrices (M=1, random initialization).

that each element of the feature vector is independent, whereas a full covariance matrix is used where all of the correlations are explictly modelled. A full covariance matrix requires more number of parameters per Gaussian model to be estimated. This increases the complexity of a model and usually result in poorer classification performance when compared to a diagonal covariance matrix. Our result in Table 3 is exactly explained by the same reasoning.

4.2.2 Random vs. K-means Clustering Initialization

Different methods for initializating $\underline{\mu}^0$ and Σ^0 were tested. Table 4 shows error rates for random initialization, and Table 5 shows error rates for k-means initialization. We also varied the number of states (S) and mixture components (M) for both tests. Graphical representations are provided in Appendix (Figure 3 and Figure 4).

	M=1	M=3	M=5	M=10
S=5	0.8000	0.6797	0.7188	0.7565
S=8	0.8361	0.6241	0.6466	0.5887
S=15	0.7774	0.6481	0.5489	0.5451
S=20	0.6842	0.7398	0.8105	0.7526

Table 4: Test error rates on random initialization method. $\underline{\mu}^0$ and Σ^0 are chosen randomly from the data. Diagonal covariance matrices are used.

	M=1	M=3	M=5	M=10
S=5	0.6707	0.6902	0.6316	0.5263
S=8	0.5865	0.5564	0.5526	0.4857
S=15	0.5850	0.4842	0.4977	0.4451
S=20	0.6060	0.5873	0.5489	0.5534

Table 5: Test error rates on k-means initialization. Diagonal covariance matrices are used.

The result shown in Table 4 and Table 5 suggests that using k-means to initialize $\underline{\mu}^0$ and Σ^0 is favorable over randomly choosing values from the

dataset. If $\underline{\mu}^0$ and Σ^0 are randomly chosen, it will take longer for the EM algorithm to converge compared to using k-means. The k-means algorithm chooses the cluster centers ($\underline{\mu}^0$ and Σ^0) by minimizing the clustering error at each step, so it will more likely to set the initial values that better represent the dataset. In our experiment, we limited the number of EM iterations to be 10. The result suggests that 10 steps of EM iterations with X_M might not be enough to converge with randomly chosen $\underline{\mu}^0$ and Σ^0, whereas k-means shortens the amount of time for the convergence.

5 Discussion

The performance gap between X_S and X_M is very large. The biggest reason for this would be the quality of data. Compared to the single signer dataset X_S, the five-signer dataset X_M has very low quality data.

First of all, X_M has 8 features for one hand (right-hand), whereas X_S has 22 features for both hands. In general, most of the sign languages require two hands to convey their meanings. There could be similar signs that are hard to differentiate only using right-hand data. In fact, for the smaller number of signs (5 signs) from X_M, the error rate is less than 0.2. Secondly, the sampling period for X_M is 40ms whereas for X_S is 10ms. This can increase the ambiguity among the dataset X_M because it might have missed some key gestures that characterize the meaning of a sign. Thirdly, X_S was captured from a native signer whereas X_M was captured from five signers (including one novice signer, one sign linguist, two Auslan interpreters, and one native signer). In X_M, the native signer has only recorded 8 samples for each sign while the novice signer recorded 20 samples. This potentially decreases the accuracy of the signs.

Lastly, but most importantly, X_S was measured using high-quality postion trackers and sensor gloves, while X_M was measured using one low-quality sensor glove only. Specifically, X_M has 8-bit accuracy for teh position information which was calculated based on the ultrasound emissions from the glove. This information is particulary susceptible to occasional "spike" caused by random ultrasound noise [2]. For X_S, however, the data of the position and the orientation information have 14-bit accuracy. This means positional error is less than $1cm$ and angle error is less than 0.5 degree. These are the reasons for the huge gap in the error rates between X_S and X_M.

Although the error rate for X_M is relatively high, we should still recognize the merit in the method. For a multiclass classification problem with 95 classes, a "random guess" algorithm - a generic baseline - would get an error rate of 99 per cent [4].

6 Conclusion and Future Work

The experiments on the high quality single singer dataset show promising result on using HMMs for sign language recognition. Further work can be done to improve the method for recognition. Different number of states can be used for for different signs to accommodate different sign complexities. With a newly acquired Cyberglove in our lab (Multimodal Understanding Group, MIT CSAIL), more training data can be collected and tested to provide more statistically significant result. We only presented isolated sign recognition here, however continuous sign language recognition will be the next step to make the system more useful. Sign langauge recognition also provides a good starting point towards natural gesture recognition.

Appendix

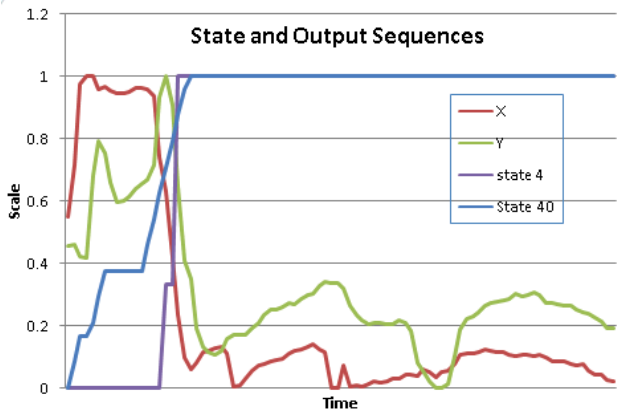

Figure 2: State and observation sequences for the sign "God" showing changes in X and Y coordinates. The blue and purple lines show the state sequences resulted from 40-state and 4-state HMMs respectively. All values are scaled between 0 and 1.

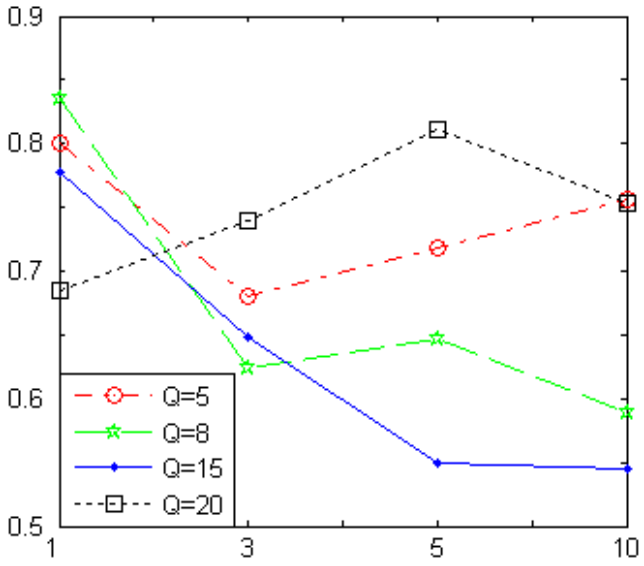

Figure 3: Error rates with random initialization for different number of mixture components (M=1,3,5,10)

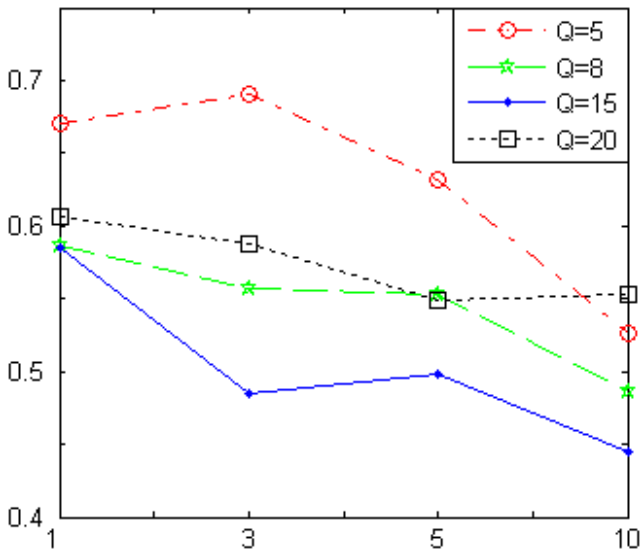

Figure 4: Error rates with K-means initialization for different number of mixture components (M=1,3,5,10)

References

[1] . URL `http://archive.ics.uci.edu/ml/datasets/Australian+Sign+Language+signs+(High+Quality)`.

[2] . URL `http://archive.ics.uci.edu/ml/datasets/Australian+Sign+Language+signs`.

[3] B. Bauer and H. Hienz. Relevant features for video-based continuous sign language recognition. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 440, Washington, DC, USA, 2000. IEEE Computer Society.

[4] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, School of Computer Science and Engineering, University of New South Wales, 2002.

[5] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.

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

[7] T. Starner and A. Pentl. Visual recognition of american sign language using hidden markov models. In *In International Workshop on Automatic Face and Gesture Recognition*, pages 189–194, 1995.

[8] U. von Agris, D. Schneider, J. Zieren, and K.-F. Kraiss. Rapid signer adaptation for isolated sign language recognition. *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 159–159, June 2006. doi: 10.1109/CVPRW.2006.165.