CS 195-5: Machine Learning
Problem Set 6

Douglas Lanman
dlanman@brown.edu
13 December 2006

1 Boosting

Problem 1

In this problem we will derive a few of the properties of the Adaptive Boosting (AdaBoost) algorithm. As presented in the provided pseudocode, Z_m denotes the sum of the weights $W_i^{(m)}$ at the beginning of iteration $m+1$. Begin by showing that the choice of the weighting coefficients α_m minimizes Z_m. Furthermore, show that Z_m is monotonically decreasing as a function of m.

From the provided pseudocode we have the following definitions for the normalization Z_m and weights $W_i^{(m)}$ after iteration m

$$Z_m = \sum_{i=1}^{N} W_i^{(m)} \tag{1}$$

$$W_i^{(m)} = W_i^{(m-1)} e^{-\alpha_m y_i h_m(\mathbf{x}_i)}, \tag{2}$$

where the initial weights are defined to be $W_i^{(0)} = 1/N$. Combining these expressions gives the following form for the normalization Z_m after iteration m.

$$Z_m = \sum_{i=1}^{N} W_i^{(m-1)} e^{-\alpha_m y_i h_m(\mathbf{x}_i)} \tag{3}$$

For this problem we assume that the set of training examples $\{(\mathbf{x_i}, y_i)\}$ are drawn from two classes such that $y_i = \pm 1$. For such two-class classifcation problems, the form of $y_i h_m(\mathbf{x}_i)$ is particularly simple; if an example is correctly classified, then $y_i h_m(\mathbf{x}_i) = 1$. If an example is misclassified, then $y_i h_m(\mathbf{x}_i) = -1$. As a result, Equation 3 can be decomposed as

$$Z_m = W_+^{(m-1)} e^{-\alpha_m} + W_-^{(m-1)} e^{\alpha_m}, \tag{4}$$

where the weighting coefficients $W_+^{(m-1)}$ and $W_-^{(m-1)}$ are defined as

$$W_+^{(m-1)} \triangleq \sum_{i \in \{h_m(\mathbf{x}_i) = y_i\}} W_i^{(m-1)} = \frac{1}{2} \sum_{i=1}^{N} \left(1 + y_i h_m(\mathbf{x}_i)\right) W_i^{(m-1)} \tag{5}$$

$$W_-^{(m-1)} \triangleq \sum_{i \in \{h_m(\mathbf{x}_i) \neq y_i\}} W_i^{(m-1)} = \frac{1}{2} \sum_{i=1}^{N} \left(1 - y_i h_m(\mathbf{x}_i)\right) W_i^{(m-1)}. \tag{6}$$

Note that $W_+^{(m-1)}$ corresponds to the sum of the weights for the correctly-classified examples, whereas $W_-^{(m-1)}$ corresponds to the sum of the weights for the misclassified examples. Also note that $Z_{m-1} = \sum_{i=1}^{N} W_i^{(m-1)} = W_+^{(m-1)} + W_-^{(m-1)}$. At this point, we observe that the minimum (or maximum) of Z_m must occur where the derivative with respect to α_m equals zero. Equating the first partial derivative of Equation 4 with zero, we find

$$\frac{\partial Z_m}{\partial \alpha_m} = \frac{\partial}{\partial \alpha_m}\left\{W_+^{(m-1)} e^{-\alpha_m} + W_-^{(m-1)} e^{\alpha_m}\right\} = -W_+^{(m-1)} e^{-\alpha_m} + W_-^{(m-1)} e^{\alpha_m} = 0$$

$$\Rightarrow \alpha_m = \frac{1}{2}\log\left(\frac{W_+^{(m-1)}}{W_-^{(m-1)}}\right). \tag{7}$$

Now we turn our attention to the form of α_m provided in the pseudocode. Recall that α_m is defined in terms of the weighted training error ϵ_m such that

$$\alpha_m = \frac{1}{2}\log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) \tag{8}$$

$$\epsilon_m = \frac{1}{2}\left(1 - \sum_{i=1}^{N} \frac{W_i^{(m-1)}}{Z_{m-1}} y_i h_m(\mathbf{x}_i)\right). \tag{9}$$

Applying Equations 1 and 6 to Equation 9 gives the following form for the weighted training error.

$$\epsilon_m = \frac{1}{2Z_{m-1}}\left(Z_{m-1} - \sum_{i=1}^{N} W_i^{(m-1)} y_i h_m(\mathbf{x}_i)\right) = \frac{1}{2Z_{m-1}}\left(\sum_{i=1}^{N} W_i^{(m-1)} - \sum_{i=1}^{N} W_i^{(m-1)} y_i h_m(\mathbf{x}_i)\right)$$

$$\Rightarrow \epsilon_m = \frac{1}{Z_{m-1}}\left(\frac{1}{2}\sum_{i=1}^{N}\left(1 - y_i h_m(\mathbf{x}_i)\right) W_i^{(m)}\right) = \frac{W_-^{(m-1)}}{Z_{m-1}}$$

Substituting this expression for ϵ_m into Equation 8 gives the following form for α_m.

$$\alpha_m = \frac{1}{2}\log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) = \frac{1}{2}\log\left(\frac{Z_{m-1} - W_-^{(m-1)}}{W_-^{(m-1)}}\right) = \frac{1}{2}\log\left(\frac{W_+^{(m-1)}}{W_-^{(m-1)}}\right)$$

Since this expression is identical to that obtained by minimizing Z_m with respect to α_m (i.e., Equation 7), we conclude that α_m in the AdaBoost pseudocode (i.e., Equation 8) minimizes Z_m.

To prove that Z_m is monotonically decreasing as a function of m, we begin by substituting Equation 7 into Equation 4.

$$Z_m = W_+^{(m-1)} \exp\left\{-\frac{1}{2}\log\left(\frac{W_+^{(m-1)}}{W_-^{(m-1)}}\right)\right\} + W_-^{(m-1)} \exp\left\{\frac{1}{2}\log\left(\frac{W_+^{(m-1)}}{W_-^{(m-1)}}\right)\right\}$$

$$= W_+^{(m-1)}\sqrt{\frac{W_-^{(m-1)}}{W_+^{(m-1)}}} + W_-^{(m-1)}\sqrt{\frac{W_+^{(m-1)}}{W_-^{(m-1)}}} = 2\sqrt{W_+^{(m-1)} W_-^{(m-1)}}$$

Substituting for Z_{m-1} and ϵ_m reduces this expression to the following form.

$$Z_m = 2\,Z_{m-1}\sqrt{\left(\frac{Z_{m-1} - W_-^{(m-1)}}{Z_{m-1}}\right)\left(\frac{W_-^{(m-1)}}{Z_{m-1}}\right)} = \left(2\sqrt{(1-\epsilon_m)\epsilon_m}\right) Z_{m-1}$$

Recall that the weighted training error must satisfy $0 \leq \epsilon_m < 1/2$. As a result, the coefficient of Z_{m-1} in the previous expression satisfies $0 \leq 2\sqrt{(1-\epsilon_m)\epsilon_m} < 1$ and we conclude the Z_m is monotonically decreasing.

$$\therefore \boxed{Z_m < Z_{m-1} \text{ for } \epsilon_m < 1/2}$$

(QED)

Problem 2

In the previous problem we established that Z_m is monotonically decreasing and that AdaBoost chooses an α_m for which the decrease is fastest. Now show that the training error (i.e., the average number of misclassified training samples) of the combined classifier

$$H_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$$

$$\hat{y}_M(\mathbf{x}) = \text{sign}\left(H_M(\mathbf{x})\right)$$

is bounded from above by Z_M.

Recall that the test error L_N (i.e., empirical loss) can be defined in terms of the 0/1 loss as

$$L_N \triangleq \frac{1}{N}\sum_{i=1}^{N} L_{0/1}\left(y_i, \hat{y}_M(\mathbf{x}_i)\right), \text{ where } L_{0/1}(y,\hat{y}) \triangleq \begin{cases} 0 & \text{if } y = \hat{y}, \\ 1 & \text{otherwise.} \end{cases}$$

From class on 11/22/06 and pages 659-661 of [2], we observe that the empirical loss can be bounded from above using the *exponential loss function* L_{exp}.

$$L_N \leq \frac{1}{N}\sum_{i=1}^{N} L_{\text{exp}}\left(y_i, H_M(\mathbf{x}_i)\right) = \frac{1}{N}\sum_{i=1}^{N} e^{-y_i H_M(\mathbf{x}_i)} \tag{10}$$

To show that the right-hand side of this expression is proportional to Z_M, let's "unroll" the recursive definition of Z_m given in Problem 1. Recall that $W_i^{(0)} = 1/N$ and $Z_0 = \sum_{i=1}^{N} W_i^{(0)} = 1$; substituting for $W_i^{(0)}$ in Equation 2 gives the following expressions for $W_i^{(1)}$ and Z_1.

$$W_i^{(1)} = \frac{1}{N}\, e^{-\alpha_1 y_i h_1(\mathbf{x}_i)} \quad \Rightarrow \quad Z_1 = \frac{1}{N}\sum_{i=1}^{N} e^{-\alpha_1 y_i h_1(\mathbf{x}_i)}$$

Continuing to the next iteration, we obtain the following forms for $W_i^{(2)}$ and Z_2.

$$W_i^{(2)} = \frac{1}{N}\, e^{-y_i(\alpha_1 h_1(\mathbf{x}_i) + \alpha_2 h_2(\mathbf{x}_i))} \quad \Rightarrow \quad Z_2 = \frac{1}{N}\sum_{i=1}^{N} e^{-y_i(\alpha_1 h_1(\mathbf{x}_i) + \alpha_2 h_2(\mathbf{x}_i))}$$

By induction, we conclude that the "unrolled" expressions for $W_i^{(M)}$ and Z_M are as follows.

$$W_i^{(M)} = \begin{cases} \frac{1}{N}\exp\left(-y_i \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x}_i)\right) & \text{if } M > 1 \\ \frac{1}{N} & \text{if } M = 1 \end{cases} = \begin{cases} \frac{1}{N}\, e^{-y_i H_M(\mathbf{x}_i)} & \text{if } M > 1 \\ \frac{1}{N} & \text{if } M = 1 \end{cases}$$

$$\Rightarrow Z_M = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} e^{-y_i H_M(\mathbf{x}_i)} & \text{if } M > 1 \\ 1 & \text{if } M = 1 \end{cases} \tag{11}$$

Substituting Equation 11 into Equation 10 proves the test error L_N is bounded from above by Z_M.

$$\therefore \boxed{L_N \leq Z_M}$$

(QED)

2 Hidden Markov Models

Problem 3

In the following problems we will examine the application of Hidden Markov Models (HMM) to speech recognition. We will begin by implementing the forward-backward algorithm required by the EM procedure for training an M state HMM with N observations. Write the function `fwdback.m` with the signature

```
[alpha, beta, gamma, loglik] = fwdback(p0,P,px),
```

where `p0` is the $M\times1$ vector of initial state probabilities $p_0(s)$, `P` is the $M\times M$ matrix of transition probabilities $p(s \rightarrow s')$, and `px` is the $M\times N$ matrix of emission probabilities $p(\mathbf{x}_t|s_t = s)$. Note that the output arguments `alpha`, `beta`, and `gamma` are $M\times N$ matrices defining the forward, backward, and posterior probabilities, respectively. Finally, `loglik` is the log-likelihood for the HMM defined by the function inputs.

Let's begin by briefly reviewing the forward-backward algorithm, as presented in class on 12/4/06 and in Section 13.2 of [2]. Most importantly, recall that the forward and backward probabilities are defined recursively and must be normalized to prevent arithmetic underflow in practical implementations. As a result, our implementation will use the scaled probabilities defined as

$$\begin{aligned}
\widehat{\alpha}_1(s) &= p_0(s)p(\mathbf{x}_1|s_1 = s)/c_1 \\
\widehat{\alpha}_t(s) &= \frac{1}{c_t}\left[\sum_{s'=1}^{M} \widehat{\alpha}_{t-1}(s')p(s' \rightarrow s)\right] p(\mathbf{x}_t|s_t = s) \\
\widehat{\beta}_N(s) &= 1 \\
\widehat{\beta}_t(s) &= \frac{1}{c_{t+1}} \sum_{s'=1}^{M} p(s \rightarrow s')p(\mathbf{x}_{t+1}|s_{t+1} = s')\widehat{\beta}_{t+1}(s') \\
\gamma_t(s) &= \widehat{\alpha}_t(s)\widehat{\beta}_t(s) \\
\xi_t(s,s') &= c_t\widehat{\alpha}_{t-1}(s)p(\mathbf{x}_t|s_t = s')p(s \rightarrow s')\widehat{\beta}_t(s'),
\end{aligned}$$

where c_t is selected such that $\sum_{s'=1}^{M} \widehat{\alpha}_t(s) = 1$. Recall that the likelihood function is given by the product of the scaling factors such that the log-likelihood has the following form.

$$\log p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{t=1}^{N} \log c_t$$

The included implementation `fwdback.m` evaluates the scaled forward-backward algorithm as defined above. First, on lines 22-25 we ensure that `p0` is in column-vector form. On lines 27-34 we extract the number of states M and number of observations N and allocate storage for the output matrices. The forward probabilities are evaluated on lines 36-46. Note that the scaling factors are determined on lines 44 and 45. Next, the backward probabilities are evaluated on lines 48-56 (using the previously-computed scaling factors). Finally, on lines 58-62 we evaluate the posterior probabilities and the log-likelihood. In conclusion, we have provided a complete implementation of the forward-backward algorithm for use with any HMM.

Problem 4

As described in [5], HMMs can be applied to speech recognition by constraining the form of the transition matrix such that $p(i \to j)$ is zero for $i > j$. The resulting constrained model is known as a *left-right HMM* and can be used to represent the coherent parts of speech (i.e., phonemes). In this situation the path through the states is restricted to subsequent states or remaining in the same state. Show that if the transition probability $p(i \to j)$ is initialized to zero, it will remain zero after each EM iteration. Explain how this fact can be used to implement left-right HMMs.

Recall from class on 12/1/06 and Section 13.2.1 of [2], that the E-step of the EM algorithm (also known as the Baum-Welch algorithm) updates the transition probabilities $p(i \to j)$ using the posterior transition probabilities $\xi_t(i,j)$ obtained using the forward-backward algorithm. Specifically, Equation 13.19 in [2] gives the updated transition probabilities after one EM iteration as follows.

$$p^{\text{new}}(i \to j) = \frac{\sum_{t=2}^{N} \xi_t(i,j)}{\sum_{k=1}^{M} \sum_{t=2}^{N} \xi_t(i,k)}$$

Substituting the expression for $\xi_t(i,j)$ from Problem 3 obtains the following form for the update.

$$p^{\text{new}}(i \to j) = \frac{\sum_{t=2}^{N} c_t \widehat{\alpha}_{t-1}(i) p(\mathbf{x}_t | s_t = j) p(i \to j) \widehat{\beta}_t(j)}{\sum_{k=1}^{M} \sum_{t=2}^{N} c_t \widehat{\alpha}_{t-1}(i) p(\mathbf{x}_t | s_t = k) p(i \to k) \widehat{\beta}_t(k)}$$

In general the denominator of this expression will be non-zero, however the numerator will be zero if $p(i \to j) = 0$. That is, if the transition probability $p(i \to j)$ is zero, then the posterior transition probability $\xi_t(i,j)$ will be zero for all $t = 2, \ldots, N$. As a result, we conclude that if a transition probability is initialized to zero, it will remain zero after each EM iteration. (QED)

From this analysis we now know that the EM algorithm will not modify transition probabilities that are initialized to zero. As a result, we can estimate the parameters of a left-right HMM using the standard Baum-Welch algorithm by initializing the transition matrix $\mathbf{A}$ (where element a_{ij} denotes $p(i \to j)$) such that $a_{ij} = 0$ for $i > j$. More concretely, we should select an upper triangular transition matrix as follows.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1M} \\ 0 & a_{22} & \ldots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{MM} \end{pmatrix}$$

Since the Baum-Welch iterations will not modify the elements initialized to zero, we find the transition probabilities $p(i \to j) = 0$ for $i > j$. In other words, it will never be possible to transition to a state with a lower index – precisely the condition required in a left-right HMM.

Problem 5

Building on the previous results, we now turn our attention to applying HMMs to speech recognition. Specifically, we will build a classifier to distinguish between spoken digits “five” and “nine”. Using the provided implementation of the Baum-Welch algorithm and your version of `fwdback.m`, train a HMM for each of the two digit classes using the included training data. Use $M = 5$ hidden states and $K = 10$ components in the Mixture of Gaussians (MoG) model for each state. Turn in your classification routine, report your test error, and discuss the results.

Before presenting the classification results, let’s review the system architecture. As described in the problem statement, we first transform each training waveform to a *Mel Frequency Cepstral Coefficient* (MFCC) representation using a set of Matlab routines by Dan Ellis [4]. (Recall that the MFCC representation is commonly selected for speech recognition due to its ability to better mimic the human auditory response than FFT or DCT representations.) After obtaining the MFCC feature set, we train a left-right HMM for each digit class. Finally, we construct an optimal Bayes classifier to assign a class label $h^*(\mathbf{x})$ to $\mathbf{x}$ using the estimated HMM parameters θ_c as

$$h^*(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \left\{ \log p(\mathbf{x}|\theta_c) + \log P_c \right\},$$

where P_c is the class prior and $p(\mathbf{x}|\theta_c)$ is the likelihood of the sample $\mathbf{x}$ under the HMM.

The included Matlab script `prob5.m` implements the proposed speech recognition system. On lines 21-34 we process each training sample to obtain the MFCC features. On line 37 we use the provide function `trainDigitHmm.m` to estimate the HMM parameters using the Baum-Welch algorithm. (Note that `kmeans.m` from Problem Set 5 is used to initialize the EM algorithm.) Finally, we classify each test sample using `classifyHMM.m` on lines 39-56. We report the following training and test errors for the speech recognition task.

Training Error	Test Error
0%	2.40%

In conclusion, we find that the HMM correctly classified all training samples. In addition, only 2.40% of test samples were misclassified – a significant result considering the relative simplicity of the left-right HMM. As a result we find that HMMs can be very effective for speech recognition tasks if sufficient labeled training data is available. Note, however, that a practical speech recognition system would have to perform several challenging pre-processing tasks in order to exploit this result, including segmenting individual words from longer sentences.

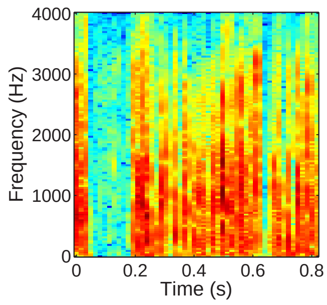
Problem 6

In this problem we will generate synthetic speech samples using the HMMs trained in the previous problem. Write a function `hmmSample.m` that will generate a sequence of artificial observations from a HMM with Mixture of Gaussian (MoG) emission models. Generate five sequences for each digit class. Plot the spectrograms of the synthetic samples and compare to spectrograms generating using test samples. What can you conclude about the ability of the HMM to capture the perceptual aspects of the utterances? Are your findings consistent with the classification results?

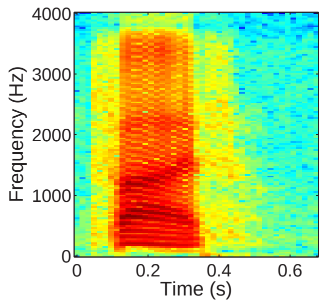
The procedure for sampling from a HMM is very similar to that required for sampling from mixture models. As described on page 613 in [2], we begin by randomly selecting an initial state s_1 with probabilities given by the state priors $p_0(s)$. Afterwards, we synthesize the first observation $\mathbf{x}_1$ using the emission probabilities $p(\mathbf{x}_1|s_1 = s)$. (For this problem each state will have a Mixture of Gaussians emission model.) Next, we randomly select the next state using the transition probabilities $p(s \rightarrow s')$. This process is repeated until the desired number of observations are obtained.

The proposed sampling procedure was implemented using `hmmSample.m`. On lines 15-21 we ensure that the covariance matrices are symmetric positive semi-definite (in order to use `mvnrnd.m` to draw observations from a multivariate Gaussian distribution). Lines 23-38 implement the general HMM-sampling procedure using the provided functions `sample_discrete.m` and `mvnrnd.m`. Note that the output of the sampling procedure will be a synthetic sample in the MFCC representation. Using the provided function `mel2wav.m` we convert each synthetic sample to a waveform representation. Spectrograms for five synthetic and five actual speech samples, for each digit class, are shown in Figures 1 and 2. In the first column of each figure we display the synthetic spectrograms. For comparison we display five actual spectrograms in the second column. Finally, we note that projection and reconstruction from the MFCC representation introduces numerous artifacts. As a result, the final column in each figure shows the second column spectrograms reconstructed from their corresponding MFCC representations. In general, we conclude that the MFCC conversion process “smooths” the spectrums, but otherwise preserves the general features.

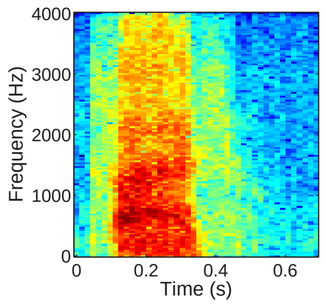
Overall, we find that the synthetic samples only roughly correspond with the actual samples. As with the HMM-based handwriting synthesis presented on page 615 in [2], the generative process only preserves general characteristics of the waveforms within each class. For example, in Figure 1(m) we find that the synthetic sample has at least two identifiable states and that, within each state, the frequency distribution approximately corresponds to the actual samples. In general, these results can be attributed to the inherent limitations of the HMM. Within any state the emission model is a simple Mixture of Gaussians which cannot capture temporal variations. As a result, a large number of hidden states would be required to capture the fine details present in the spectrograms. We conclude by observing that, while the synthetic samples only roughly approximate the true spectrograms, this is not inconsistent with the classification results presented in Problem 5. In particular, the HMMs are effective for determining which of two digit classes is more likely, however they do not have a sufficient number of hidden states to achieve accurate speech synthesis.



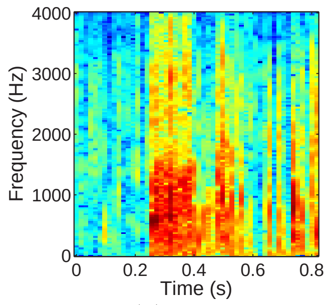
(a) SDL_5A



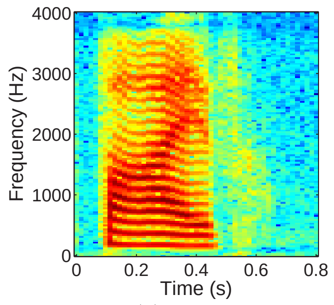
(b) MMI_5B



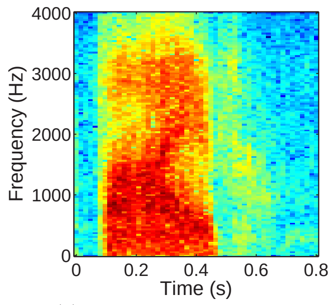
(c) Reconstructed MMI_5B



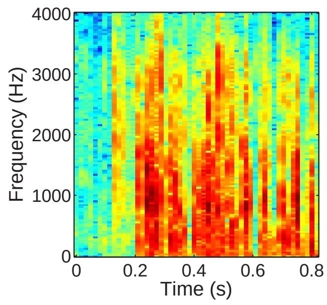
(d) SDL_5B



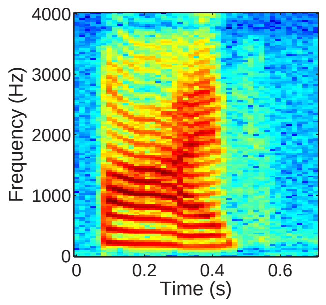
(e) FLL_5A



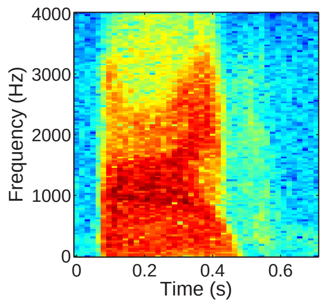
(f) Reconstructed FLL_5A



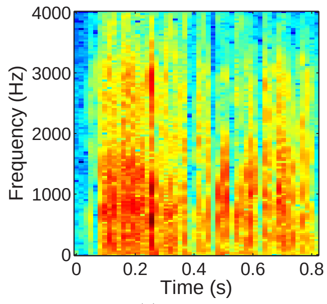
(g) SDL_5C



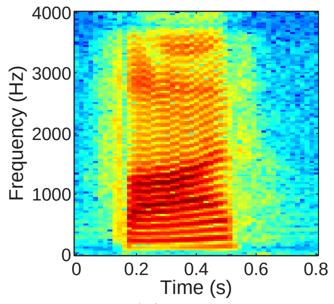
(h) FGK_5B



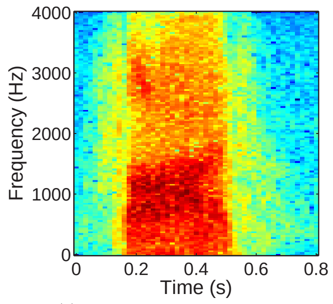
(i) Reconstructed FGK_5B



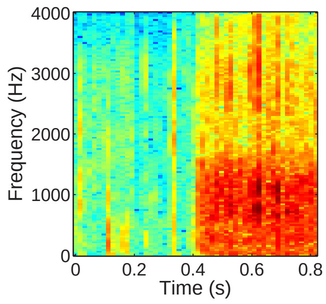
(j) SDL_5D



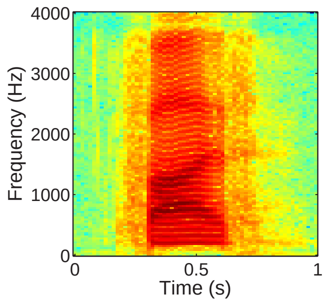
(k) MHC_5A



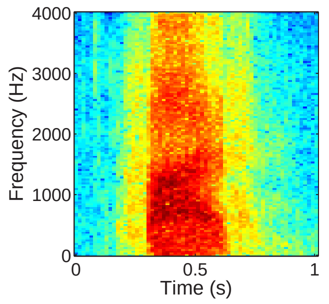
(l) Reconstructed MHC_5A



(m) SDL_5E



(n) MNP_5A



(o) Reconstructed MNP_5A

Figure 1: Comparison of synthetic and recorded spectrograms for the spoken digit "five". The first column shows HMM-generated speech, whereas the second shows actual test samples. The third column shows the test sample spectrograms after reconstructing from their MFCC representations.

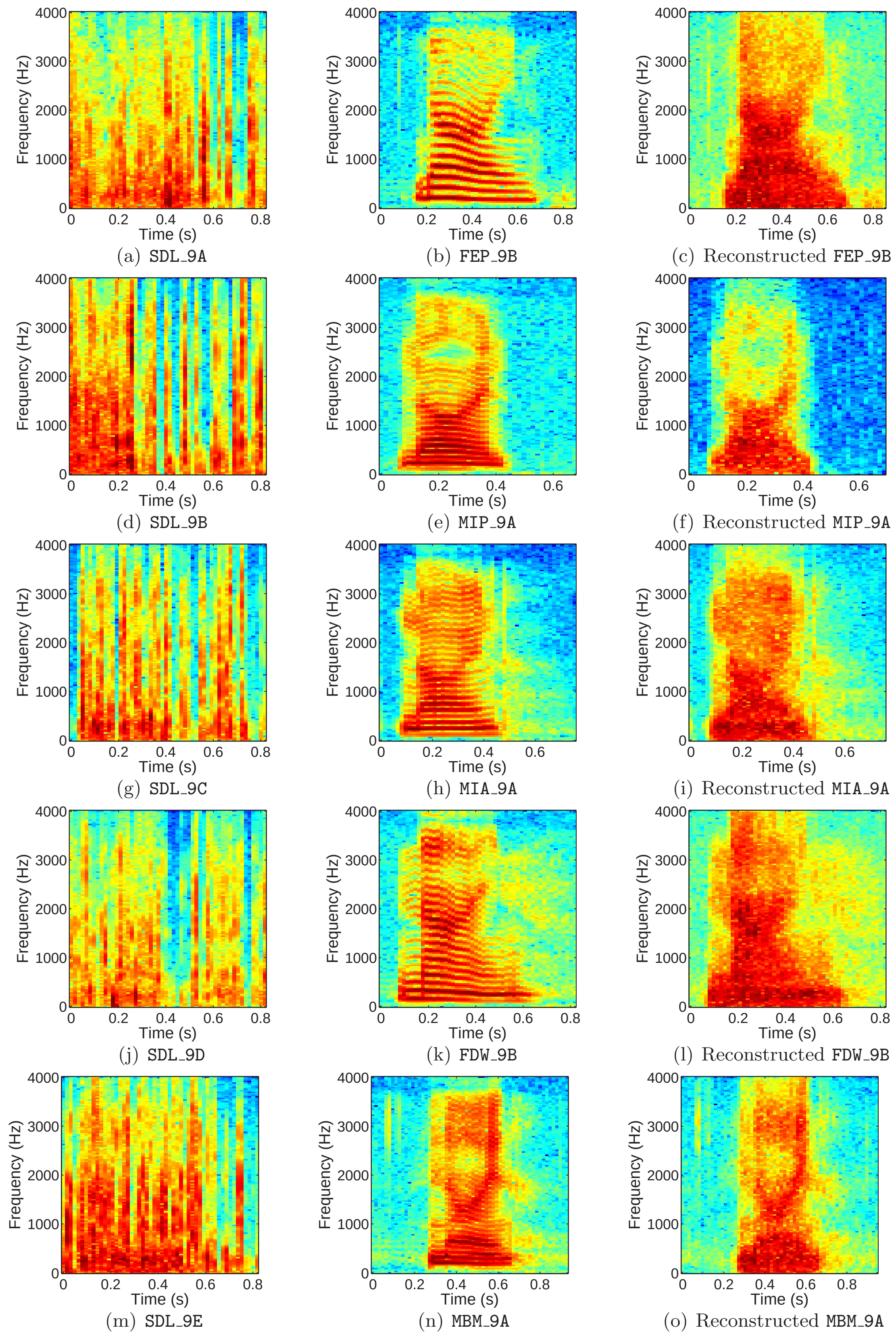


Figure 2: Comparison of synthetic and recorded spectrograms for the spoken digit "nine". The first column shows HMM-generated speech, whereas the second shows actual test samples. The third column shows the test sample spectrograms after reconstructing from their MFCC representations.

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1996.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag, 2006.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (Second Edition).* Wiley-Interscience, 2000.

[4] Dan Ellis. Rasta/plp/mfcc feature calculation and inversion. `http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/`.

[5] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.