

CS195-5: Introduction to Machine Learning
Fall 2006

Problem Set #6 : Avocado

Out: December 1, 2006

Due: December 13, 11am EST

Instructions

In order to identify problem sets to the automated submission system we will assign code names to each problem set. This one will be called “avocado”.

How and what to submit? Please submit your solutions using the automated system in the following way.

1. Create a directory (say, `ps6`) in which you will work on your solutions. All files mentioned below are assumed to reside in that directory.
2. Typeset the written parts of your solution in \LaTeX . The final results should be a document in PDF (preferable) or in PostScript. please name this document `avocadosolution.pdf` or `avocadosolution.ps`, according to the format. **Important: please start the solution of every problem with `\newpage`.**
3. Create Matlab code (extension `.m`) and data (`.mat`) files needed.
4. Create a file named `README`, and include in it a short description of all Matlab files you are submitting.
5. **From the directory `ps6`**, run
`/course/cs195-5/bin/cs195-5handin avocado`

You can resubmit your work as many times as needed, and each subsequent version will override the previous one.

Late submissions: there will be a penalty of 25 points for any solution submitted past the deadline until 11am on Friday, Dec 15. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it; if multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important.

1 Boosting

In this problem we will derive a few of the quantities and properties in the AdaBoost algorithm. The algorithm is summarized in Figure 1; note that in contrast to the formulation in the lecture we have omitted the normalization of the weights. This does not affect the performance of the algorithm (we do normalize the weights when computing the weighted error).

Problem 1 [15 points]

In this modified version of AdaBoost, Z_m stands for the sum of weights in the beginning of iteration $m + 1$. Show that the choice of α_m in AdaBoost minimizes Z_m . Furthermore, show that Z_m is monotonically decreasing as a function of m .

End of problem 1

Advice: Look at Z_m as a function of α_m , and find the value for which the function achieves its minimum.

We have established that Z_m is decreasing, and that AdaBoost chooses α_m with which that decrease is fastest. We now will see why this is important.

Problem 2 [15 points]

Show that the training error (the average number of misclassified training examples) of the combined classifier after m iterations of boosting,

$$\hat{h}_m(\mathbf{x}) = \sum_{t=1}^m \alpha_t h_t(\mathbf{x}),$$

is bounded from above by Z_{m+1}/N .

End of problem 2

Input: Set of n labeled examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$, $y_i = \pm 1$.

Set of associated weights $W_i^{(0)} = 1/N$, for $i = 1, \dots, N$.

Required number of iterations, M .

for $m = 1, \dots, M$ **do**

Let $Z_{m-1} = \sum_{i=1}^n W_i^{(m-1)}$.

Find a weak classifier h_t , which outputs binary predictions $h_t(\mathbf{x}) = \pm 1$, such that its weighted training error

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

satisfies $\epsilon_m < 1/2$.

Set the vote $\alpha_m = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$.

Update the weights : for each $i = 1, \dots, N$ set

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

end for

Output: the combined classifier defined by

$$\text{sign} \left(\sum_{m=1}^M \alpha_m h_m(\mathbf{x}) \right)$$

Figure 1: AdaBoost, slightly modified from Lecture 29.

Advice: Note that the error of the combined classifier is not weighted, i.e. it is just the ratio of misclassified examples to N .

In the definition of the algorithm, the weights (and therefore Z_m) are defined recursively. You will find it helpful to “unroll” the definition, and write out the value of Z_{m+1} in terms of the initial weights (which are all 1) and the subsequent updates, as a function of $\alpha_1, \dots, \alpha_m$ and h_1, \dots, h_m . You may also find the following (simple) fact helpful: for any $x \geq 0$, $e^x \geq 1$.

2 Hidden Markov Models

In this part of the problem set we will look at the application of HMM to speech recognition. An excellent tutorial on this topic by L. Rabiner is posted on the course website; here we will briefly describe the details necessary to proceed with the assignment. Before you start experimenting with the data, please add to your path¹ the directory tree at `/course/cs195-5/matlab/speech`; the speech files reside under `/course/cs195-5/data/speech`. The archive containing the code and data is also available on the course website. We will be using a Matlab Dan Ellis

The “raw” representation for speech signal consists of the recording of the acoustic waveform, sampled at a given sampling rate. We will work with `wav` files, which contain such sampled waveforms in a standard format. In Matlab, we can read in and display an “acoustic clip” as follows:

```
[wf,sr] = wavread('MAE_9B.wav');  
figure;plot((1:length(wf))/sr,wf);
```

This produces the plot in Figure 2. The sampling rate (variable `sr`) in the file is 8,000 Hz, and the length of the utterance (i.e. the recorded piece of speech) is about 0.8 seconds. The recording therefore contains thousands of samples, and in order to see the details we would need to zoom in on the figure, as shown in Figure 2 on the right.

A more explicit visualization of the waveform can be obtained by plotting its *spectrogram*:

```
specgram(wf, 256, sr);
```

The result, shown in Figure 3, describes the energy of different frequency components of the wave estimated within a 256 samples window sliding along time (with 128 samples step).

In addition to visualizing the waveform, we can listen to the recorded sound (it is speech, after all):

```
soundsc(wf,sr);
```

We can now take this raw representation and transform it using the chosen feature set. A commonly used features in speech recognition are *Mel Frequency Cepstral Coefficients* (MFCC). In the provided Matlab implementation, these are computed as follows (we will be using the

¹See help on `addpath` and `genpath`

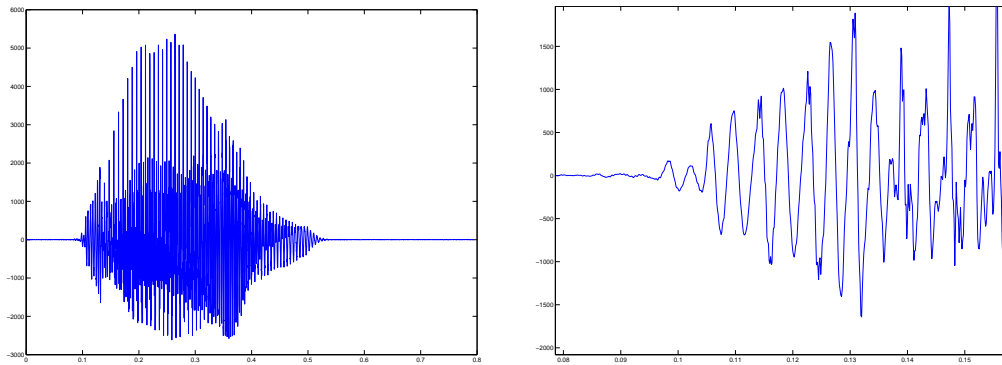


Figure 2: The waveform read from the .wav file. Plot on the right is zoomed on the area between 80 and 150 milliseconds.

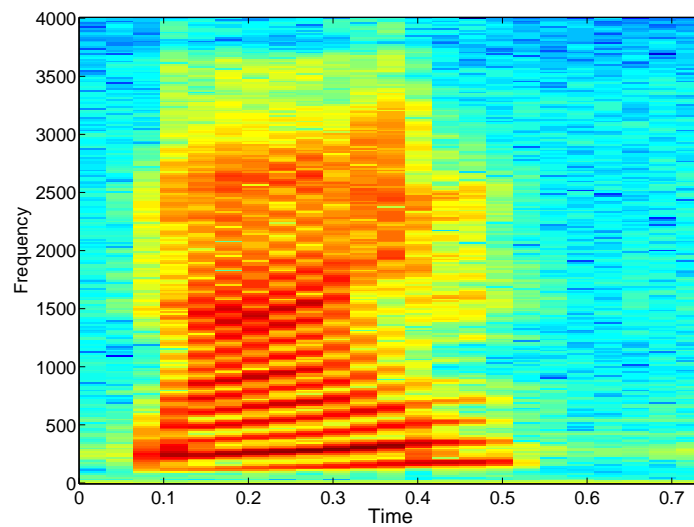


Figure 3: Spectrogram of an utterance.

```
x=melfcc(wf, sr, 'wintime', .01, 'hoptime', .005);
```

This will yield the default of 13 coefficients computed in 100 ms windows taken at each 50 ms. We encourage you to read elsewhere about the MFCC representation; however these details are beyond the scope of our class and we will treat MFCC as a “black box” that takes in the acoustic wave and

produces a sequence of multidimensional measurements.

We have provided you with a function `readMFCC.m` that reads in the waveforms and computes MFCC for all `wav` files in the current directory, using specific values for a number of parameters; in particular. See comments in the file for usage details. We also have provided you with the Matlab code necessary to learn parameters of HMMs; there is however one missing piece you will need to fill in.

Problem 3 [25 points]

Write the function `fwdback.m` that will implement the forward-backward algorithm. The function signature is as follows:

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

The input arguments are described below. We denote by K for the number of components in the Gaussian mixture model of $p(\mathbf{x} | s)$, M the number of states, and N the number of observations in the sequence.

`p0` the $M \times 1$ vector of initial state probabilities, $p(s_1)$.

`P` the $M \times M$ transition probability matrix \mathbf{P}

`px` the $M \times N$ matrix containing the likelihood $p(\mathbf{x}_i | s)$ for each observation \mathbf{x}_i , $i = 1, \dots, N$ and state $s = 1, \dots, M$.

The output arguments (see the lectures for definitions and details):

`alpha` the $M \times N$ matrix of forward probabilities, $\alpha_t(s)$

`beta` the $M \times N$ matrix of backward probabilities, $\beta_t(s)$.

`gamma` the $M \times N$ matrix of state posteriors, $\gamma_t(s)$.

`loglik` the log-likelihood of $\mathbf{x}_1, \dots, \mathbf{x}_N$ (a scalar).

Turn in the code of the function, with reasonable documentation inside.

End of problem 3

Advice: Code used in the following parts of the assignment relies on `fwdback`, so you should test it on some simple toy example, such as a discrete HMM with binary outputs, for which you can easily verify the results. Note that `px` for the MoG HMM can be computed using the provided function `mixgauss_prob.m`.

We are ready now to experiment with HMMs for speech recognition. Specifically, we will build a classifier that will discriminate between spoken digits “five” and “nine”. For obvious reasons, these two digits are the most confused among ten spoken digits; moreover, we will develop a speaker-independent recognition system, i.e. the classifier won't be restricted to the user(s) on whose speech it was trained. On the other hand, problem settings we will work with are very simple: the utterances are clean (there is no significant noise), and the digits, in both training and test data, are *isolated*. Each utterance is given in its own `wav` file, i.e. we don't face the challenging problem of word segmentation.

The states of the HMM are meant to represent the coherent parts of the utterance, roughly corresponding to *phonemes*. We can number these states according to the order of the corresponding parts in the utterance. The path through the states should be restricted to transitions to subsequent states (with the possibility, of course, of remaining in the same state), and never to past states. The resulting constrained model is known as *left-right HMM*.

Problem 4 [10 points]

Show that if the transition probability $p(i \rightarrow j)$ is initialized to zero, it will remain zero after the EM has converged; explain how this fact can be used to produce left-right HMM.

End of problem 4

Problem 5 [20 points]

Train an HMM for each of the two classes on the data in `/course/cs195-5/data/speech/train`.

Use the provided function `trainDigitHmm.m` with $M = 5$ states and $K = 10$ component in the mixture of Gaussian model for each state. Note: you will need `kmeans.m` you had to write for Problem Set Rambutan (it is used here to initialize the Gaussian means).

Write the function `classifyHmm`

```
yhat = classifyHmm(x,hmms);
```

that predicts the label of utterance `x`, represented in MFCC domain as returned by `readMFCC`, using class-conditional HMMs in `hmms` as returned by `trainDigitHmm`. Turn in the function, Matlab code that uses it to classify the utterances in `/course/cs195-5/data/speech/test`, and report the test

error.

End of problem 5

Advice: We have provided `mhmm_logprob.m` that calculates the log-likelihood of data under the HMM model; this should prove useful in solving the problem above.

HMM is a generative model, and it would be interesting to examine how well it actually captures the distribution of features. One way to assess that is *tosample* from the model.

Problem 6 [15 points]

Write a function `hmmSample.m`

```
x = hmmSample(hmm,N);
```

that will generate a sequence of “fake” observations from HMM with a mixture of Gaussians model of emission probabilities.

Generate five sequences for each class. You can now convert them to wavforms using the provided `mel2wav.m`. Do that, and plot the spectrograms of the resulting “fake” utterances using `specgram`. Also, select (randomly) five real utterances from each class, and plot their spectrograms. What can you say about the ability of HMM to capture the perceptual aspects of the utterances? Are your findings consistent with the classification results?

End of problem 6

Advice: The provided function `sample_discrete.m` and the standard `mvnrnd` will be helpful in generating the samples from relevant distributions.