Lab 1 Sample Code

```
function [confusion, errrate] = tidigitsasr(CorpusDir, k)
% confusion = tidigitsasr(CorpusDir, k)
% Given the root directory for the TIDIGITS corpus, train VQ models
% with k codewords and evaluate them against test data.
%
% Expected subdirectories:
%   mfc/train - training features
%   mfc/test - test features
%
% Returns a confusion matrix and error rate [0,1]
% Confusion matrices are 10x10 matrices representing the following classes:
%   { zero or oh, one, two, three, four, five, six, seven, eight, nine }
%
% confusion(3,4) means How many things that were actually class 3
% were predicted as class 4?

classes = 'z123456789';

traindir = fullfile(CorpusDir, 'mfc', 'train');

for cidx = 1:length(classes)
    % Find names of training files for this class.
    omega = classes(cidx);
    featurefiles = findfiles(traindir, [omega '._01\.mfc'], 'regexp');
    if strcmp(omega, 'z')
        % Zero is a special case, also include people saying "O"
        featurefiles = [featurefiles;
            findfiles(traindir, ['o' '._01\.mfc'], 'regexp')];
    end
    % Load the training data
    features = loadfeatures(featurefiles);

    fprintf('Training model %s\n', classes(cidx));
    % Train the model
```

```
    models{cidx} = VQ_Train(k, features);

end


confusion = zeros(length(classes), length(classes));

testdir = fullfile(CorpusDir, 'mfc', 'test');
for cidx = 1:length(classes)
    % Load test files for this class
    omega = classes(cidx);
    featurefiles = findfiles(testdir, [omega '._01\.mfc'], 'regexp');
    if strcmp(omega, 'z')
        % Zero is a special case, also include people saying "O"
        featurefiles = [featurefiles;
            findfiles(testdir, ['o' '._01\.mfc'], 'regexp')];
    end

    distortions = zeros(length(classes), 1);
    fprintf('Testing %d tokens of class %s\n', length(featurefiles), omega);
    for tidx =1:length(featurefiles)
        % Read in test token's features
        features = spReadFeatureDataHTK(featurefiles{tidx});
        % Determine score against each model
        for midx = 1:length(classes)
            distortions(midx) = VQ_MeanMinDistortion(features, models{midx});
        end
        % Make decision
        [~, decision] = min(distortions);
        % Note decision
        confusion(cidx, decision) = confusion(cidx, decision) + 1;
        if cidx ~= decision
            % If we wanted to record which ones were misclassified, this
            % would be the place to do it.
        end
    end
```

```
end

[errrate, incorrect, N] = errorrate(confusion);
fprintf('k=%d Error %.3f (%d/%d)\n', k, errrate, incorrect, N);
```

```
function CodeWords = VQ_Train(NCodeWords, TrainingData, StopFraction)
% CodeWords = lloydvq(NCodeWords, TrainingData, StopFraction)
%
% Given a matrix TrainingData where each column consists of one training
% sample, create a vector quantizer with NCodeWords.
%
% Training is done with the iterative Lloyd algorithm which is said to
% converge when the ratio of the current iteration's distortion to the
% previous one is StopFraction x 100%.  StopFraction defaults to .99 when
% it is not specified.

if nargin < 3
    StopFraction = .99;  % set default
end

[Dim, N] = size(TrainingData); % N samples of dimension Dim


CodeWords = VQ_InitCodebook(NCodeWords, TrainingData);

% Prime the loop --------------------

% Fudge a large improvement between the "previous iteration"
% (non-existant) and the current one so that we enter the loop and have
% "previous" values set correctly.
PrevAvgDistortion = Inf;
DistortionRatio = 0;

% distortion(i, j) will contain the distortion between
```

```
% the i'th codewords and the j'th training vector.
distortions = distortion3(TrainingData, CodeWords);

% By examining each column of distortion, we find the index
% of the smallest entry.  MinDist tells us how far we are
% from the smallest codeword and MinDistIdx tell us which
% codeword produced the smallest distortion.  Note that we
% explicitly tell min to operate on columns so that if
% a single codeword is specified we will not take the minimum
% of all distances.
[MinDist, MinDistIdx] = min(distortions, [], 1);

% Average distortion is the mean of the distortion between
% each training vector and the minimal distortion codeword.
AvgDistortion = mean(MinDist);

% Iterate until good enough --------------------
iteration = 0;
while DistortionRatio < StopFraction

  % Generate new codebook
  for cwidx = 1:NCodeWords

    % Find new centroid of each partition:
    % Take mean value across each row for vectors in partition cwidx.
    %

    % Handle single codewords case...
    PartitionIndices = find(MinDistIdx == cwidx);
    if ~ isempty(PartitionIndices)
      CodeWords(:, cwidx) = mean(TrainingData(:,PartitionIndices), 2);
    end
  end

  % Compute the distortions and set up for next
  % iteration through loop.
```

```
  distortions = distortion3(TrainingData, CodeWords);
  [MinDist, MinDistIdx] = min(distortions, [], 1);

  % Compute new mean distortion.  We could have called VQ_MeanEucDist,
  % but since we already have the information, itit seems simpler to just
  % take the mean.
  AvgDistortion = mean(MinDist);

  DistortionRatio = AvgDistortion / PrevAvgDistortion; % New/old
  PrevAvgDistortion = AvgDistortion;  % What is new becomes old

  iteration = iteration + 1;
  fprintf('iteration %d Mean distortion %f, ratio to previous %f%%\n', ...
          iteration, AvgDistortion, DistortionRatio*100);
end
```

```
function mu = VQ_MeanMinDistortion(ColVecs, Codebook)
% VQ_MeanMinDistortion(ColVecs, Codebook)
%
% Compute the average minimum distortion of all column vectors ColVecs
% with respect to the given Codebook (column vector codewords).

% Compute pair wise distortions
% distortions(i,j) is distortion between ColVecs(:,i) and Codebook(:,j)
distortions = distortion3(ColVecs, Codebook);

% For each ColVec i, min distortion codeword is one with smallest
% distortion.  Take average of smallest distortions
mu = mean(min(distortions));
```