PURPOSE
To gain an understanding of feature extraction, windowing, MFCCs.

SECTION 1 SEGMENTING INTO EVERY N ms FRAMES

Segmenting: Chopping up into frames every N seconds
Previously, we’ve either chopped up a signal by the location of its onsets (and taking the following 100 ms) or just analyzing the entire file.

Analyzing the audio file by “frames” is another technique for your arsenal that is good for analyzing entire songs, phrases, or non-onset-based audio examples.

You easily chop up the audio into frames every, say, 100ms, with a for loop.

```matlab
frameSize = 0.100 * fs;  % 100ms
for i = 1: frameSize : (length(x)-frameSize+1)
    currentFrame = x(i:i+frameSize-1);  % this is the current audio frame
    % Now, do your feature extraction here and store the features in some matrix / array
end
```

Very often, you will want to have some overlap between the audio frames - taking an 100ms long frame but sliding it 50 ms each time. To do a 100ms frame and have it with 50% overlap, try:

```matlab
frameSize = 0.100 * fs;  % 100ms
hop = 0.5;  % 50% overlap
for i = 1: hop * frameSize : (length(x)-frameSize+1)
    currentFrame = x(i:i+frameSize-1) + eps;  % this is the current audio frame
    % Note that we add EPS to prevent divide by 0 errors
end
```

Note that it’s also important to multiple the signal by a window [e.g., Hamming / Hann window] equal to the frame size to smoothly transition between the frames.

SECTION 2 MFCC

Load an audio file of your choosing from the audio folder on /usr/ccrma/courses/mir2012/audio
Use this as an opportunity to explore this collection.

BAG OF FRAMES
Test out MFCC to make sure that you know how to call it. We’ll use the CATbox implementation of MFCC.

```matlab
currentFrameIndex = 1;
for i = 1: frameSize : (length(x)-frameSize+1)
    currentFrame = x(i:i+frameSize-1) + eps;  % this is the current audio frame
    % Note that we add EPS to prevent divide by 0 errors
```
% Now, do your other feature extraction here

% The code generates MFCC coefficients for the audio signal given in the current frame.
[mfceps] = mfcc(currentFrame,fs); %note the transpose operator!
delta_mfceps = mfceps - [zeros(1,size(mfceps,2)); mfceps(1:end-1,:)]; %first delta

% Calculate the mean and std of the MFCCs, MFCC-deltas.
MFCC_mean(currentFrameIndex,:) = mean(mfceps);
MFCC_std(currentFrameIndex,:) = std(mfceps);
MFCC_delta_mean(currentFrameIndex,:) = mean(delta_mfceps);
MFCC_delta_std(currentFrameIndex,:) = std(delta_mfceps);
currentFrameIndex = currentFrameIndex + 1;
end

features = [MFCC_mean MFCC_delta_mean]; % In this case, we'll only store the MFCC and delta-MFCC means
% NOTE: You might want to toss out the FIRST MFCC coefficient and delta-coefficient since it's much larger than
% others and only describes the total energy of the signal.

You can include this code inside of your frame-hopping loop to extract the MFCC-values for
each frame.

Once MFCC's per frame have been calculated, consider how they can be used as features for
expanding the k-NN classification and try implementing it!

Extract the mean of the **12 MFCCs (coefficients 1-12, do not use the "0th" coefficient)** for
each onset using the code that you wrote. Add those to the feature vectors, along with zero
crossing and centroid. We should now have 14 features being extracted - this is starting to get
"real world"! With this simple example (and limited collection of audio slices, you probably won't
notice a difference - but at least it didn't break, right?) Try it with the some other audio to truly
appreciate the power of timbral classification.

**SECTION 3 CROSS VALIDATION**
You'll need some of this code and information to calculate your accuracy rate on your classifiers.

**EXAMPLE**
Let's say we have 10-fold cross validation...

a. Divide test set into 10 random subsets.

b. 1 test set is tested using the classifier trained on the remaining 9.

c. We then do test/train on all of the other sets and average the percentages.

To achieve the first step (divide our training set into k disjoint subsets), use the function
crossvalind.m (posted in the Utilities)

INDICES = CROSSVALIND('Kfold',N,K) returns randomly generated indices
for a K-fold cross-validation of N observations. INDICES contains equal
(or approximately equal) proportions of the integers 1 through K that
define a partition of the N observations into K disjoint subsets.

You can type help crossvalind to look at all the other options.
% This code is provided as a template for your cross-validation computation. Replace the variables “features”, ”labels” with your own data.
% As well, you can replace the code in the "BUILD" and "EVALUATE" sections to be useful with other types of Classifiers.
%

%% CROSS VALIDATION
numFolds = 10;                      % how many cross-validation folds do you want - (default=10)
numInstances = size(features,1);     % this is the total number of instances in our training set
numFeatures = size(features,2);     % this is the total number of instances in our training set
indices = crossvalind('Kfold',numInstances,numFolds)   % divide test set into 10 random subsets

clear errors
for i = 1:10
   % SEGMENT DATA INTO FOLDS
   disp(['fold: ' num2str(i)])
   test = (indices == i) ;    % which points are in the test set
   train = ~test;       % all points that are NOT in the test set

   % SCALE
   [trainingFeatures,mf,sf]=scale(features(train,:));

   % BUILD NEW MODEL - ADD YOUR MODEL BUILDING CODE HERE...
   model = knn(numFeatures,2,3,trainingFeatures,labels(train,:));

   % RESCALE TEST DATA TO TRAINING SCALE SPACE
   [testingFeatures]=rescale(features(test,:),mf,sf);

   % EVALUATE WITH TEST DATA - ADD YOUR MODEL EVALUATION CODE HERE
   [voting,model_output] = knnfwd(model ,testingFeatures);

   % CONVERT labels(test,:) LABELS TO SAME FORMAT TO COMPUTE ERROR
   labels_test = zeros(size(model_output,1),1); % create array of 0s
   labels_test(find(labels(test,1)==1))=1;  % convert column 1 to class 1
   labels_test(find(labels(test,2)==1))=2;  % convert column 2 to class 2

   % COUNT ERRORS
   errors(i) = mean ( model_output ~= labels_test )
end

disp(['cross validation error: '  num2str(mean(errors))])
disp(['cross validation accuracy: ' num2str(1-mean(errors))])

Copyright 2012 Jay LeBoeuf
Portions can be re-used by for educational purposes with consent of copyright owner.