Project in Introduction to Machine Learning

Phoneme Classification

1. Introduction

The project’s goal is to automatically classify basic speech units that make up words; these units are called **phonemes**. In most languages, a phoneme is a consonant or vowel (for further reading http://en.wikipedia.org/wiki/Phoneme). The input is a vector of features that contains the measurements of the instantaneous spectrum in a specific time frame of a spoken English speech signal recording. The instantaneous spectrum is not represented by the Fourier coefficients of the signal in a specific time frame, but is calculated from them. This representation is called MFCC (for further reading http://en.wikipedia.org/wiki/Mel-frequency_cepstrum). The features vector contains the first 13 coefficients of the MFCC, their first and second time derivative, the overall energy in the time frame and its first time derivative. The dimension of the feature vector is therefore 41.

The output of the system (the prediction) has binary values, which represent the classification of an input sample to the first or second class, which are consonants and vowels (which means that this is **binary classification problem)**.

The purposes of the project are: (1) To use the training set data in order to build a binary classifier and (2) to evaluate the classifier’s performance for speakers who aren't it the training set.

In this project, you will use data from a database called TIMIT, which consists of recordings of various speakers who read different sentences. You are not required to use the recordings themselves but just the features and the binary labels, which were calculated for you in advance.

2. Data

In the course website, you can find the file that you will need in the first stages of the project:

1) `data.mat` - To load this file into MATLAB use the command : `load('data.mat')`
This file contains both the training set data (in the trainSet variable) and the testing set data (in the `testSet` variable). There is no overlap between the training and testing sets, therefore the testing set will be used to evaluate the ability of the classifier to generalize to new samples not seen in the training phase.

Both variables `testSet` and `trainSet` are of type struct, each of them contains the following fields:

- X- A matrix sized DxN, where D is the number of features and N is the number of examples (i.e. each column is a vector that contains all feature values for a specific example)
- Y – Labels vector of consonants and vowels, where the value 1 indicates vowel and a value of 0 indicates a consonant (or other non-vowel frame, e.g. silence etc.).

Before you begin with the tasks, we advise you load the data file into MATLAB to observe the given data. Make sure you understand the meaning of the different variables.

3. Model Parameters Selection and Performance Evaluation

Parameters Selection

For some of the algorithms which you will use during the project, you will need to choose parameters manually (as opposed to the parameters that are learned/estimated during the training phase). For example, in K-NN classification you will need to decide how many neighbors to use.

Parameter selection is performed on the training set (trainSet) as follows:

- Set the value of the parameter you need to choose to some value $K = K_0$.
- Divide the training examples into $N = 10$ disjoint groups of approximately equal size. For each of these groups ($i = 1, \ldots, N$) perform the following process:
  - ➢ Train the classifier on all the examples except for group i (i.e. on 90% of the examples).
  - ➢ Calculate the empirical error (percentage of examples that are misclassified) over group i (i.e. on the remaining 10% of the examples)
- Compute the mean and standard deviation of the empirical error, over the 10 trials.
- Repeat this process for all values of the parameter you need to choose, $K_0, K_1, \ldots$.

  Report the mean and standard deviation of the errors (use a graph to display the results). Select the parameter which gives the minimal average classification error.

Note: The above description applies to choosing a single parameter. If you have to choose more than one parameter, you need to do the same process for each set of values. For example, if parameter K can get values $[k_1, k_2]$ and parameter L can get values $[l_1, l_2]$, you have to repeat the process 4 times, once for every possible pair of values.

This method is called 10-fold (or generally, N-fold) **cross validation** and it is a general method for evaluating the performance of a classifier.

Performance Evaluation

Once you have chosen the parameters, you should use the testing set (testSet) to evaluate the classifier's performance . Train the classifier again on all of the training set (trainSet), and calculate the empirical error on test set (testSet). Compare to the error on the training set.

4. Binary Classification

In this project, you need to perform binary classification. Performance evaluation should be performed on the testing set in order to check the generalizability of the model.

First Task – KNN Classifier

In this section, you need to examine the performance of the K-Nearest Neighbors algorithm. Some important points regarding this part:

- In this approach, the system is not learning a model, but only memorizing the training set examples, which are used only during test phase. You should evaluate the generalizability of the KNN algorithm.
- There are at least two degrees of freedom (parameters) in choosing the classifier. One of them is K, which is the number of neighbors that are used in the prediction step. Another parameter you need to choose is the metric used to calculate the distance between the examples. There is no need to examine values of K greater than 30. In addition it is recommended to check only odd values of K in order to ensure that there will always be a decision.
- You can use existing implementation of KNN (i.e. MATLAB `knnclassify` function, see the documentation of the function).

- Before you begin your experiments, think: which graph will best illustrate the tests you perform, and the comparison between them? What will be the X-axis and Y-axis? What curves will you show on the graph?

To be submitted :

1. Put your experimental results in a table or graph and explain the results obtained. Specify the value K and the metric you have chosen and explain. (You need to compare the metric you have chosen with at least one additional metric)
2. Suggest a way to improve the performance of the system.
3. If you chose to implement the algorithm yourself, attach a well-documented code.
4. If you chose to use MATLAB implementation, attach the call for the MATLAB function. **Please note that the calling code should contain values for all the options that the function allows, even if it is the default value**. Explain the values you selected.

Second Task – Naïve Bayes Classifier

In this section, you will use the training data to train a naïve Bayes classifier.
The parameter you need to set manually (that is, to choose using cross-validation) is the type of the conditional distribution. Select the distribution that minimizes the empirical error. You may use the MATLAB functions `NaiveBayes.fit` and `NaiveBayes.predict`. Read the documentation of these functions, and explain the options you chose (note that the property 'distribution' has two values intended for evaluating the distribution of a discrete variable, you obviously do not use these values since in our problem the input variables are continuous).
You also have to decide whether the conditional distributions estimation is done parametrically or not, and what estimation method to use. You can use MATLAB function `hist` in order to get some intuition regarding the shape of the distributions (recall that histogram is one of the non-parametric methods you have learned).
The `NaiveBayes.fit` function will estimate the a-priori and conditional distributions for you, depending on the options you give it.

- Explain the considerations behind your choice of whether to use a parametric method or not. If you choose the parametric method, explain the considerations behind choosing the appropriate distribution. Particularly explain the advantages and disadvantages of the distributions you have used and why they are appropriate for the type of the task you are dealing with. If you choose the non-parametric method, explain the considerations behind the specific method you have chosen (for example, if you chose to do interpolation with a window function, explain which window you chose and why).

- Explain what problem can emerge when using this kind of a classifier? How can this problem be solved?
- Performance evolution - Run the algorithm, and use the guidance in section (3) to evaluate the performance.

To be submitted:

1) Put your experimental results in a table or graph and explain the results obtained for the classification problem.
2) Suggest a way to improve the performance of the system.
3) Attach the call for the MATLAB function. Please note that the calling code should contain values for all the options that the function allows, even if it is the default value. Explain the values you selected.

Third Task – The Perceptron Algorithm

In this part you have to implement and train a linear classifier using the Perceptron algorithm, according to the algorithm you saw on the lecture and in the tutorial (tutorial 6). This algorithm is called Rosenblat’s Perceptron Algorithm. This is an algorithm which uses online update rule, that is in each step the algorithm receives a single example and its label. The samples are received with repetitions, meaning each sample can be given as an input more than once during training.

- Implement Rosenblat’s Perceptron algorithm. Make sure your code is clear and well documented.
- Note that the algorithm assumes that the labels are in $\{-1,1\}$, so you have to correct the labels you have which are in $\{0,1\}$.

- Make sure the algorithm receives all samples in the training set before he receives repeated samples. One way to do it is to make a pass on all samples in a random order, and then repeat a few passes.
- Note that you have to set a stopping condition for the algorithm. Make sure your code will not run infinitely. **Hint**: Plot the average error of the classifier at each step, and note how does the error behaves as a function of the iteration index.
- Evaluate the performance of the algorithm using the instructions in section 3.

For submission:

1. A well-documented code
2. Plot a graph of the convergence of the algorithm, that is the classification error as a function of iteration. Show on the same axes both the training error and the test error.
3. Explain the results you got. Explain your choice of stopping condition.

GOOD LUCK!