Instituto Superior Técnico

**Departamento de Engenharia Electrotécnica e de Computadores**

# Machine Learning

$3^{rd}$ Lab Assignment

Shift__________ Group number__________

Number__________ Name________________________________________

Number__________ Name________________________________________

**Naive Bayes classifiers. Estimation of statistical quantities.**

## 1 Naive Bayes classifiers

Naive Bayes classifiers normally are relatively simple, and are very effective in many practical applications. One of these applications is language recognition. In the first part of this assignment, you will make some of the code of a Naive Bayes language recognizer, and you will then test it. The training data for the recognizer are provided to you. Most of the code of the recognizer is also provided, but the parts that specifically concern the classifier's computations are missing. You will be asked to provide them. After that, you will be asked to test the recognizer.

### 1.1 Software and data

The Matlab code for the recognizer is given in the file `languagerecognizer.m`. This code is incomplete, and should be completed by you as indicated ahead. The code consists of two parts, which are clearly identified by comments:

- The *first part* reads the trigram counts of the training data for the various languages, from files that are supplied. The names of these files are of the form `xx_trigram_count_filtered.tsv`, where `xx` is a two-character code identifying the language that the file refers to (`pt` for Portuguese, `es` for Spanish, `fr` for French, and `en` for English).

  The aforementioned files contain the data of one trigram per line: each line contains the trigram, followed by the number of times that that trigram occurred in the corresponding language's training data. Before counting the trigrams in the training data, all upper case characters were converted to lower case. The set of characters that was considered was {abcdefghijklmnopqrstuvwxyzáéíóúàèìòìâêîôûäëïöüãõñ .,:;!¡?¿-'} (note that there is a blank character in the set). Trigrams containing characters outside that set were discarded. You may want to look into the trigram count files to have an idea of what are their contents, or to check the numbers of occurrences of some specific trigrams.

  After executing the *first part* of the code, the following variables are available:

  - `languages`: Cell array that stores the two-character codes for the languages. For example, `languages{4}` contains the string `'en'`. Note that the argument is between braces, not parentheses.
  - `total_counts`: Array that contains the total number of trigrams that occurred in the training data, for each language. For example, `total_counts(4)` contains the

total number of trigrams that occurred in the training data for English. Trigrams that occurred repeatedly are counted multiple times.

The *first part* of the code is complete: you shouldn't add any code to it.

- The *second part* of the code consists, basically, of a loop that repeatedly asks for a line of input text and then classifies it. Each iteration of the loop performs the following operations:

  - Ask for a line of input text and read it.
  - Check whether the input text contains only the word `quit`. If so, exit the loop (this will end the program).
  - Convert all the input text to lower-case.
  - Perform a loop on the languages. Within this loop, perform a loop on all the trigrams of the input text.
  - Print the scores of the various languages, the recognized language and the classification margin.

  This description of the operations performed by the *second part* of the code may sound somewhat incomplete, because this part of the code actually is incomplete. You should complete it by adding code, as described below.

The places where you may need to add code are clearly marked, with comments, in the file `languagerecognizer.m`. Those places are identified, in the comments, as Code Sections 1, 2 and 3. You will need to use those identifications later on.

The code that is provided already contains all the loops that are needed, as well as a few more commands. You will need to add the code that performs the calculations for the recognizer itself, using the data produced by the *first part* of the program (described above), as well as some data that are computed by already existing code of the *second part* of the program. Take into account the following indications:

- The basic structure of the *second part* is as follows:

  - There is an outermost loop, which repeatedly asks for input text and then proceeds to classify it.
  - That loop contains a loop on the languages.
  - The loop on the languages contains a loop on all the trigrams of the input text. In the beginning of this loop, the trigram that is to be processed in the current iteration is placed in the variable `trigram`, and the number of occurrences of that trigram in the training data for the current language is placed in the variable `trigramcount`.

- In Code Section 3, the final results of the calculations that you perform should be placed in an array called `scores`, of size 4, with an element for each language. For example, `scores(4)` should contain the score for English. The scores should be computed so that a higher score corresponds to a language that is more likely to be the one in which the input text was written.

- The end of the *second part* of the code already contains the instructions that will find the language with the highest score and output the results. The program outputs the scores of the various languages, followed by the identification of the language that has the highest score, and by the *classification margin*, which is the difference between the two highest scores.

### 1.1.1 Practical assignment

1. Complete the code given in the file `languagerecognizer.m`. Transcribe here the code that you have added to the program. Clearly separate and identify Sections 1, 2 and 3 of the added code.

2. Once you have completed the code and verified that the recognizer is operating properly, complete the table given below, by writing down the results that you obtained for the pieces of text that are given in the first column.

   The last piece of text is intended to check whether your recognizer is able to properly classify relatively long pieces of text. It is formed by the sentence "I go to the beach. " repeated ten times (in the table, the piece of text is abbreviated). Note that the given sentence has a blank space after the period, so that the repeated sentences are grammatically correct. You may use copy and paste operations to ease the input of this piece of text.

| Text | Real language | Recognized language | Score | Classification margin |
|---|---|---|---|---|
| O curso dura cinco anos. | pt | | | |
| El mercado está muy lejos. | es | | | |
| Eu vou à loja. | pt | | | |
| The word é is very short. | en | | | |
| I go to the beach. ... I go to the beach. | en | | | |

3. Comment on the results that you have obtained.

## 2 Estimation of statistical quantities

Systems trained in a supervised manner with certain cost functions allow the estimation of several useful statistical quantities. In this part of the assignment we shall address the estimation of some of those quantities. The cost functions available are the mean squared error and dual slope.

The files *STAT1.TXT*, *STAT2.TXT* and *STAT3.TXT* contain training sets corresponding to one-variable functions perturbed by additive noise. The values of the independent variable are given by

$$x^k = -1 + (2k-1)/400, \quad k = 1, \cdots, 400,$$

i.e., there are 400 values, uniformly spaced between $-1$ and 1. The functions in the three files are all of the form

$$s^k = A\cos(\omega x^k + \varphi).$$

The desired values that appear in the training sets are given by

$$d^k = s^k + \epsilon^k,$$

where $\epsilon^k$ is zero-mean Gaussian noise. The noise variances in the files *STAT1.TXT* and *STAT2.TXT* are independent of $x^k$. In *STAT3.TXT*, the noise variance depends on $x^k$.

### 2.1 Practical assignment

1. Graphically visualize each of the training sets, to better understand what data each one of them contains. To visualize the data from a file, set it as the training set in *neural_gui*. After that, click the *Plot* button, and then left-click on the rectangle marked *error*. You'll obtain two plots, named Figure 1 and Figure 2. In Figure 2, the red crosses depict the training set. The blue line corresponds to the output of the MLP. You can ignore this line for now, because the MLP hasn't been trained yet.

---

2. By appropriately training MLPs, estimate the parameters $A$, $\omega$, and $\varphi$ for each of the three files. For the first two files, also estimate the variance of the noise (try to do this in the simplest way possible). Indicate the estimates that you have obtained. Describe how you have performed the estimation of the various parameters. Indicate the structures of the networks that you have used. Explain your choices, including those of the numbers of hidden units, activation functions and cost function.

3. Estimate the conditional 10% quantile of the training data from the file *STAT3.TXT*. For that purpose, the appropriate cost function is the one named *dual slope*. When using this function, after clicking on *OK* in the *Training configuration* window, you'll be asked for the values of the function's two slopes. For a proper training with this kind of cost function, you'll need to set the *cost tolerance* parameter, in the *Training configuration* window, to a value slightly above 1 (e.g. 1.0001).

   Indicate how many training points should, theoretically, lie below the conditional 10% quantile line. Visualize the training data and the MLP's output, as indicated above, and count the number of points that actually lie below that line (you may have to zoom in a lot on some of the points to see whether they are above or below the 10% quantile line).

| | |
|---|---|
| Theoretical number of points: | Actual number of points: |

4. Indicate how many training points should, theoretically, lie within the conditional 80% prediction interval. Find the number of points that actually lie within that interval (it will be easier to find the number of points that lie outside the interval, and then subtract it from 400).

| | |
|---|---|
| Theoretical number of points: | Actual number of points: |

5. Estimate a conditional 80% prediction interval of the training data, and visualize it. You can show two superimposed plots in the same figure by using the Matlab command *hold on*. Sketch the plot of the estimated conditional interval that you obtained, together with the training data. Comment on the results.