Machine Learning
4th Lab Assignment
Instituto Superior Técnico

Bayes classifiers

1 Bayes Classifiers

Bayes classifiers normally are rather simple, and are very effective in many practical situations.

1. (T) Describe in your own words how the Bayes and Naive Bayes classifiers work. Be precise. Use equations when appropriate.

`Note` In this assignment, you'll often find between brackets, as in {`command`}, suggestions of Python commands that may be useful to perform the requested tasks. You should search Python documentation, when necessary, to obtain a description of how to use these commands.

2 A simple example

In this part of the assignment, you'll make a naive Bayes classifier and a Bayes classsifier for a very simple set of data. The input data are two-dimensional, and belong to one of three classes.

1. Load the numpy files for `data1` which have already been split into training data (variables `xtrain` and `ytrain`) and test data (variables `xtest` and `ytest`).{`load from numpy`}
2. Obtain a scatter plot of the training and test data, using different colors, or symbols, for the different classes. Don't forget to use equal scales for both axes, so that the scatter plot is not distorted. {`scatter from matplotlib`}
3. Make a script that creates a naive Bayes classifier based on the training data, and that finds the classifications that that classifier gives to the test data. The script should print the percentage of errors that the classifier makes on the test data. Do not assume prior probabilities are equal. Write your own code, do not use any Python ready made function for Naive Bayes classification.{`mean and var from numpy, norm from scipy`}

Suggestion: You will need to estimate probability densities of certain sets of data, to solve this item. For the estimation of each density, use a Gaussian distribution. Estimate its mean and variance, respectively, as the mean and variance of the corresponding set of data (for the variance estimates, divide by N, and not by $N-1$, where N is the number of data points). The estimator that you'll obtain in this way is the maximum-likelihood estimator of the Gaussian distribution.

4. Obtain a scatter plot of the test data classification. {`scatter from matplotlib`}
5. Indicate the percentage of errors that you obtained in the test set. {`accuracy_score from sklearn.metrics`}
6. Repeat the previous 3 items but using the Bayes classifier instead of the Naive Bayes classifier. {`multivariate_normal from scipy`}
7. Comment the results from both classifiers on these data.

The classification problem that you have just solved is very small, and was specially prepared to illustrate the basic working of naive Bayes classifiers. You should be aware, however, that the real-life situations in which these classifiers are normally most useful are rather different from this one: they are situations in which the data to be classified have a large number of features and each feature gives some information on which is the correct class. Normally, for each individual feature, there is a significant probability of giving a wrong indication. However, with a large number of features, the probability of many of them being simultaneously wrong is very low, and, because of that, the naive Bayes classifier gives a reliable classification. The second part of this assignment addresses such a situation.

3 Language recognizer

One of the applications in which naive Bayes classifiers give good results and are relatively simple to implement, is language recognition. In the second part of this assignment, you will implement a naive Bayes language recognizer, which you will then test.

3.1 Data

The training data are provided as different files containing the trigram counts for the various languages. The names of these files are of the form `xx_trigram_count.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.

3.2 Practical assignment

3.2.1 Training

1. Load the data containing the trigram counts for each language. Note that for tsv files the separator is tab. {`read_csv from pandas`}
2. Check data format and contents. {`shape, head`}
3. Build a training matrix `X_train` where each language corresponds to one sample and with as many features as the number of trigrams. Create the corresponding vector `y_train` containing a different label for each language.
4. Create a Multinomial Naive Bayes model. Use Laplace smoothing and assume equal priors for all languages. {`MultinomialNB from sklearn`}
5. Fit the model using the `X_train` and `y_train` matrices you created. {`fit`}

3.2.2 Testing

Verify that the recognizer is operating properly and then 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.

1. To verify that the recognizer is operating properly make predictions for your training data and calculate the accuracy of class predictions. {`predict, accuracy_score from sklearn`}
2. Build a sentences matrix containing the test sentences from the table.
3. Create a vectorizer in order to obtain the trigram counts for the sentences in your sentences matrix. Use as vocabulary the set of trigrams from the training data. {`CountVectorizer from sklearn`}
4. Learn the data trigram counts for the given sentences and store them in `X_test` matrix. Build the corresponding `y_test` vector.{`fit_transform`}

5. Make predictions for your test data using the Naive Bayes model. {`predict`}
6. Compute the classification margin, which is the difference between the two highest probabilities. {`predict_proba, sort`}

Text	Real language	Recognized language	Score	Classification margin
Que fácil es comer peras.	es			
Que fácil é comer peras.	pt			
Today is a great day for sightseeing.	en			
Je vais au cinéma demain soir.	fr			
Ana es inteligente y simpática.	es			
Tu vais à escola hoje.	pt			

7. Give a detailed comment on the results that you have obtained for each sentence.