Assignment 1

Dates and rules.

This assignment is for groups of 2 students. Student groups must be registered by October 31.

The assignment can be submitted between November 1 and November 14.

The deadline for submitting the assignment is November 14, 23:59.

There is an additional tolerance of 48 hours for solving any problems with your submission. This period should be used exclusively for this purpose. No submissions will be accepted after 23:59 of November 16.

You must submit a zip file containing, at least, these four files, named exactly as specified (names are case-sensitive):

TP1.txt
: This is the questions and answers file.

TP1.py
: This is a Python 3.x script that can be used to run your code for this assignment.

NB.png
: The plot of training and cross-validation errors for the KDE kernel width of your implementation of the Naïve Bayes classifier.

SVM.png
: The plot of training and cross-validation errors for the gamma parameter of the SVM classifier.

Optionally, you can include other python modules if you wish to separate your code into several files.

Objective

The goal of this assignment is to parametrize, fit and compare Naive Bayes and Support Vector Machine classifiers. The data set is inspired on the banknote authentication problem in the UCI machine learning repository, but the data was adapted for this assignment. You can download the data files from here:

TP1_train.tsv

This is the training data set. Use this to optimize parameters and train the classifiers.

TP1_test.tsv

This is the test set for estimating the true error and comparing the final classifiers.

You must implement your own Naïve Bayes classifier using Kernel Density Estimation for the probability distributions of the feature values. For this, you can use any code from the lectures, lecture notes and tutorials that you find useful. Also, use the `KernelDensity` class from `sklearn.neighbors.kde` for the density estimation.

You will need to find the optimum value for the `bandwitdh` parameter of the kernel density estimators you will use. Use the training set provided in the TP1_train.tsv for this.

The second classifier will be the Gaussian Naïve Bayes classifier in the `sklearn.naive_bayes.GaussianNB` class. You do not need to adjust aditional parameters for this classifier

Finally, use a Support Vector Machine with a Gaussian radial basis function, available in the `sklearn.svm.SVC` class. Use a regularization factor C = 1 and optimize the `gamma` parameter with cross-validation on the training set.

After training your classifiers and fine tuning all parameters, compare the performance of the three classifiers, identify the best one and discuss if it is significantly better than the others.

The data are available on .tsv files where each line corresponds to a bank note and the five values, separated by commas, are, in order, the four features (variance, skewness and curtosis of Wavelet Transformed image and the entropy of the bank note image) and the class label, an integer with values 0 for real bank notes and 1 for fake bank notes.

In addition to the code, you must include two plots with the training and cross-validation errors for the optimization of the bandwidth parameter of your Naïve Bayes classifier (this plot should be named `NB.png`) and the gamma value of the SVM classifier (this plot should be named `SVM.png`). These plots should have a legend identifying which line corresponds to which error.

Furthermore, you must also answer a set of questions about this assignment. The question files are available here (choose one only). Right-click the links and save the chosen file locally. You must include the file with your answers in the zip file when submitting the assignment.

- TP1.txt, English version
- TP1.txt, Portuguese version

Optional: fine tune the SVM classifier

The SVM classifier you will use has two adjustable parameters, `gamma` and `C`. In this assignment we fixed parameter `C` at a value of 1 but you can try to improve the performance of this classifier by adjusting simultaneously these two parameters. This is an optional exercise, worth only 1/20 of the assignment grade: optimize both parameters, compare the optimized SVM with the previous classifiers and discuss whether this additional optimization was useful in this case.

Guidelines for the implementation

These are suggestions to help you understand what is required in this assignment. This section may be updated to clarify questions that arise during the assignment.

- Process the data correctly, including randomizing the order of the data points and standardizing the values.
- Determine the parameters with cross validation on the training set, leaving the test set for the final comparisons.
- Use the same bandwidth value for all the Kernel Density Estimators in each instance of your Naïve Bayes classifier, and try values from 0.02 to 0.6 with a step of 0.02 for the bandwidth.
- Use the default kernel ('gaussian') for the Kernel Density Estimators.
- To optimize the gamma parameter of the SVM try values from 0.2 to 6 with a step of 0.2
- When splitting your data for cross validation use stratified sampling.
- Use 5 folds for cross validation
- Use the fraction of incorrect classifications as the measure of the error. This is equal to 1-accuracy, and the accuracy can be obtained with the `score` method of the classifiers provided by the Scikit-learn library.
- For your Naïve Bayes classifier, you can implement your own measure of the accuracy or use the `accuracy_score` function in the `metrics` module if you find it useful.

- For comparing the classifiers, use the approximate normal test and McNemar's test, both with a 95% confidence interval

Note on plagiarism

Plagiarism is any attempt to make you seem the author of text, code or any work which is not really yours. Plagiarism breaks the necessary trust for fair evaluation and any student submitting plagiarized work will fail the course immediately. If you rely on material other than what was provided in this course you must identify it and credit your sources.

Also note that you will be graded on the work you actually did, so even if you avoid plagiarism by crediting your sources you still need to do your own implementation.