MAP654I Practical introduction to Machine Learning
Practical Session 3 Classification

Rémi Flamary

The objective of this practical session is to manipulate and understand the machine learning problems and methods discussed in the course. The practical session will be done in Python 3 and it is strongly recommended to have a working Anaconda environment. The different sections of the session should be implemented in Jupyter notebooks and it is strongly recommended to take notes in the notebook during the session.

The individual reports (with figures and discussions) for the session will be uploaded on moodle in python notebook format. It is expected to have a working code (reproducible figures) and a short discussion in markdown format for all the results obtained in the practical session. Note that ALL plotting and visualization questions in the practical session require a **critical discussion** of the visualized result that will be graded.

The end of the report must contain a personal discussion about the session (what was hard to understand and implement, how you would do it next time, what was new, discussion of relation with the course, personal discussion about how to use these tools in a professional setting, ...).

Importing libraries

In this practical session we will use Numpy/Scipy Python libraries for handling numerical data and Matplotlib for plotting them.

```
import numpy as np
import pylab as pl
import scipy as sp
```

Note that a more standard import for matplotlib is `import matplotlib.pyplot as plt`. You can use the name you want for the plotting module but if you choose `plt` you need to replace it in the function names given in the following (`plt.plot` instead of `pl.plot`). We will also need to have access to some functions and classes in the Scikit-learn toolbox.

1 Datasets description

1.1 Pima dataset

Context This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Content The datasets consists of several medical predictor variables and one target variable, Outcome (whether the person is actually diabetic). Predictor variables included are

- **Pregnancies** Number of times pregnant
- **Glucose** Plasma glucose concentration a 2 hours in an oral glucose tolerance test

- **BloodPressure** Diastolic blood pressure (mm Hg)
- **SkinThickness** Triceps skin fold thickness (mm)
- **Insulin** 2-Hour serum insulin (mu U/ml)
- **BMI** Body mass index (weight in kg/(height in m)^2)
- **DiabetesPedigreeFunction** Diabetes pedigree function
- **Age** Age (years)

Acknowledgements Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.

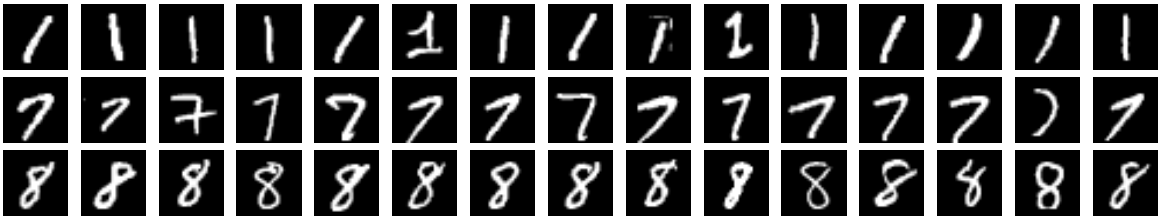
1.2 Digits dataset

This dataset is a small subset of the well known MNIST dataset (3 classes only 1000 samples per class on rain data) and contains images of written numbers. One interesting complementary information is of course the class of those images (the number that was written).

The file `"digits.npz"` contains the following matrices:

- `"x"` and `"xt"`: data matrices containing respectively $n = 3000$ and $n_t = 1500$ training example of written digits. Each line in those matrices is a 28×28 image stored as a transposed vector (a line of size 784).

  Some examples of the images in the training sets:



- `"y"` and `"yt"`: the labels of the images described above. they are vectors containing the classes $(1, 7, 8)$ of each images in **x** and **xt**. The samples are sorted by class.

Acknowledgements Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

1.3 Loading the data

For both datasets do the following:

1. Load the data with `np.load("filename.npz")` and store the different matrices in memory (for instance in variables `x1,y1` for Pima data and `x2,y2,xt2,yt2` for digits data). For the digits dataset, it is better to perform one simple pre-processing that scales the values between $[0, 1]$ by dividing the data matrix by 255.
2. Do a quick look at the data, compute the mean values for each variable and interpret it.

2 Predicting Diabetes on the Pima dataset

On this dataset we will evaluate the performance of the classifier with the accuracy and with the AUC.

2.1 Know the data

- Visualize the data as scatterplots between pairs of variables (where the color is the class).
- What are the variables that seem to help predict the class? Do those variable make sense from a medical perspective ?
- Split the data in training/test by keeping $n = 300$ samples for training/validation and the remaining for test.
- Do the feature have similar variances/scaling? Is that a problem?
- Apply a standardization to the training and test data (`StandardScaler`).

2.2 Bayesian decision and linear classification

- Train a Linear Discriminant Analysis (LDA) classifier with the default parameters, compute its accuracy and AUC on the test data (`LinearDiscriminantAnalysis,roc_auc_score`). Note that in order to compute the AUC you will need to compute the score with `est.predict_proba` and keep the second column (probability of the class 1).
- Perform a cross validation `GridSearchCV`) for the parameters of the method (`shrinkage`). Does the validation leads to better performance? What is the optimal value for the parameter?
- Train a Quadratic Discriminant Analysis (QDA) classifier with the default parameters, compute its accuracy and AUC on the test data (`QuadraticDiscriminantAnalysis`). Is the performance better than LDA?
- Perform a cross validation for the parameters of the method (`reg_param`). Does the validation leads to better performance? What is the optimal value for the parameter.
- Train a Gaussian Naive Bayes (NB) classifier (`GaussianNB`). What is its performance with respect to QDA and LDA?
- Train a Logistic regression classifier (`LogisticRegression`) with the default parameters. Compute its performance and compare it to the previous classifiers.
- Perform a a cross validation for the parameters of the model (`C`) by setting the penalization to L1. Is the model sparse? What variables were removed from the model? Is the classifier performing well?
- What is the best decision method so far? Is the best model linear (LAD,LR) on quadratic (QDA,NB)?
- Interpret the separability of the sample in the predicted score space by plotting histograms for the samples for each class in 1D.
- Interpret the weight for a good linear model. What is the effect of each variable on the risk of diabetes? Does it make medical sense?

2.3 Nonlinear methods

For the following non-linear classifiers, `RandomForestClassifier`, `SVC`, `MLPClassifier`, `GradientBoostingClassifier` do the following:

- Fit the model with the default parameters and compute its prediction performance. Is it better than a linear estimator?
- Do a quick validation of some of the important parameters (manually or with sklearn classes). Can you find a better performance?

2.4 Comparison and interpretation

- Collect the test performances for all methods investigated above in a table (in a dataframe and printing it for instance). Which methods work the best in practice?
- Which model is best from a medical/practical perspective? Do we need non-linearity in this application?
- For the best model, compute the confusion matrix for the test data. What is the false negative rate (FNR) (FN/(FN+TP)) for this classifier? Is it good for this kind of applications?
- Since a false negative can have an important medical impact, propose a new threshold for the predicted score that leads to a FNR of less that 10% (this can be done by changing manually the value of the `intercept_` in the trained classifier).

3 Predicting Classes on the Digits dataset

You already used this dataset in a previous session so it is assumed that you already know its properties. The data is already split in training/test sets.

3.1 Evaluate the different supervised methods

For at least the following classifiers `LinearDiscriminantAnalysis,LogisticRegression,SVC,MLPClassifier` do the following.

- Fit the model with the default parameters and compute its prediction performance (accuracy on test data).
- Do a quick validation of some of the important parameters (manually or with sklearn classes) to get a better performance if possible.
- Store the model and the accuracy for the best parameter configuration.

3.2 Interpreting the classifier

- Compare the performances of the different methods (with different metrics). Which model is the best on test data?
- Select the best classifier from the previous section and use it to predict labels on the test data.
- Compute the confusion matrix and interpret the errors made by the classifier. What is the class that is the most difficult to recognize? Are there somme classes that are harder to discriminate?
- Plot some of the sample that are miss-classified. Are they difficult to recognize ? Why did the classifier fail to recognize them?
- Pick a well classified sample and create 1000 noisy samples of it by adding gaussian noise (`np.random.randn`). Pick a level of noise that allows you to still distinguish clearly the class. Compute the accuracy of the classifier on those 1000 noisy samples. If the accuracy is 1, increase the noise level or choose another well classified sample.
- How robust is the classifier? Visualise some of those "adversarial" examples when the accuracy on the noisy samples is not 1.

Bonus: Convolutional Neural network (CNN)

Implement a CNN and train it on the data (you will need to reshape it to store it as images). Investigate te performance of the CNN when varying its parameters. Does it have better performance than the model above? Is it more robust to adversarial examples?.