ML Project

Milestone 1

Prune Ollier, Ainhoa Vilanova, Elias William

Machine Learning
EPFL
26/04/2023

1 Introduction

This paper is the report for Milestone 1 of the EPFL CS-233 course project. The mentioned implementation as well as this report is the work of Ainhoa Vilanova, Elias William and Prune Ollier. Our goal is to implement in python the following methods: K-means, Logistic Regression and SVM, seen in the course and to test them on a dataset. We will be evaluating them on evaluation metrics specific for the tasks and present our results at the end this report.

2 Method

Throughout the project, we took our main inspiration for implementation from the exercise series of the CS-233 course. Indeed, having done the exercise series, we had already understood the principles asked in the MS1 which are k-means, linear regression and SVM. The series were a simple way to test our implementation and understanding of these principles, and the true challenge of the MS1 was to adapt our implementation to a new purpose and framework. Therefore, we used the dummy_methods as a template, in order for our methods to be sectioned into the initialization (__init__()), the data training (fit()) and the predicting (predict()) methods as required.

2.1 Data preprocessing

The preprocessing of the data consisted randomizing the order of the training data, splitting the training data set into both training and validation sets. If we use the flag –test, no validation data is created. The first 80% of the original training data set is attributed to actually training the models and the remaining 20% to validation. Then the training, test and validation datasets are normalized with respect to the mean and standard deviation of the training dataset. A bias term is also added to the data. This is done to improve accuracy of the model by making it more flexible.

2.2 K-means

As stated above a lot of inspiration was taken by the completed exercise sessions. Functions declared in the k-means class that only should be used internally by the class begin with __, variables only used by the class begin with _. The initial cluster centers are chosen as random samples in the training dataset. To validate the model cross validation was used. By using the new argument "–visualize" when running the script this cross validation is performed. Both the accuracy of the model and the f1 score is shown dependent on the number of clusters for both training and validation data.

2.3 Logistic Regression

The given data being given as regular labels, it was needed to convert the labels into the one hot format, as assumed in our implementation. This conversion is done in the fit() method before any manipulation of the data in the logistic regression method. The weights were passed as attributes to the logistic_regression class in order to be easy of access in many of the methods in logistic_regression.py. To validate the best the model the learning rate was changed. The amount of iterations that it took for the model to converge was calculated aswell as the validation accuracy and f1-score.

2.4 SVM

The SVM script is really short in code because we're the sklearn package, especially the SVC() class that implements Support Vector Machines for classification tasks. It is possible to specify various parameters such as the kernel function (linear, polynomial, rbf), the penalty parameter (C), the gamma parameter, the degree and the coef0. The kernel function is used to transform the data into a higher-dimensional space so that it becomes separable by a hyperplane. The fit function first trains the model by SVM, returning predicted labels for training data and then the predict function is used to run prediction on test data.

3 Experiments/Results

3.1 K Means

By using cross validation the best value for the number of clusters can be found. According to figure 1, 40 clusters seem to be the best choice due to the flattening of the curve.

3.2 Logistic Regression

In Figure 2 the validation with different learning rates for logistic regression is shown.

3.3 SVM

The iterations to find the best fitting model were done manually. Thanks to the sklearn package and

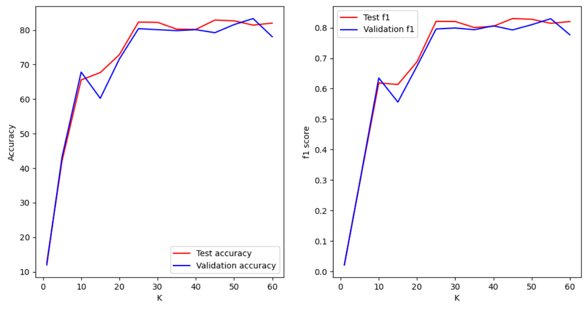

Figure 1: Cross validation for K-means.

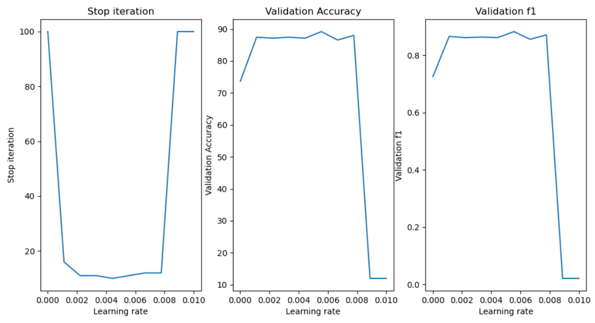

Figure 2: Validation for logistic regression.

GridSearchCV, a cross validation has been done between the 4 parameters.

3.4 Main script

All the preprocessing of the data is implemented in the main script of the project. Also a new argument was added called –visualize. Depending on the current method that is being used by the main script, this argument performs and visualizes the method performance depending on some hyperparameters.

4 Discussion/Conclusion

In this project three different machine learning algorithm was implemented and tested. By changing the hyperparameters of each model we can conclude that the performance of the model is heavily dependent on these choices. For example in Figure 2 we can see that a learning rate that is too high makes the algorithm unstable. It is also important to validate the performance in regard to overfitting. An example can be seen in Figure 1. The accuracy is still good for a K value of 60, but not much better than 20. By using these methods we could find the best hyperparameters for the different models as seen in Table 1. For the SVM the RBF kernel was the best performing. Theoretically with the correct parameters the polonomial kernel should perform equally good.

One thing that was noticed was that the some best performing hyperparameters, for example the learning rate for the logistic regression, was not the exact same for the validation set and the test set. For example in 2 the value 0.006 for learning rate seems like a good choice. This learning rate on the test set does not make the model converge. This must be because we are not using the same set to train the model for the validation as opposed for the test set. When using validation the training set is split, when using test there is no split in the training set performed. This choice was made to increase the sample size to train the model when testing. A better choice might have been to just use the test set as the validation also to get better agreement for the validation and test.

Model	Parameter(s)	Test Accuracy	Test f1 score	Train Accuracy	Train f1 score
K means	K = 40	81.34%	0.8089	82.67%	0.8221
Logistic reg.	lr = 0.001	92.090%	0.9207	100%	1.0
Linear SVM	C = 0.001 ; gamma = 0.00001 ; degree = 1 ; coef0 = 0	95.292%	0.951174	97.942%	0.979520
Polynomial SVM	C = 0.001 ; gamma = 0.01 ; degree = 2 ; coef0 = 10	93.785%	0.935914	96.305%	0.963091
Radial Basis Function SVM	C = 3 ; gamma = 0.001 ; degree = 1 ; coef0 = 0	96.422%	0.962389	100%	1.0

Table 1: Table of best performing models