Aristidis Chronarakis 389

13/05/18

T6- Machine Learning

Project #1

Exercise 1

The purpose of this exercise is to develop an Optical Character Recognition system using four different classifications methods. The whole implementation was made in Python using the Machine Learning library (scikit-learn). More specifically for each classification method we use different values on the parameters in order to find the most suitable, which are these with the best prediction accuracy. The accuracy that is measured on each tuning parameters' attempt is the mean prediction accuracy from the iterations of a 10-fold cross validation procedure on the train dataset (separating the train dataset on ten disjoint sets and using nine of them for the training procedure and one for the computation of method's accuracy), and its optimal value is 1. After the selection of the parameters and since our model for this method is trained with the best parameters we make our predictions for the test dataset. Finally we compare the the predictions with the ground truth in order to find the accuracy of our predictions on the test dataset.

Results Overview

*Multi-class SVM classifier with Linear & RBF kernel (using libsvm)

The parameters that make sense to be tuned on the multi-class SVM classifier with the Linear kernel option, are the penalty parameter of the error term (C), the tolerance for stopping criterion (tol), and the limit of

iterations within solver (max_iter) . The results from the 10-fold cross validation with different combinations show that for C=1.0, tol=0.001 and with not a limit on iterations within a solver (which are the classifier's default options on scikit-learn) we get good results. More specifically classifier's mean accuracy score about the predictions on the 10-fold cross validation procedure is 0.98 where the prediction accuracy score on the test dataset is 0.96 out of 1.
A parameter that makes sense to be tuned on the multi-class SVM classifier with the RBF kernel option except the previous referred on the Linear kernel is the kernel coefficient (gamma). With the 10-fold cross validation we try to find the best gamma value, having selected C=1.0, tol=0.001 and not a limit on iterations within a solver as optimal values for the previous referred parameters. With gamma=1/(number of features) we get mean accuracy score results close to 0.66-0.67 out of 1. The optimal value for the gamma parameter is equal to 1 / (n_features * variance_of_train's_elements) which gives 0.99 out of 1 mean accuracy prediction score on the 10-fold cross validation. With this selection as parameters' values we get 0.98 prediction accuracy score on our test dataset.

***MLPs with 1 hidden layer**

The main parameters that make sense to be tuned on the MLP classifier with 1 hidden layer are the number of neurons on the hidden layer, its activation function (activation), the solver for the optimization (solver), the penalty parameter about L2 (alpha), the number of maximum iterations(max_iter), the tolerance for the optimization (tol) and also the support or no for an early stopping (early_stopping). Having the default values for all parameters (alpha=0.0001, max_iter=200, tol=0.0001,early_stopping=False) except the activation function on the hidden layer, the number of neurons on this layer and the solver, we try to find the best combination which gives the best mean prediction

accuracy score using the 10-fold cross validation. We get similar results for different solver selections and thus between them we select as optimal the solver which is from the family of quasi-Newton methods which may converges faster and gives better results. After solver's selection there are many combinations that can give results with accuracy over 0.9 out of 1. If we use identity as activation function then 10 neurons on the hidden layer are enough to give as good results. If we use tanh or logistic or rectified linear unit functions we need more neurons on the hidden layer. As optimal selection is choosen this with the use of identiy as activation function and the 10 neurons on the hidden layer because is the simpler which also gives good results. About the results, the mean prediction accuracy result from the cross validation is 0.96 out of 1, where the prediction accuracy on our test dataset is 0.95 out of 1.

***Summary**

The results show that all the previous tested methods after the essential tuning procedure can be used as classifiers for our Optical Character Recognition system.

Exercise 2

The purpose of this exercise is to develop a regression system and compare four different methods for their accuray. The accuray of the regression is measured using as metrics the mean squared error regression loss and the score related with the coefficient of determination (variance score), on a 10-fold cross validation procedure on the dataset (separating the dataset on ten disjoint sets and using nine of them for the training procedure and one for the computation of method's accuracy). The whole implementation was made in Python with using the Machine Learning library (scikit-learn). For our metrics we want mean squared error values as close as possible to 0 and coefficient

of determination values as close as possible to 1, which would mean that we have a very good regression system for these data.

Results Overview

*Linear Regression

The Linear Regression method gives as mean values from the 10 iterations of the 10-fold cross validation procedure these :
mean squared error (mean) = 4.93
variance score (mean) = 0.53

*Polynomial Regression

The Polynomial Regression method gives its best results for degree equal to 2. As we increase the degree, except that we need more compution time, we also get worse results. This is due to the fact that by increasing the degree we have to handle more complicated functions. The mean values from the 10 iterations of the 10-fold cross validation procedure are :
mean squared error (mean) = 6.75
variance score (mean) = 0.34

*Lasso's method

On Lasso's method as we decrease the constat that multiplies the L1 term from its default value (alpha=1.0), we get better results. The best results are given with alpha<0.01. The mean values from the 10 iterations of the 10-fold cross validation procedure when also alpha=0.001 are :
mean squared error (mean) = 4.92
variance score (mean) = 0.53

*MLP with 1 hidden layer

On MLP regression method with 1 hidden layer we can make various

combinations on parameters, in order to get good results. First of all, as optimization's solver is selected an optimizer in the family of quasi-Newton methods since is ideal for small datasets (converges faster and gives better results). The best results are given with the use of tanh or logistic as activation function on the hidden layer, on a hidden layer in which 5 neurons are enough to give good results. The mean values from the 10 iterations of the 10-fold cross validation procedure when we use 5 neurons and tanh are:
mean squared error (mean) = 4.30
variance score (mean) = 0.59

***Gaussian Processes with Linear & RBF kernel**

The Gaussian process with the Linear kernel option gives results which are close to the results of the previous methods. The mean values from the 10 iterations of the 10-fold cross validation procedure are :
mean squared error (mean) = 4.94
variance score (mean) = 0.52
The Gaussian process with the Gaussian (RBF) kernel gives bad results for all the different sigma values that are used. The best results between those bad results are given with sigma=0.01. Except the big mean squared error tha we get ,we also get negative value for the coefficient of determination which indicates that our model is not too good for our data and that possibly we face an overfitting condition.

***Summary**

The Linear regression method, the Lasso's and the MLP regressor seem that are those which give the best regression results for our data. The MLP regressor with tanh as activation function and with 5 neurons on the hidden layer gives slightly better results over all other methods.

*****The full code for the both implementations can be found on Github*****

https://github.com/chron56/ML-microprojects