CS771 Introduction to Machine Learning Assignment 2

Aditya Kankriya
22111072
adityask22@iitk.ac.in

Haris Khan
22111026
hariskhan22@iitk.ac.in

Srujana Sabbani
22111083
srujanas22@iitk.ac.in

Anshul Sharma
22111009
anshulsh22@iitk.ac.in

Madhav Maheshwari
22111037
madhavm22@iitk.ac.in

Faizal Khan
22111022
faizalk22@iitk.ac.in

1 Questions and Solutions

1.1 Suggest a method that you would use to solve the problem. Describe your method in great detail including any processing you do on the features, what classifier(s) you use to obtain the final predictions, what hyperparameters you had to tune to get the best performance, how you tuned those hyperparameters (among what set did you search for the best hyperparameter and how) e.g. you may say that we tuned the depth of a decision tree and I tried 5 depths 2, 3, 4, 5 and found 3 to be the best using held out validation.

Solution :

Here, We have used Logistic Regression classifier as our method to train our model.

Loading:
The file contains text based sparse vectors with 225 features and these vectors are mappped to one of the 50 error code classes The first element of each line can be used to store a target variable to predict. We have loaded the file using load_svmlight_file() since the data provided to us is sparse matrix. We have used the following paramters for converting the text data into CSR matrix.
n_features = 225
mutlilabel = false
offset = 1

Preprocessing:
In preprocessing we visualized data and the error codes 33,36,38 are rarely classified to datapoint hence we considered them as 0 in predicted values of error code.

Spliting the data:
The data is split into train and validation sets with 70% and 30% respectively using train_test_split()

from sklearn after splitting we convert the csr matrix to dense matrix using the function todense() from numpy.

Training:
In this problem for mutliclass classification we have used Logistic Regression. This implemented using function from sklearn the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr'.
solver='liblinear'
class_weight = 'balanced'
tol=0.0001
C=5
random_state=0

Tuning of Hyperparameters:
1. tol: Tolerance for stopping criteria. Default value in most program is 0.0004 then we decreased the value to 0.0001 and found the result good, but to confirm we increased the value to 0.001 and the result was same, so we again increased the value of 0.01 and the result was decreased compared to previous one, so we moved towards 0.001 and fixed 0.0001 as it was nearer to default value

2. C: Inverse of regularization strength. Started with default value = 1.0 , and we increased the value to 5 and accuracy got increased, and the value was fixed

3. multi_class: From theory ovr with linlibear would perform good here and by default, auto selects ovr , so we have explicitly mentioned ovr

4. class_weight: Weights associated with classes in the form class_label: weight. Due to imbalance input data, we have chosen 'balanced'. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data.

5. solver: Algorithm to use in the optimization problem. By default with ovr, liblinear is used

test:
after the model is trained we used validation data for testing the model. we predict the error code for all the validation data points and compare the predicted error code with the actual validation data error code to get accuracy.
for all the data points to get the probablites of each error code being classified we used predict_proba() function.

1.2 Discuss at least two advantages of your method over some other approaches you may have considered. Discuss at least two disadvantages of your method compared to some other method (which either you tried or wanted to try but could not). Advantages and disadvantages may be presented in terms of prediction, macro precision, training time, prediction time, model size, ease of coding and deployment etc.

Solution :

The other methods we tried is using decision trees.
Advantages:
1. Prediction accuracy with Decision Tree is 74 with validation data which is less compared to Logistic Regression.
2. Macro precision at k = 1 for Logistic Regression is 0.79 which is better compared to Decision Tree's macro precision is 0.72 .
3. For the same data the prediction time is less for Logistic Regression (8.4 seconds) and Decision Tree is 10.36 seconds.

Disadvantages:
1. The training time for Decision Tree(0.14 seconds) is way faster than Logistic Regression(3.14 seconds).
2. Precision at k = 1 for Decision Tree 0.86 is better compared with Logistic Regression whose precision at k=1 is 0.77(but eventually Logistic Regression gives more precision at k = 5).