CS 771-A Assignment 2 Submission

Team Yo
Sarthak Kohli (200886)
Dishay Mehta (200341)
Shubhan Ravi (200971)
Mohit Gupta (200597)
Shrey Mehta (200580)

Abstract

This is the report of the second assignment of Introduction to Machine Learning (CS771A) 2022-23-I offering.

Question 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

The method we are using is as following: **Multi-layer Perceptron classifier** of sklearn library MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting. This implementation works with data represented as dense numpy arrays or sparse scipy arrays of floating point values.

The hyperparameters which we optimised are:
hidden_layer_sizes: The ith element represents the number of neurons in the ith hidden layer.
activation: Activation function for the hidden layer.
solver: The solver for weight optimization.
learning_rate: Learning rate schedule for weight updates.
alpha: Strength of the L2 regularization term. The L2 regularization term is divided by the sample size when added to the loss.
The hyperparameters we found to be best after extensive trial and error are:
hidden_layer_sizes=200, **activation**='relu', **solver**='adam', **learning_rate**='adaptive', **alpha**=0.03
The rest of the parameters were set to their default value. We did not do any pre-processing to the train features before training the model. We used a validation set of 10% of total the given train set to validate our model and the rest of the 90% to train the model.
We tuned hyperparameters in the following way:

1. Initially, all the parameters were the default values, which were **hidden_layer_sizes**=100, **activation**='logistic', **solver**='adam', **learning_rate**='constant', **alpha**=0.001
2. We now tuned hidden_layer_sizes from 100 to 250 in intervals of 25 and found 200 to perform the best. Other parameters were constant during this step.

Preprint. Under review.

3. We now tuned the activation function, tried 'tanh', 'logistic' and 'relu', but we found 'relu' to perform the best, while other parameters were constant.
4. We then tuned alpha from 0.01 to 0.1 in intervals of 0.01 and found alpha=0.03 to perform the best. Alpha here represents the strength of the L2 regularization term. Other parameters were constant during this step.
5. Finally, we set learning_rate to be adaptive, and this improved results even further.
6. We could not improve results further by changing any of the parameters, and we stopped.

Question 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

We tried OvA initially. The best results we could get after extensive hyperparameter tuning manually are as follows:
Total time taken is 0.126072 seconds
prec@1: 0.764 prec@3: 0.899 prec@5: 0.933
mprec@1: 3.611e-01 mprec@3: 5.228e-01 mprec@5: 6.086e-01

Then we tried decision tree classifier of sklearn library. The best results we could get after extensive hyperparameter tuning manually are as follows:
Total time taken is 0.055312 seconds
prec@1: 0.750 prec@3: 0.804 prec@5: 0.816
mprec@1: 4.013e-01 mprec@3: 4.958e-01 mprec@5: 5.405e-01

Finally we tried Multi-layer Perceptron classifier of sklearn and got these results which were the best so far:
Total time taken is 0.329221 seconds
prec@1: 0.917 prec@3: 0.961 prec@5: 0.964
mprec@1: 6.048e-01 mprec@3: 6.856e-01 mprec@5: 6.997e-01
As we can see, the mprec@1 keeps increasing as we tried different things, which is a good sign.
The method described above is better than using either just DT or just OvA individually to generate the prediction data.

Advantages:
1) Can be applied to complex non-linear problems
2) Works well with large input data
3) Provides quick predictions after training
4) The same accuracy ratio can be achieved even with smaller data

Disadvantages:
1) It is not known to what extent each independent variable is affected by the dependent variable. Computations are difficult and time consuming
2) The proper functioning of the model depends on the quality of the training

Reference

https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html