CS771 - Introduction to Machine Learning - Assignment 2

Problem 2.1 - The Code Corrector

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.

1. Model Selection

We first tried Decision Trees for our model as they generally have faster prediction. But we could not obtain 100% accuracy even on increasing the height of the tree a lot - we found out that this was because the data in the train file is corrupted and some data points are classified as two separate 'error labels'. This gave us two major advantages with using OvA as compared to Decision Trees -

1. Since the height has to be increased for a more accurate model, the Decision Trees model has a larger training time. The saving and loading time of classifier was also high for DT.
2. This increased height also led to a much larger model size than OvA.
3. Bagging classifier with Random forest classifier as base estimator in the normal DT model did increase accuracy but the issue of large model size still remained.
4. With Gradient Boosting Decision Trees, the accuracy as well as mprec and prec scores suffered

Then we experimented with Neural Networks for solving the problem. Initially, we were getting worse prec and mprec scores than OvA and a bigger model size. But after tuning the hyperparamteres involving neurons and dropout rate we were able to achieve much better accuracy scores in Neural Networks and a comparable model size. **Thus we used Neural Networks method to solve the problem and obtain the final predictions.**

2. Data Pre-processing

1. As mentioned above, we observed was that some of the data was mislabeled. The exact same 'errors' were labelled differently. To rectify this, we only kept the most frequent label for a particular input.
2. **Data Encoding** We directly used sparse matrices instead of making them dense. Using sparse functions allowed us to reduce the pre-processing computation as we did not have to convert data points and labels to one-hot vectors.

3. Parameter and Hyper-parameter Tuning

We tried varying the following things

1. **Number of inner layers** - We tried a network with one and two layers. We found that one layer is sufficient and an additional layer does not appreciably increase accuracy, but requires more time to train.
2. **Number of neurons** - We tried layers with 32, 64, 128 and 256 neurons. Out of these, 256 was giving the best score. But on further investigating scores on 216, 240 neurons, we ended up going with 225 neurons as this lead to the highest accuracy on the out of sample testing dataset.
3. **Activation Function** - We tried using both sigmoid and relu. Due to speed considerations (exponentiation in sigmoid is expensive), we ended up using relu. There was no significant difference in the accuracy achieved in either case.
4. **Regularization** - We tried dropout between the range of 0.1 and 0.4 to prevent overfitting. Optimal results were achieved at 0.25

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.

Following are the statistics of the two Decision Models - Random Forest and Vanilla DT, Neural Network and OvA methods that we tried.

Criterion	Random Forest DT	Vanilla DT	OvA	Neural Network
Training Time	23 s	**0.140355 s**	40.29625 s	45 s
Loading Time (for uncompressed model)	2.237570 s	0.005609 s	**0.000900 s**	1.180001 s
Total Prediction Time (Loading Time + Time taken to predict k topmost classes)	3.563355 s	**0.022778 s**	0.031400 s	2.413176 s
Model Size (without compression)	675MB	600KB	**85KB**	726 KB
Model Size (with compression)	29 MB	-	-	-
prec@1	0.756	0.760	0.676	**0.827**
prec@3	0.896	0.855	0.897	**0.941**
prec@5	0.943	0.881	0.946	**0.970**
mprec@1	0.3995	0.4472	0.6205	**0.6435**
mprec@3	0.5458	0.5616	0.8014	**0.8271**
mprec@5	0.6714	0.6330	0.8486	**0.8960**

Loading Time is the time taken to load trained model from a file.

Both the Decision Tree models here are of optimal height (which comes out to be 15) to give the best accuracy. From the table, we can obsesrve the following.

Disadvantages of Neural Network:

1) **Complexity:**
   - The classifier obtained after training in **OvA** is quite simple. If there are k classes, then the model will have k ω and k bias terms only. Hence the classifier/model size is lowest among all.
   - For **DT**, in case of large datasets, Decision-tree learners can create over-complex trees that do not generalize the data well. The model size for DT is larger than OvA because of the accuracy-height trade- byoff in the former.
   - The model trained **Neural Network** is also quite complex and has significant size compared to OvA.
2) **Training Time:** Neural Network has a much larger training time than both the DT models and a slightly larger training time than that of OvA.
3) **Prediction Time:** The prediction time for Vanilla DT and OvA is comparable but much less than that for Random Forest DT. Since the model size for Random Forest Classifier is quite big, most of the prediction time is spent on loading the model from file. For Neural Network, loading and prediction took similar amounts of time.

Advantages of Neural Network:

1) **Accuracy:** The prec score of OvA, while lower than Vanilla DT, is comparable to that of Random Forest DT. However, OvA performs much better with mprec score as compared to both the DT models. Here, **Neural Network** outperformed even OvA, leading to a final prec score of 97% and an mprec score of almost 90%.
2) **Ease of Implementation:** Due to the availability of handy libraries, coding with Neural Networks is much easier as compared to OvA.