CS5304 Assignment 4 - Playing with Deep Learning

Group Members

Kuo-wei Tseng (kt535)
Chong-yee Gan (fc297)

Instructor

Giri Iyengar

Table of Contents

Section 1: Dataset Details

The anonymized advertising dataset consisted of 2 class labels and approximately 20,000 sparse boolean features. Each row corresponded to a user and each column a signal that they expressed, such as a site they visited or a demographic segment they belong to etc. The data was in SVMLight format, and consisted of training, validation and testing sets:

Data Set	Number of samples	Number of Features	Number of Class Labels
Training Set	5,000,000	20,000	2
Validation Set	2,500,000	20,000	2
Testing Set	2,500,000	20,000	2

Table 1: Summary Statistics of Dataset

Section 2: Logistic Regression

2.1 Model Training

The sklearn.linear_model.LogisticRegression module was used to train the model. Whilst we tried putting all training data to fit the model, this endeavour proved inefficient because if the training data is too large, sklearn logistic model will not use multi-core properly and it only uses one core to fit while the AWS EC2 module has 36 cores.

Thus, we used two different way to solve this. We first tried using stochastic gradient classifier in sklearn to divide the training data into smaller batches, and using partial_fit function to train the model. In this case, AWS module can use all cores (36 cores) to train, but the performance of SGDClassifier was slightly lower than LogisticRegression model. (test accuracy drop from 0.905 to 0.893).

Secondly, we tried using smaller data set to train. In this case we use 1/10 of original training set to fit the model, and then we do cross-validation to find best C (regularization term), and used sklearn **CalibratedClassifierCV(5-fold)** to find a better result. We put a pre-fitted base estimator into CalibratedClassifierCV and use validation set to calibrate the result.
We also trained on **whole data set**, but this part we skip cross-validation selecting C part. We still do **CalibratedClassifierCV** fit in this part.

2.2 Model Evaluation

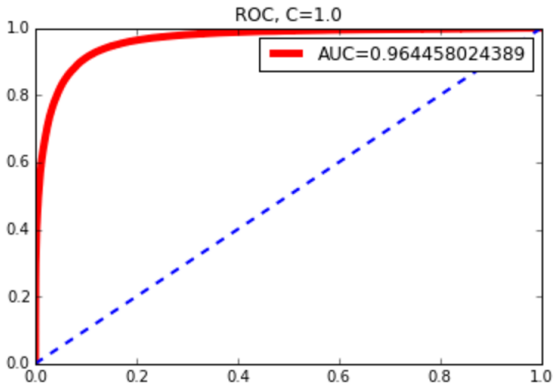
Here are some results of our logistic regression.

1. 1/10 Data Set:

Our accuracy of directly feeding training data to default LR Classifier:

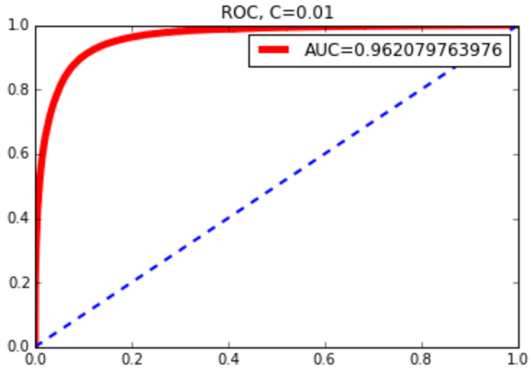
```
Accuracy:  0.905838853922
Lost 0.245712002355
```

ROC curve: (test on 1/10 training set)

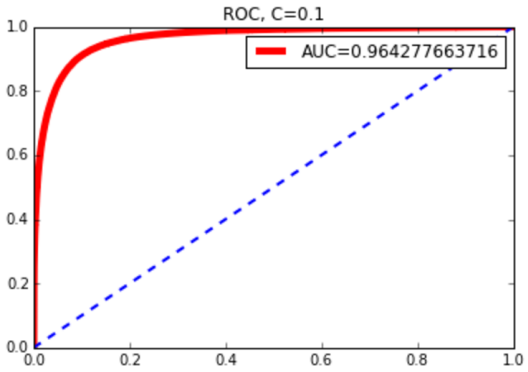


Cross-validation results:

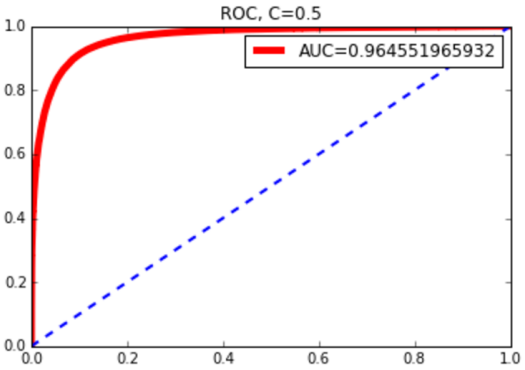
C=0.01



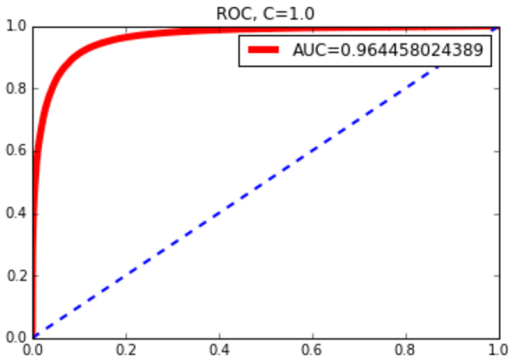
C=0.01



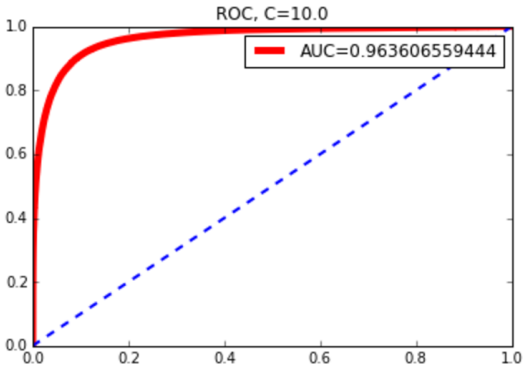
C=0.5



C=1.0

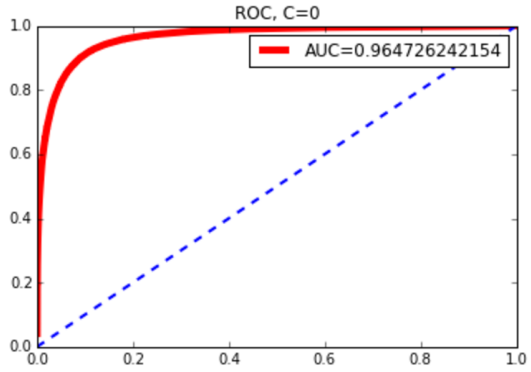


C=10.0

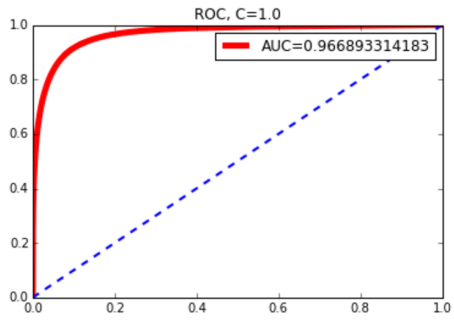
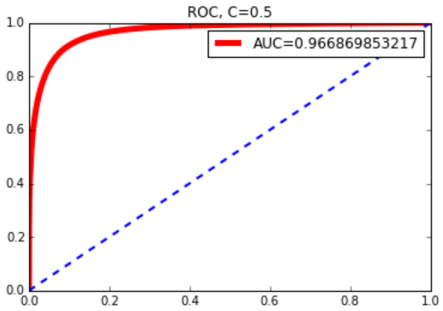


2. CalibratedClassifierCV on 1/10 dataset:

```
Accuracy:  0.909370955321
calibrated score (5-fold:) 0.23242860372
```



Whole Data Set (Base v.s. Calibrated):

	Base	**Calibrated**
ACC	`Accuracy:  0.908175114672` `Lost 0.237142791181`	`Accuracy:  0.91107498371` `calibrated score (5-fold:) 0.225667403633`
ROC		
Brier_loss	`Brier Score: 0.0684220752001`	`Brier Score: 0.0663818392683`

Although whole data set training accuracy drop down slightly, the testing accuracy improved by 0.5% and AUC. This means that our model is more generalized and able to deal with testing data we never learn before better.

Section 3: Multi-Layer Perceptron

3.1 Model Training

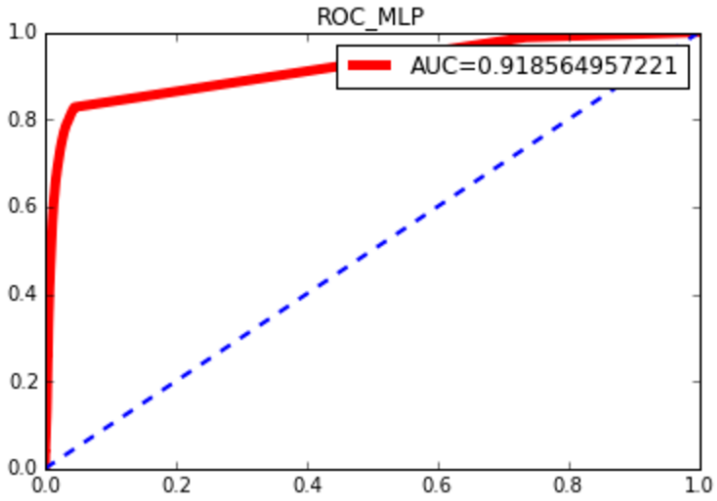
The Multi-layer Perceptron model was trained using Tensor Flow with 2 hidden layers and 50 hidden nodes. The attributes of the training data was first defined using the placeholder_inputs() function for both the features (X) and labels (y). Thereafter, the neural net layers were built before training the data on subsequent steps of batch size = 100.

Whilst we initially trained the model with a loop of 50, our kernel restarted when we exceeded the AWS memory limit during model evaluation, when we tried to calculate their probabilities. Although we managed to solve this problem later on (Section 3.2), our final results were from a model that was trained with only 5 loops due to time limitations.

3.2 Model Evaluation

Once the model was trained, the results were tested in terms of accuracy, ROC and AUC. The ROC and AUC was determined by first finding the probabilities via y.eval() method, and plotting the corresponding curve using scikit learn. However, due to memory limitations, these probabilities were evaluated in batches, before being appended to a final array. These outputs were also calibrated, to be compared for their Brier loss and Brier curves, as shown below:

```
Lost 0.361089439128
Brier Score: 0.0713458523226
```



As mentioned in Section 3.1, our model were only trained for 5 loops. Thus, it had yet to converge, and consequently produced results that performed worse than Logistic Regression.