IN4320: Machine Learning Final Assignment

Zheng Liu (4798406), NetID: liuzhengliu

July 7, 2019

As required, I put my loss in the first page. This is my estimated loss: 0.6527.

1 Data Preprocessing

For both training and testing data the following preprocessing methods were applied to attempt to speed up the training process and improve the accuracy.

1.1 Numerical features

The numerical features 1-6 were normalised into range $[0, 1]$ respectively via equation 1 which is also known as min-max rescaling.

$$x_i^{'} = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{1}$$

where x_i is the original value in the i^{th} dimension, $x_i^{'}$ is the normalised value, $\max(x_i)$ and $\min(x_i)$ are the maximum and minimum value in certain dimension respectively.

1.2 Categorical features

As described in [1], some learning algorithms cannot deal with categorical data directly, poor performance and unexpected results would occur since the model defaults that there is a nature order of integer categorical data. I applied the "One Hot Encoding" to deal with the problem, the example is shown in Figure 1. Each original feature vector would be transferred into a binary matrix where the corresponding position is 1 and all the others are 0.

1.3 Feature 14

For feature 14, as there is not such a feature in test set, I decided not to use it in training process and simply just dropped it out in programming, taking feature 1-13 only. The reason is that, firstly, if I used it for training, the classification mapping could not fit test data as there are only 13 dimensions but classifier is the mapping for 14 dimensions input; secondly even I reduced dimensions from 14 into 13, the classifier containing information with feature 14 would be tricky for testing data. So, I decided to ignore this feature.

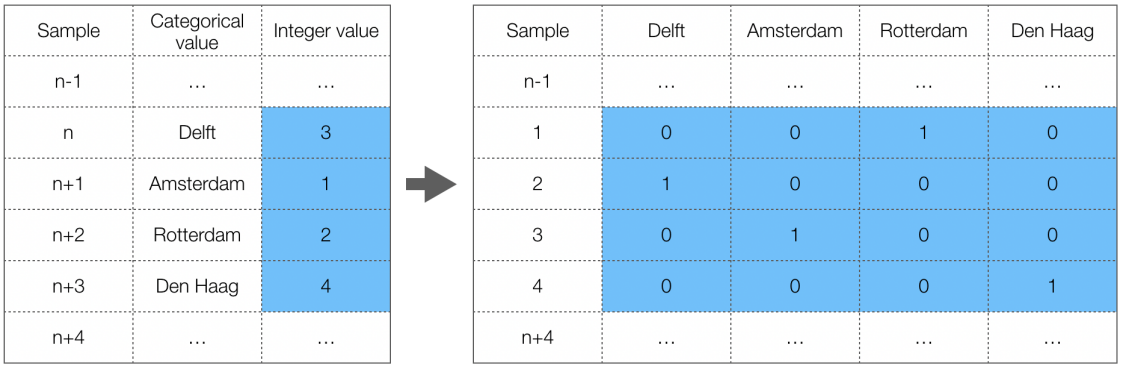

Figure 1: One Hot Encoding demonstration example

	KNN	Linear Regression	Logistic Regression	Naive Bayes	AdaBoost
Loss	0.8791	0.7508	0.6952	0.7289	0.6527
Training error	75.49%	17.03%	15.22%	15.91%	14.27%

Table 1: Classification performance in different classifiers

2 Classification

I tried several classical classifiers:Linear Regression, Logistic Regression, SVM, KNN, Naive Bayes, AdaBoost and Random Forest, then randomly chose 16000 training data and 4000 validation data from original `train.csv`, repeated the experiment 100 times for each classifier. For SVM and Random Forest, the computation cost is rather high because of the high dimension issue, so I did not show it here. The result is as follow, shown in Table 1:

By modifying the size of training and validation set, together with (16000,4000), (12000, 8000), (8000,12000), (4000, 16000) and (2000,18000), I obtained the learning curve as in Figure 2. From this error-bar plot we can see that our trained model is not over-fitting. It is suitable to choose the classifier trained by 16000 samples. I obtained the `.csv` file via `test.csv` and my classifier as `labels.csv` which is attached with my submission.

3 Constrain and Loss analysis

These following constrains mean given a certain label, the probability of the sample being misclassified should be similar with different value of feature 13, which is described as the fairness to feature 13. Also we can see from the loss function, if the probabilities are not equal respectively, the higher one would be used to calculate the loss, in other word, the better the constrain is fixed, the lower the loss would be.

In my implementation, the classifier highly satisfies the constrain in equation 2:

$$P(c(X) = 1|L = 2, X_{13} = 1) = P(c(X) = 1|L = 2, X_{13} = 2) \tag{2}$$

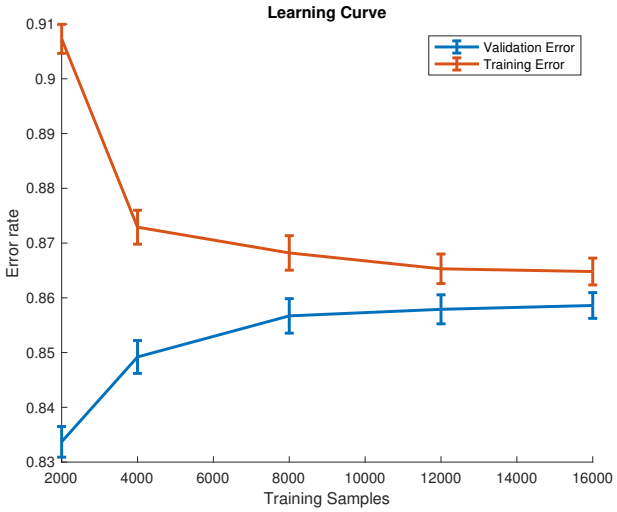

Figure 2: Learning curve

however, although I already achieved accuracy 85.73%, and obtained one of the constrain, I find it hard to fit the other constrain in equation 3:

$$P(c(X) = 2|L = 1, X_{13} = 1) = P(c(X) = 2|L = 1, X_{13} = 2) \tag{3}$$

As I achieved my loss with training and validation set, the detailed calculation of loss=0.6527 is: $(p_{2|11} = 0.0772, p_{2|12} = 0.0268, p_{1|21} = 0.4211, p_{1|22} = 0.4179)$, 3.0772+0.4211=0.6527.

4 Result analysis

I was interested in the relation between value of loss and value of error rate (1-accuracy). For explore whether they are related to each other, I chose the classifier I used and trained it 1000 times and obtained 1000 loss and accuracy, I plotted them shown in Figure 3.

I tried to fit the curve, the result illustrates that loss and accuracy have negative correlation. Loss would decrease with the increments of accuracy, thus it has positive correlation with error rate.

Another thing to say is that as we do not have any standard to evaluate the loss value, I say the accuracy 85.73% $(1 - 14.27\%)$ is intuitively a good performance.

5 Conclusion

After whole assignment, I have some ideas might improve the performance but due to time limit I put them below as my discussion and conclusion. For data preprocessing, I did training set and test set separately, I thought to do normalisation and One Hot Encoding with training and testing data together and then split then would be better, because they would have a unified standard, for

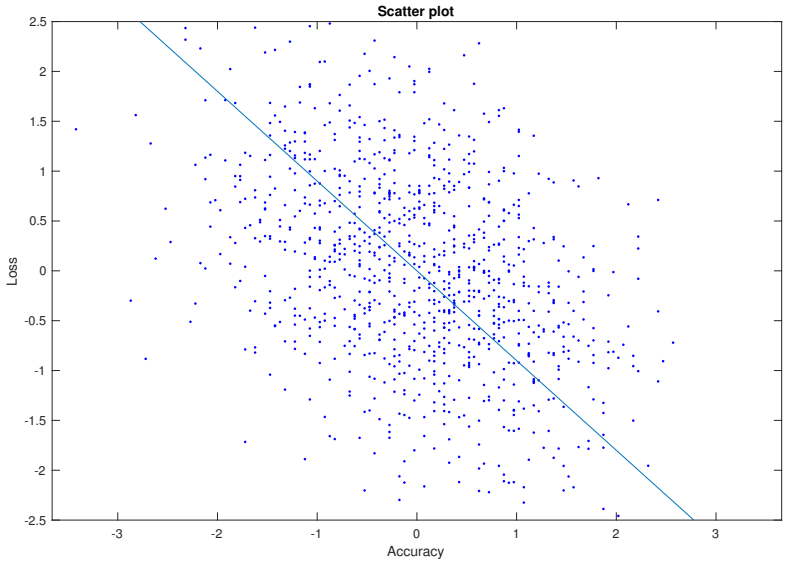

Figure 3: Curve fitting of accuracy and loss

example, a same value in training and testing set would have different normalised value, or there are 42 categories in feature 8 in training set but only 41 in test set, which may cause dimension chaos. Secondly, as I applied OHE and consequently made rather large dimension size, I would like to use PCA technique to reduce the dimension. Next, I may try some other methods to fix the constrain. Last but not least, it would be interesting to explore the relation between loss and accuracy mathematically.

In total, there are 792 words including this.

References

[1] Jason Brownlee. Why one-hot encode data in machine learning? `https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/`. July 28, 2017.