CS583A: Course Project

Mahmoud Ayyad

May 16, 2019

1 Summary

I participated in an active competition of binary classification of the customers who will make a specific transaction. The final model we choose is a deep convolution neural network architecture, which takes information about unreal customers with the same structure as the real data and the corresponding class label. We implement the convolution neural network using Keras and run the code on a Dell precision workstation with one Xeon W-2155 CPU and 64 GB memory and one NVIDIA Quadro P2000 GPU. The performance is evaluated using the Area Under the Receiver Operating Characteristic (ROC) curve (AUC). In the public leaderboard, my score is 0.872; I rank within the early $6,000$s among $8,802$ teams. In the private leaderboard, my score is 0.872; I rank in the end of the $5,000$s among the $8,802$ teams.

2 Problem Description

Problem. Santander is a well known bank which works hardly to help its customers to understand their financial capabilities and to identify the products and services that might help them to achieve their goals. In this competition, it is required to identify which customers will make a specific transaction regardless the amount of money. Thus, the problem is as a binary classification, whether the customer will accept or decline the transaction. The competition is at `https://www.kaggle.com/c/santander-customer-transaction-prediction`.

Data. The data is defined by a 200 features with one corresponding label. The number of training samples is $n = 200,000$. The number of classes is 2. The training set is imbalanced: $n_{\text{accept}} = 179,902$ and $n_{\text{decline}} = 20,098$ which means that about 10% declines the transaction and the rest accept it.

Challenges. The problem is hard to solve as the data is extremely imbalanced. The undersampling decreases the training set dramatically and lose some important features in training while the oversampling generates repeated or wrong extra data.

3 Baseline

A Logistic regression is used as the baseline. The lbfgs optimizer is used. A $3,000$ maximum number of iterations. The area under the curve is 0.63.

4 Solution

Model. The model I finally choose is a standard deep convolution neural network. The model consists of three hidden layers. The first layer has $20,000$ nodes while the second hidden layer has a $2,000$ nodes and the last hidden layer has 200 nodes. A ReLU activation layers are used. For the output, the sigmoid activation layer is used. A 30% dropout is used before the second the hidden layer. A total of $64,422,401$ parameters are trained using the $200,000$ data samples. The model summary is shown in the table 1

Table 1: The convolution neural network model summary

Layer Type	Output Shape	Parameters #
Dense	(None, 20,000)	24,020,000
Dropout	(None, 20,000)	0
Dense	(None, 2,000)	40,002,000
Dense	(None, 200)	400,200
Dense	(None, 1)	201

Implementation. I implement the convolution neural network using Keras with TensorFlow as the backend. I run the code on a Dell precision workstation with one Xeon W-2155 CPU and 64 GB memory and one NVIDIA Quadro P2000 GPU. It takes about 22 minutes to train the model.

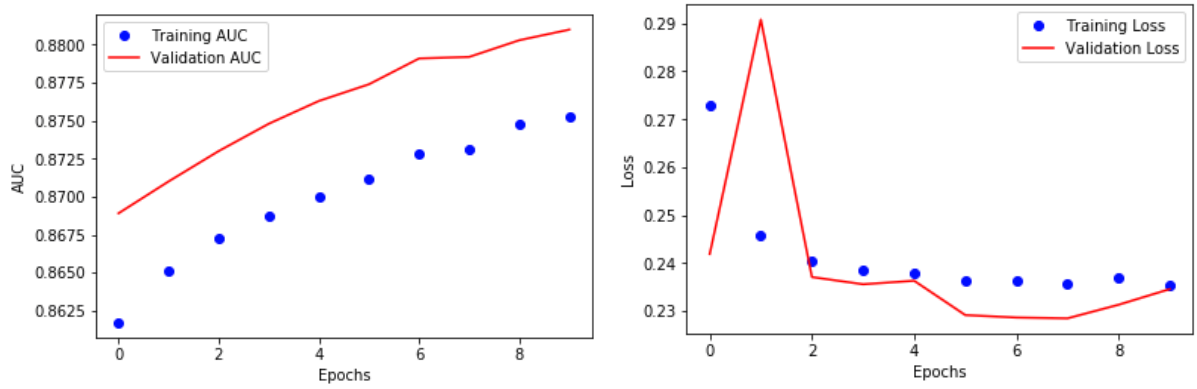
Settings. The loss function is the binary cross-entropy. The opitmizer is RMSprop with learning rate $1e-4$. 10 epochs and 128 batch size are used. A 30% dropout is used before the second hidden dense layer.

Advanced tricks. I added couple of new features to have higher order of error. I got the second, third and fourth power of the features. Also got the rank and the normal cumulative density function of the features. This improved the validation error by 1.2%. Also, I scaled all the data as in equation 4.1 which improved he validation error by .

$$\text{scaled data} = \frac{\text{data} - min(\text{data})}{max(\text{data}) - min(\text{data})} \tag{4.1}$$

Moreover, I used a 30% dropout to alleviate the over fitting.

Cross-validation. I tuned the hyper-parameters by partitioning the training data to 90%-10%. Figure 1 shows the convergence curves on the training and the validation data. By increasing the number of epochs, the validation AUC decreases while the training AUC keeps increasing. Thus, I used 10 epochs to train the model.



(a) The AUC on the training set and validation set.

(b) The loss on the training set and validation set versus the number of epochs.

Figure 1: The convergence curves.

5 Compared Methods

The autoencoder is used. The validation AUC is 0.863 which is slightly less than the AUC of the convolution neural network. Figure 2 shows the extracted validation features. Figure 3 shows the convergence of the autoencoder loss. The autoencoder shows that the data are very overlapped and it is hard to classify it with normal classification methods (like SVM).

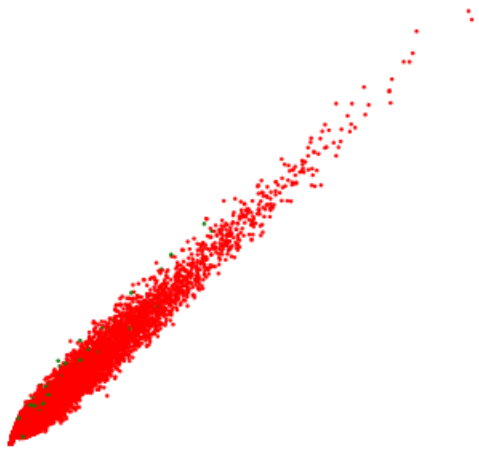


Figure 2: Extracted features of validation.

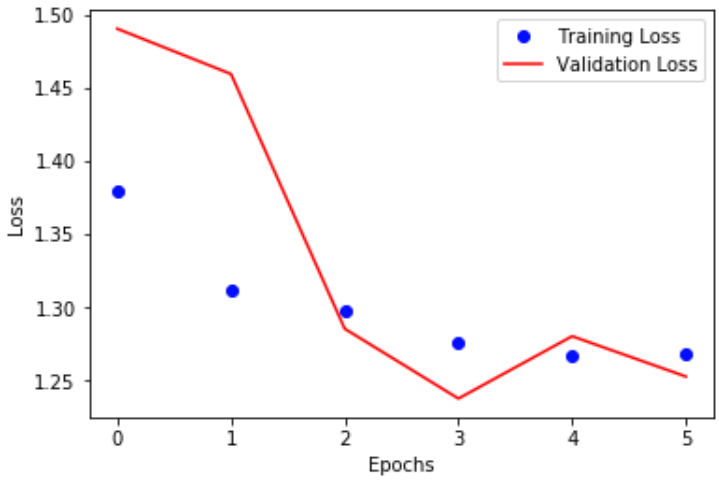


Figure 3: The convergence curve.

6 Outcome

I participated in an active competition. My score is 0.872 in the public leaderboard and 0.872 in the private leaderboard. I rank around the 6,000s/8,802 in the public leaderboard and end of the 5,000s/8,802 in the private leaderboard. The CNN model has higher AUC than the baseline model, which is based on the logistic regression.