Deep learning Assignment1

Jaw Yuan,Chang

April 2019

Introduction

The purpose of this assignment is walk through the underlying for whole training process and implement both forward propagation and backward propagation. This assignment will also analysis the training process and the discussion of the features of the training data.

Method

In this project, deep neural network has been completed with Fully Connected layer and activation layers include ReLU and Softmax.

The first model's architecture is designed very similar to the second model, which is provided by TA. However I choose to increase the neural number of each layer. There are 48000 parameters in training data and the second model has 41 parameters that available for training. Hence, I decide to increase the parameters that is trainable to 176 which is nearly 4 times compare to the second model. and the neural layer by layer is chosen to be [6 12 6 2] with ReLU layer between each layers output and input. The main reason discard the sigmoid layer is due to the instability of exponential function. I have counter many error in progress construct with Softmax layer because the exponential term will lead gradient exploding. The batch size is chosen to be 16 which is the smallest recommended number because the training data only has 800 piece. The learning rate(1) is chosen to 0.01 which is the default setting of pytorch and then changes to 0.099 for better performance The decay is chosen to be 0.97 which is empirical number that suit for this project. The initailize weights is Gaussian function and the multiple it with $\frac{1}{\sqrt{m}}$, where m is the number of input and than multiple it with std 1e-2 because it is essential to keep the weight from exploded. The coding style has referred to CS231n(2)

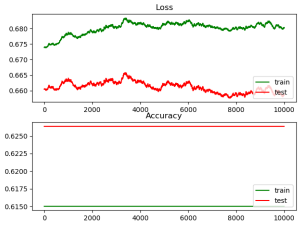
The second model framework is provided by TA which is the sequential fully connected layer and the neural is arranged by[6 3 3 2]. The activation function between these neural is ReLU expect the last layer which is Softmax. The weight has been initialized by normal.

The second model is design base on some principle: The activation between layers is ReLU, without using the exponential function can accelerate the training speed of

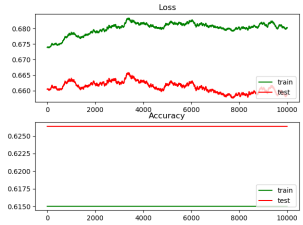
neural network and prevent the model from gradient exploded.

Result

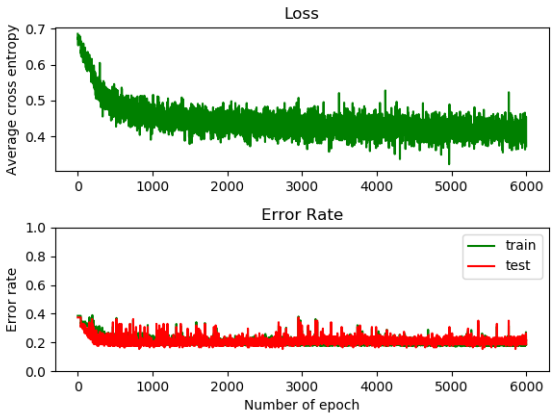
In the first model, the starting point of weight is chosen as normal distribution without multiple the standard deviation. However, this setting is totally catastrophe because the learning curve and the accuracy curve in Figure 1(a)(b) has shown that the model has learned nothing. The reason is because sometime the weight is too big and it eventually goes to infinite and the model will not update anymore. Hence, it is very important to have a correct starting point which is shown in Figure 1(c)



(a) model1



(b) model1



(c) model1

Figure 1: Learning carve and error rate of model 1

In the second model, based on the experience of first model, the result is shown in Figure2. From the result, it seems that reduce the trainable parameters will makes the loss curve more concussion and require more epoch to convergence.

If the data has been normalization(3), which I have implemented not only Fare but also age and Pclass because by observing other data by column these data are exceeding the range of $[0, 1]$.This may cause some unwanted effect when updating the DNN. The result of normalization has shown in Figure 3. It is important here to revisit the initializing the weight because in the previous step, the smaller weight is desired to prevent model form exploded. However, as all the parameters input has

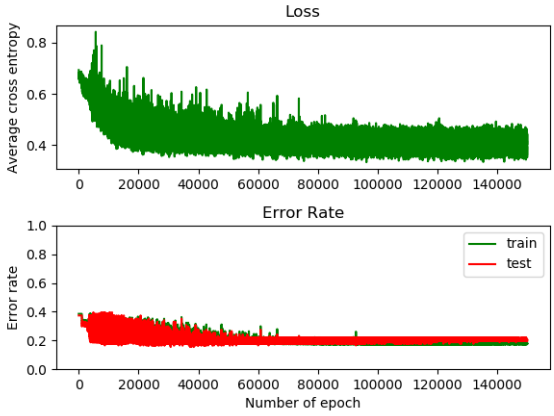
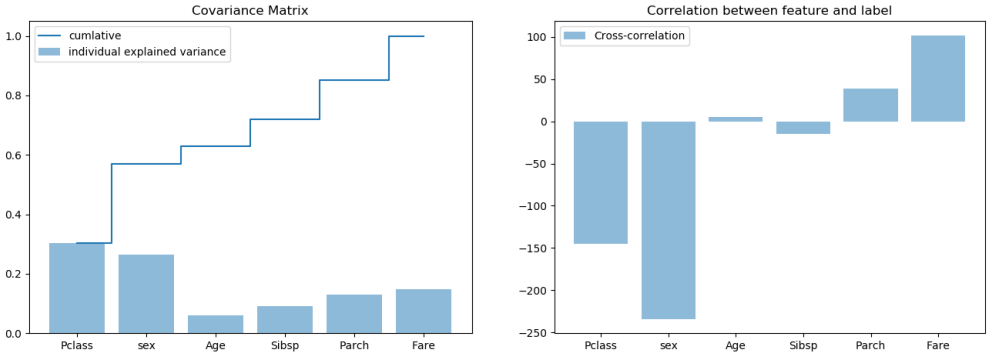


Figure 2: Learning carve and error rate of model 2

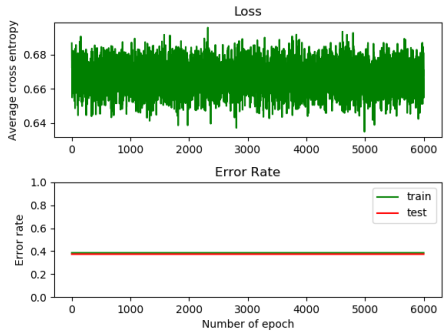
been normalized the chance of gradient exploded could become smaller. Also, the standardized data can do something with training a model Figure 3. (c) shown the result of standardized data. As the learning curve shows that the oscillation of loss curve smaller and the model has the best result in the early epoch with accuracy 0.82. In order to find out where's the position with best performance, the loss curve of testing data is append on Figure 3.(d). This model has just fit as epoch 200.

There's several ways to find out which feature dominates model the best. The first method is from the machine learning which is construct a covariance matrix and observe which feature dominated the data at most. It looks like Pclass, sex, and fare has dominated the data in the Figure 4.(a) However, the data contains the label. It is important that checking the correlation between the label and data. As the result shown in Figure 4.(b) The Sex has dominated whether you will survived in Titanic. The performance will be affect if the most correlation remove from the input.

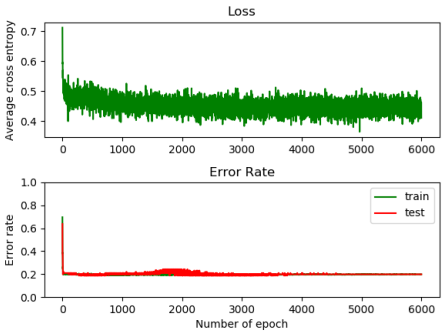


(a) Covariance Matrix (b) Correlation

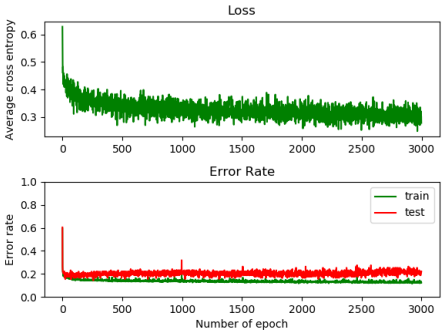
Figure 4: Feature dominance



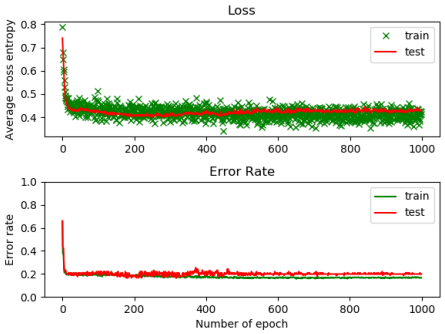
(a) smaller weight initial



(b) bigger weight initial



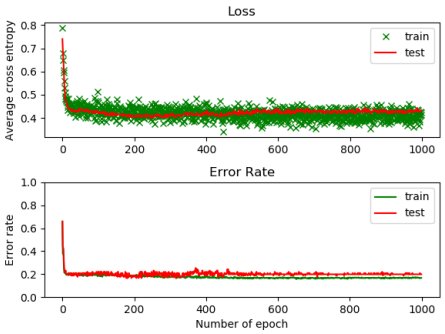
(c) standardized



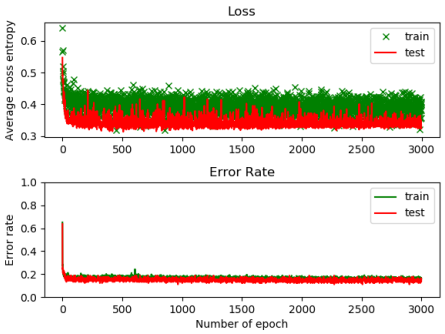
(d) standardized

Figure 3: Learning carve and error rate of Normalization input for model 2

There's still some preprocessing that can boosting the performance which is transfer Pclass by one hot decoder. Since the relationship of Pclass can not described surely by a single scalar it is better to decode data into one hot type and also modify the model architecture to [8 3 3 2] .



(a) Without one hot



(b) One Hot

Figure 5: One hot data

Although the loss curve looks more oscillated the best performance has increased for 6% in my model and the loss curve and error rate have been shown in Figure 5.

Since the project walk through many relationship between data. I'll design my own data with Pclass = 1, Sex = 0, Age = 5, SibSp = 1, Parch = 0,Fare = 500. I expect this person will survive. The output of model is survived. This is because this person is a female, her Pclass is 1 and the Fare is high which makes her survived.

Discussion

At the beginning, It is frustrated that the model has little chance to fit the data. The loss will stop updating at the certain point and lead the model shutdown. After long time of tunning, finally I get the result approximate to 0.81. It is an exhausted process.

Conclusion

It is very important to implement the neural network form underlying. There's plenty bugs and problem to overcome. However, training every question for a brand new hyper parameters takes too much time, the hyper parameters is same for question 2-5.

References

[1] PyTorch,"TORCH.OPTIM",https://pytorch.org/docs/stable/optim.html.

[2] CS231n,"Assignment 1", http://cs231n.github.io/neural-networks-2/.

[3] Sklearn,"Data preprocessing", https://scikit-learn.org/stable/modules/preprocessing.html