Practicum Report - Deep Learning Model:

1. Data Processing
2. Data Leakage Check
3. Feature Engineering
4. Normalization
5. Model Structure
6. Hyper-Parameter Tuning
7. Model Result and Validation
8. Model Comparison

III. Feature Engineering

Income normalization by zip code:

We want the income feature to truthfully reflect the level of income of each applicant. This means that an 80K income in Champaign is not the same as 80K income in Los Angeles, because average income in LA is much higher than Champaign. Thus, raw income data are not a good indicator of applicants’ income level. Therefore, I normalized income distribution to N(0,1) distribution by each applicants’ zip code. Therefore, income information reflects the income level in each applicant’s living region.

IV. Standardization

Normalize Train/Test Set Separately:

To ensure our model best reflect the real-world problem, we have separate the train/test set before data standardization. Following our assumption in data leakage check, the test set should not have any knowledge about the distribution of the data, we conduct train/test split before data standardization. We compute the mean and standard deviation of each features and stored them in a dictionary. Then, I used the train sets’ mean and standard deviation to normalize test set. In practices, when new data arrives for prediction, its features will be normalized using passed mean and standard deviation we learned from pass data.

V. Model Structure

Deep learning model consist of 7 dense layers with 6 dropout layers in between. The first layers consist of 57 neurons corresponded to 57 input features, 3 layers consist of 64 neurons, then 2 layers consist of 32 neurons, and the last layers have 16 neurons. There are 6 dropout layers between each layer. The output layers consist 2 neurons for two classes of outputs. We used Adaptive Moment (ADAM) optimizer and sparse categorical crossentropy for optimization method and loss function.

I compare 3 model designs: gradient descent, Adam optimizer, and Adam optimizer with dropout layers. Experiments shows that Adam optimizer with dropout layers shows the best performance.

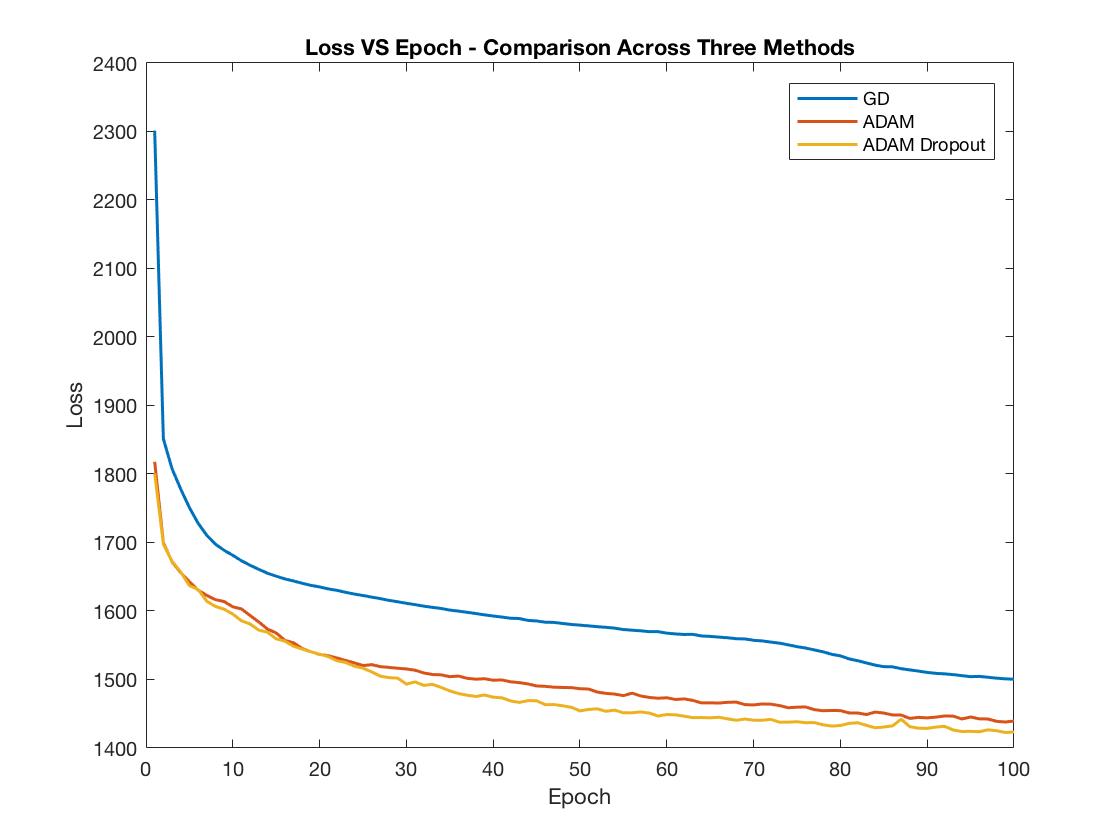


Figure 1: The ADAM Optimizer with dropout layers shows the best performance. Please note that his plot is generate using model under TensorFlow framework. Later, I wrote and tested my final result in Keras. The loss in Keras is not comparable with this plot. This plot helped us determined optimizers and set a foundation for our final model.

|  |
| --- |
| **Optimizer** |
| In our deep learning model, the model is trained to minimize the **objective function** – Sparse Categorical Crossentropy. Thus, we need to choose the best optimization algorithm for our problems. The two candidates are Gradient Descent and ADAM.  **Gradient Descent:** Gradient Descent (GD) tries to minimize the function by iterative moving in the direction of steepest descent. In deep learning model, the descent is used to update parameters for neurons.  is the parameter that needs update and is the learning rate.  (Goodfellow, Ian; Yoshua Bengio; Aaron Courville n.d., 84)  **ADAM:** ADAM is a stochastic gradient descent optimizer (SGD). Similar to GD, ADAM introduced an improvement over traditional GD with fixed learning rate. Traditional GD used a fixed learning rate, while ADAM uses an adaptive learning rate. ADAM optimizer will adjust the value of as it gets closer to the minima. Thus, it generates a better result than fixed learning rate GD optimizer. ADAM also wins against other SGDs.  (Diederik P. Kingma;Jimmy Lei Ba, 2017) |

VI. Hyper-Parameter Tuning:

We want to choose the best hyperparameter for our model. I first test 3 optimization method: gradient descent, ADAM, and ADAM with dropout. As we can see that the ADAM with dropout layers tend to perform the best. Also noted that this model is not our final model, I plotted this in TensorFlow framework. Later, I wrote my final model in Keras. Therefore, the numerical value of losses is not comparable. The key is ADAM shows better performance than regular gradient descent because ADAM adjust learning rate as the model trains, thus the model can reach closer to the optimal weight than gradient descent optimization method.

After we determine the optimizer. I used a grid search to find the best hyper-parameters on following choices: batch size: [8, 32, 64], learning rate: [0.00001, 0.0001,0.001], neurons initialization method: [uniform distribution, normal distribution], and Dropout rate: [0.0, 0.2, 0.4, 0.6]. The complete grid search result will be attached in appendix.

Limited by our computation resources, I was unable to use a bigger grid. GPU implementation might help accelerate this process. I used amazon AWS cloud computing services to run this grid. In order to reduce the number of combinations and accelerate the process, I fixed number of epochs to 10 and used 3-fold cross validation. I also fixed dropout rate to 0.0 in the grid and tuned it manually after other optimal hyper-parameters have been found. My grid search returns the best hyper-parameters:

Best: 0.867080 using {'batch\_size': 8, 'dropout\_rate': 0.0, 'epochs': 10, 'init\_mode': 'normal', 'learn\_rate': 0.001}

However, I realize that this result is not optimal under our fix 10 epochs grid because the grid returned the largest learning rate and smallest batch size. This essentially means that the optimal set of hyper-parameters is the one that trains the model with most iterations and larger learning rate. However, this doesn’t result the optimal performance when you increase the number of echoes. We want a set of hyper-parameters that will return the best performance when the model slowly converges to the global minima.

Instead, I used the second-best set of hyper-parameters as the starting points.

0.867049 (0.002364) with: {'batch\_size': 64, 'dropout\_rate': 0.0, 'epochs': 10, 'init\_mode': 'uniform', 'learn\_rate': 0.0001}

Then, I increase the number of epochs to 1000 and test a set of dropout rates [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]. The test shows the optimal dropout rate here is 0.1.

|  |
| --- |
| **Hyper-Parameters** |
| **Batch Size:** batch size refers to the number of training examples utilized in one [iteration](https://radiopaedia.org/articles/iteration-machine-learning). Here we used 64 as batch size, so we utilized 64 training examples in one training iteration. Usually, larger batch size provides more accurate estimate of gradient.  **Dropout Rate:** dropout is a common way to prevent overfitting in deep learning model. Dropout layers randomly discard units in each hidden layer. Dropout rate is the percentage of neurons to be discard in each hidden layer. We found that dropout rate of 0.1 works best in our model.  **Epochs:** one epoch is single pass of the full training set. For a stable model, larger number of epochs will not hurt the performance. |

VII. Model Result and Validation

Our model trained using tuned hyper-parameters on 1000 epochs. It results 89.41% mean accuracy on test set. The following two figures summarized our result on test set. Our model result 90.14% accuracy on train set. Our model shows little overfitting issue due to our dropout regularization.

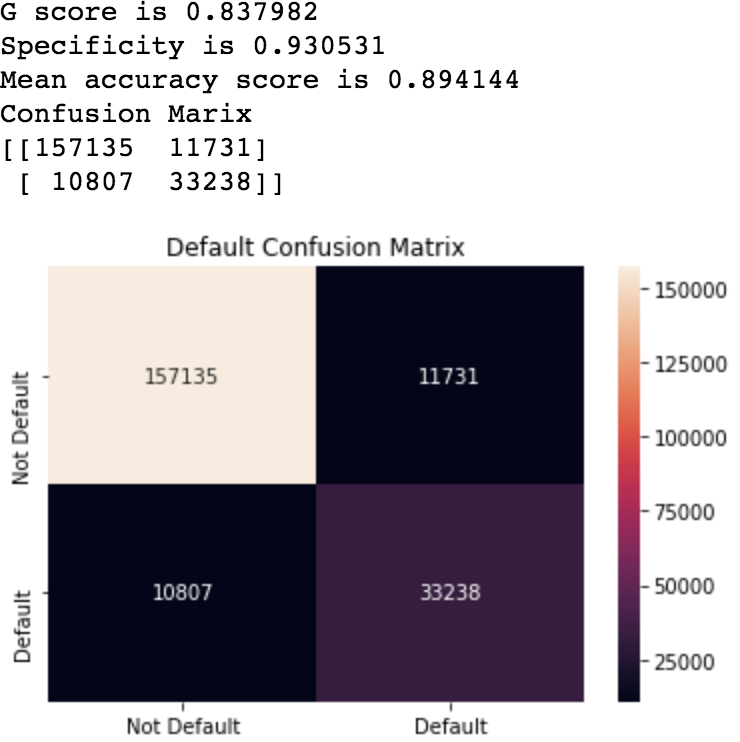
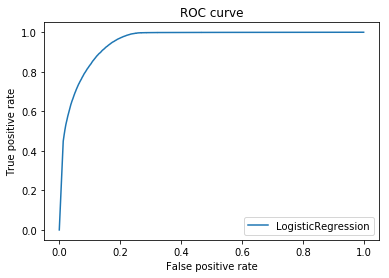


Figure 2: ROC curve on test set. AUC = 0.953250. The x-axis is Figure 3: Confusion Matrix on Test set

false positive rate and the y-axis is the true positive rate.

There is a trade-off between false negative and false positive.

|  |
| --- |
| **Understand Model Performance** |
| **Specificity:** specificity is the true negative. It measures the percentage of not default loan who are correctly identified as not default. rate.  **Sensitivity:** sensitivity is the true positive rate. It measures the proportion of default that are correctly identified as default.  **Accuracy:** accuracy is the percentage of correct predictions over size of test set |

Validation:

For model validation, I first test if my model can overfit the training set with enough epochs of training. The assumption is that any appropriate machine learning model can overfit the training data, otherwise your model choice is not good for this type of problem. I validate my model by randomly choice 10% data from my total training set and perform 1000 epochs to see if my model can outfit training set to 99% accuracy. I only 10% of total training set because it can accelerate the process of choosing the best model with enough layers and neurons. The initial 2-layer neural network had difficult fit the training set to 99% accuracy, after I increased number of neurons and layers, my model can easily fit training set to 99% accuracy with roughly 1000 epochs of training.

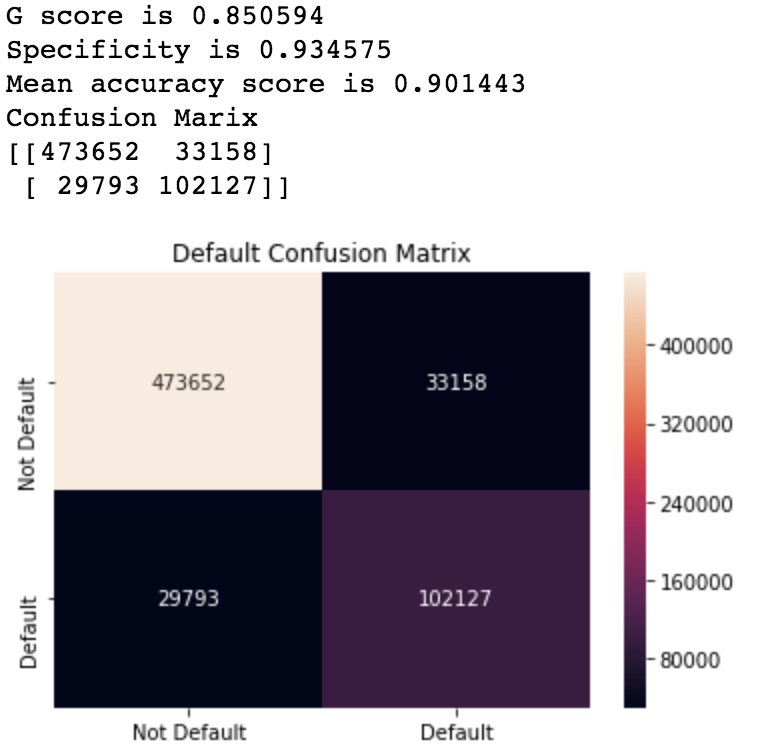
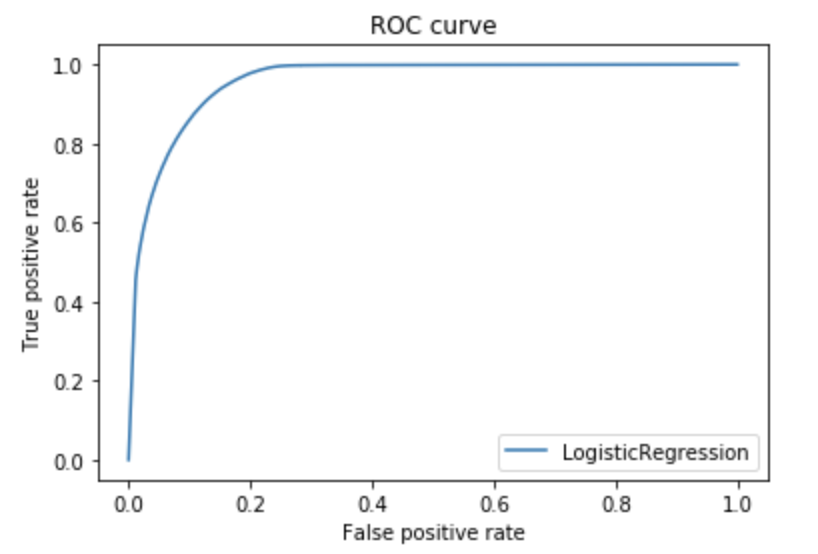
I also test my model against overfitting by comparing the model’s accuracy on both training set and test set. Following the model performance on training set.

Figure 4: ROC Curve on Train Set. AUC = 0.958079

Figure 5: Confusion Matrix on train set.

Extreme Case Test:

I also trained my model by selecting only observation with default loan status and fully paid loan status. We expected to see a 100% accuracy as sanity check.

References

*"10-Minute Tutorials with Amazon Web Services (AWS)." Amazon Web Services, Inc. Accessed April 28, 2018. https://aws.amazon.com/getting-started/tutorials/.*

*Brownlee, Jason. "How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras." Machine Learning Mastery. March 10, 2018. Accessed April 28, 2018. https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/.*

*Chollet, François. Deep Learning with Python. Shelter Island, NY: Manning, 2018.*

*PART 1: THE FUNDAMENTALS OF DEEP LEARNING*

*DanB. "Data Leakage." Data Leakage | Kaggle. Accessed April 29, 2018. https://www.kaggle.com/dansbecker/data-leakage.*

*"Deep Learning." Coursera. Accessed April 28, 2018. https://www.coursera.org/specializations/deep-learning.*

*Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep Learning. Cambridge, MA: MIT Press, 2017. Ch. 4, 5, 7*

*James, Gareth. An Introduction to Statistical Learning: With Applications in R. New York: Springer, 2014. Ch. 4, Classification*

*Ng, Andrew. "Deep Learning." Deep Learning Specialization. Accessed February 28, 2018. https://www.coursera.org/specializations/deep-learning.*