Report of StateFarm Pre-employment Assessment

Chen Zhao
1512 Meadow View Drive, Richardson, TX 75080
charliezhaoyinpeng@gmail.com, (518)977-0834

1 Data Clean

Before using the given training (*exercise_20_train.csv*) and testing data (*exercise_20_test.csv*), both of data sets went through a cleaning procedure which complies with the following steps:

- **Step 0**: Check raw data summary. This returns a simply summary, including data dimension (sample size and feature dimension) and summary of data types for each feature (*e.g. float*, *int* and *object*).
- **Step 1**: Convert *object* features into *float* format. After observation, I found that there are 6 columns whose data type are *object*. To deal with, I sub-categorize them in three:
  - Simply remove unnecessary symbols (*e.g.* $\$, \%$) and convert to *float*, such as $x41$ and $x45$.
  - Directly convert features from *object* to *category* and then *float*, such as $x34$, $x68$ and $x93$.
  - Rename values which have the same meaning (*e.g.* 'fri' and 'friday') and convert to *category* and then *float*. For example, $x35$.
- **Step 2**: Handle missing values. There are many ways to handle missing values, such as drop rows containing n/a, fill n/a with aggregated values (*e.g.* mean, median), and impute n/a with imputers (*e.g. IterativeImputer*, *KNNImputer*, *SimpleImputer*). In this project, I chose *KNNImputer* to handle missing values. Note that the number of neighbors ($n\_neighbors$) is considered as a hyperparameter. I used 5, but other values can be applied.
- **Step 3**: Split *exercise_20_train.csv* into training and validation dataset. Validation dataset will be used for hyperparameter tuning. The split ratio is $8:2$.
- **Step 4**: Dealing with data imbalance for training samples. There are several ways to handle data imbalance (*e.g.* SMOTE [1] and Tomek Links [2]) and the intuitive method is data re-sampling. Since under-sampling may cause overfitting and poor generalization, I chose to use over-sampling technique, which is described as adding more copies to the minority class. Noice that it is fine if the validation dataset is imbalance.
- **Step 5**: Data standardization. Data features for all datasets are normalized with zero mean and unit variance.

	exercise_20_train.csv		*exercise_20_test.csv*
	Training Data	Validation Data	Testing Data
Sample Size	51,092, $\{0: 25546, 1: 25546\}$	8,000, $\{0: 6307, 1: 1693\}$	10,000
Feature Dimension	100	100	100
Target Dimension	1	1	-
Data Types	*float64*(100), *int64*(1)		*float64*(100)

Table 1: Key characteristics of experimental data after cleaning.

Before move to the training stage, key characteristics of experimental data after cleaning is shown in Table 1.

2 Predictive Models

As mentioned in Sec.1, *exercise_20_train.csv* was split into a training dataset and a held-out validation dataset with a split ratio $8 : 2$. To make a fair prediction, class imbalance in training dataset is considered by using a random over-sampling technique. Note that there is no need to make sure class imbalance for the validation dataset. The goal of using validation dataset is to tune hyperparameter of training models.

Notations: In this report, scalars are denoted by lower case italic letters, vectors by lower case bold face letters, and matrices by capital italic letters. Let $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ denote the training data matrix with sample size of n and d feature dimension. Specifically, in our case, $n = 25546 \times 2$ and $d = 100$. $\mathbf{x}_i^T \in \mathbb{R}^d$ represents the i-th training sample, where $i \in [n]$. Let $\mathbf{y} = [y_1, \cdots, y_n]^T \in \{0, 1\}^n$ denote the binary ground-truth labels. Let $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ be a mapping function specified by different classifiers. Our goal is to learn a good model parameter which minimizes the summation of the regularized binary cross-entropy function $BCE(\cdot) : \mathbb{R} \to \mathbb{R}$ (see Table 2). We denote $||\cdot||$ as the Euclidean (ℓ_2) norm.

Problem Formulation: A binary classification problem takes the form that

$$\min_{\mathbf{w}} \quad \frac{1}{n}\sum_{i=1}^{n} BCE(f(\mathbf{x}_i, \mathbf{w}), y_i) + \frac{\epsilon}{2}||\mathbf{w}||^2 \tag{1}$$

where $\epsilon > 0$ is trade-off parameter. Note that the dimension of model parameter $\mathbf{w}$ or W is determined by classifiers and its notation is abused in Eq.(1). For example, in our case, model parameter $\mathbf{w} \in \mathbb{R}^d$ is learned in logistic regression; but in the artificial neural network (ANN, will be introduced shortly), a parameter matrix W is applied.

Next, we study two required predictive models, both of which share the same objective function in Eq.(1) but with different classifiers $f(\cdot)$. In order to make a comparison, I will elaborate both models on three aspects.

2.1 Logistic Regression (Model 1)

- **Classifier**: In logistic regression classifiers, one maps the feature vectors $\mathbf{x}_i$ to the class labels y_i by means of a probability distribution:

$$f(\mathbf{x}_i, \mathbf{w}) = \hat{y}_i = \sigma(\mathbf{w}^T\mathbf{x}_i) \tag{2}$$

  where $\sigma(\cdot)$ is an activate function that normalize the predictive probability into $[0, 1]$.

  - In binary classification, a sigmoid function is applied, that is

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

    Therefore, $\sigma(\mathbf{w}^T\mathbf{x}_i) = p(y_i = 1|\mathbf{x_i}, \mathbf{w})$ and a good $\mathbf{w}$ is obtained by solving a maximum likelihood problem over the training set.
  - In multi-class classification problems, however, a softmax function takes the form

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad \forall i \in [K] \quad \text{and} \quad \mathbf{z} = [z_1, \cdots, z_K] \in \mathbb{R}^K$$

    where i denotes the index of entries of $\mathbf{z}$ and K is the number of classes. Note that in multi-class cases, instead of using the aforementioned BCE loss function, a cross-entropy (CE) loss will be applied.

- **Optimizer**: Gradient descent is used to iteratively minimize the objective function until reaching the stop criteria. Other optimizers (*e.g.* SGD) can also be applied. For simplicity, a plain gradient descent is considered in the first model.
- **Pros and Cons of Model 1**:
  - Pro 1: Simple and easy. Good performance for easy dataset and less hyperparameters.
  - Pro 2: Everything is convex. Both classifier and loss function are convex. Therefore both optimizer convergence and global optima is guaranteed. Theoretic analysis are easy to develop.

	Logistic Regression (LR, Model1)	**Artificial Neural Network (ANN, Model2)**
Linearity	Linear	Non-linear
Model Parameter	$\mathbf{w} \in \mathbb{R}^d$	$W = [\mathbf{w}_1 \in \mathbb{R}^{h1 \times d}, \mathbf{w}_2 \in \mathbb{R}^{h2 \times h1}, \mathbf{w}_3 \in \mathbb{R}^{1 \times h2}]$
Loss Function	Binary Cross-Entropy, $BCE(\hat{y}_i, y_i) = -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$	
Optimizer	Gradient Descent	Adam [3]
Convexity of the Objective Function	Convex	Non-convex (due to non-linearity of activation functions)
Global Optima	Yes	No
Number of hyperparameters	Less	More
Explainability	Easy	Hard

Table 2: A Brief Summary of Comparison Between Two Classifiers

- Pro 3: Explainability. Due to strong statistic background, logistic regression model is explainable.
- Con 1: Linear model. Bad performance for dataset with high dimension or in linear non-separable case.
- Con 2: Difficult to handle multi-class classification problem. Although, there are several ways to adapt logistic regression (*e.g.* one-vs-all, replace sigmoid to softmax) to handle multiple classes, in contrast to deep learning models, neural networks perform better.

2.2 Artificial Neural Network (Model 2)

- **Classifier**: A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear. In this work, I simply used 2 hidden layers and each contains 10 neurons. Both of these two layers follow with ReLU activation functions. Similar to the logistic regression model, in binary classification (this project), sigmoid is applied in the output layer. As for multi-class classification, softmax is selected. An overview of structure of the neural network is shown in Figure 1.

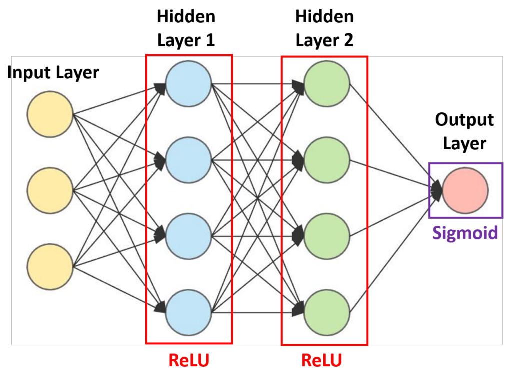


Figure 1: A schematic overview of ANN used in this project.

- **Optimizer**: Adam [3] is an optimization algorithm which is proposed in 2015 and commonly used in deep learning. It is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. In contrast to other popular optimizer, Adam converges faster and is less expensive.
- **Pros and Cons of Model 2**:
  - Pro 1: Better performance than conventional statistic models (*e.g.* logistic regression, SVM, *etc*) due to non-linear classifier with high dimensional model parameters.
  - Pro 2: Easy to generalize with multi-class-classification cases (just simply replace the last layer activation function to softmax and loss function to cross-entropy).

- Con 1: A complicate model in contrast to logistic regression. NN contains more hyperparameters, such as number of hidden layers, number of neurons in each layer, and selection of activation functions. Therefore, the model is easy overfitted. One needs to pay attention to this when training.
- Con 2: Non-convexity. Due to the non-linearity of activation functions, the classifier is non-convex and hence the objective function is non-convex. Therefore, a global optimum is not guaranteed.
- NN is hard to explain. The explainability of a NN is related to hyperparameter tuning. Researches in causality and graphic inference are cutting-edge studies to explain effectiveness and efficiency of neural networks.

3 Experimental Settings

All experiments are tested on a server of Intel Xeon(R) 2.40GHZ E5-2680 with 251GB of RAM. The code is written in Python 3.8. Codes are attached in a separated folder.

3.1 Hyperparameter Tuning

- **Logistic Regression**. In order to achieve the best performance, instead of using existing well-packed models (*e.g.* `sklearn`, `statsmodels`, *etc*), the logistic regression model is implemented from scratch. This model contains four hyperparameters: number of iterations (m_1), learning rate (ρ_1), tolerance (τ) and regularization trade-off (ϵ_1).
- **ANN**. The second predictive model is consist of 2 hidden layers where each contains 10 neurons. `Pytorch` [4] is selected for implementation. This ANN model contains 3 hyperparameters: learning rate (ρ_2), number of epochs (m_2), and weight decay (ϵ_2).

In order to tune hyperparameters for both models, training losses and validation losses are plotted in Figure 2. ϵ_1 and ϵ_2 are chosen from $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$ and learning rate ρ_1 and ρ_2 are chosen from $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$. The tolerance standard τ is set to be 0.0001 as given in `sklearn.linear_model.LogisticRegression`. Notice that, in order to avoid model overfitting, m_1 and m_2 are selected by using early stop. For example, in the right figure of Figure 2, validation loss first decreases and then increases, which indicate model is over-fitted with $m_2 = 30000$. Therefore a good m_2 for the second model is 7000 (*i.e.* stop at the red arrow).

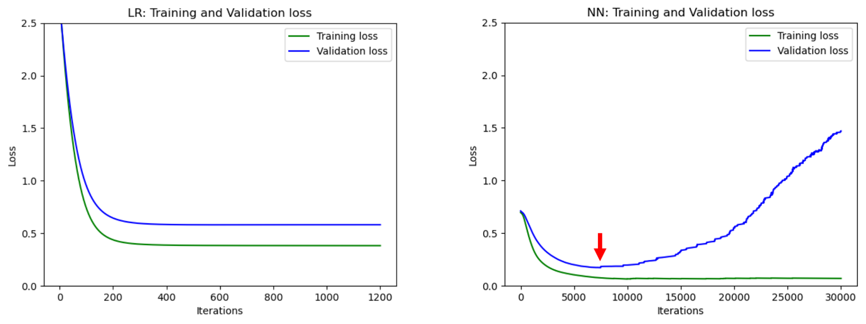


Figure 2: Training verse validation losses. (Left) Logistic Regression; (Right) Neural Network.

3.2 Reproduciblility

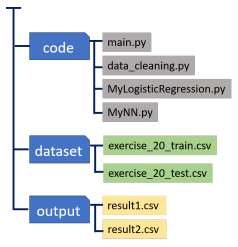
Hyperparameters. To reproduce results reported in this work, tuned parameters are given. (1) LR: $m_1 = 1200$, $\rho_1 = 0.1$, $\tau = 0.0001$, and $\epsilon_1 = 0.00001$; (2) NN: $\rho_2 = 0.0001$, $m_2 = 7000$, and $\epsilon_2 = 0.00001$.

Requirement. Python packages and versions used for development are listed in Table 3a.

File Directory. See Figure 3b.

Packages	Version
`torch`	1.6.0
`sklearn`	0.23.1
`numpy`	1.18.5
`pandas`	1.0.5
`imblearn`	0.7.0
`matplotlib`	3.2.2

(a)



(b)

Figure 3: (a) Required Installed Packages and (b) File Directory

4 Results and Discussion

Evaluation Metrics. The goal of using evaluation metrics is to assess the performance of machine learning models. Since there are so many of them, for simplicity I used two of them to illustrate results but others may also apply.

- **Accuracy**. It measures how many observations, both positive and negative, were correctly classified. Note that accuracy is meaningless if training is imbalanced. To this end, in the data cleaning procedure, training dataset is balanced before experiments.
- **ROC Curve**. It visualizes the trade-off between true positive rate (TPR) and false positive rate (FPR). Basically, for every threshold, TPR and FPR are calculated and then plotted. The higher TPR and the lower FPR is for each threshold the better and so classifiers that have curves that are more top-left side are better.

	Validation Accuracy
Logistic Regression (Model 1)	72.47±0.05%
Neural Network (Model 2)	94.55±0.34%

Table 3: Validation Accuracy. All experiments are repeated 10 times. Results are represented using mean followed by standard deviation.

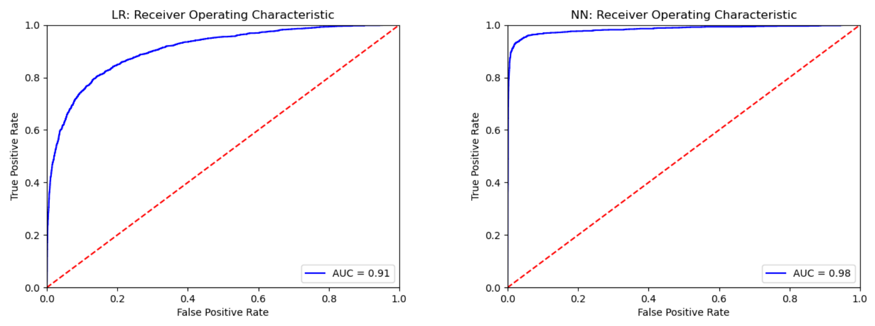


Figure 4: ROC curves on validation dataset. (Left) Logistic Regression; (Right) Neural Network.

Performance based on held-out validation dataset are given in Table 3 and Figure 4. Both experiments were repeated 10 times using the same setting. Table 3 provides the final predicted validation accuracy. Results are shown using mean followed by standard deviation. It indicates Model 2 increases predictive accuracy by 22% than using Model 1. Although the validation dataset is imbalance (6307:1693, for details see Table 1) regarding to binary labels, training dataset is balanced (25546:25546) and thus the reported results are reliable. Besides, according to ROC curves shown in Figure 4, Model 2 has higher AUC value than that of the Model 1. ROC curves more top-left in Model 2 indicates higher TPR and lower FPR than those in Model 1. Therefore, NN (Model 2) performs better than Logistic Regression (Model 1).

Question: How would explain to a business partner the concept that one model is better than the other?
There are three key points to determine a good model:

1. Performance. This is always the top priority.
   (a) Accuracy. In classification task, accuracy is always the most important thing. In this project, the second model increases 22% of prediction accuracy than the first one. However, higher accuracy is not enough. Because this may results from other issues, such as class imbalance or high false positive rate. We therefore need to check TPR and FPR.
   (b) Confidence score. Confidence score is defined as the maximum probability over all classes. It measure how confident of the predicted class. For example, both $[0.6, 0.4]$ and $[0.9, 0.1]$ will be rounded as $[1, 0]$. But the later result is more confident to be used to illustrate the prediction. In this project, confidence scores in model 2 are higher by using model 1.
   (c) TPR and FPR. There are many ways to check true positive rate and false positive rate. A reasonable good model is required to have a high TPR and a low FPR and the more the better. ROC curve is one of metrics to evaluate TPR and FPR. In the result, in contrast to model 1, model 2 has higher TPR and lower FPR, as shown in Figure 4 ROC curve on the right is more top-left.
   (d) Losses. Figure 2 shows that although model 1 demonstrate faster convergence, both training loss and validation loss drop more dramatically in model 2.
2. Ability of generalization. This is also very important to determine a model is good or not.
   (a) Binary to multiple classes and complicate dataset. Logistic regression is well known for binary classification. This model is powerful when data is simple and easy to be separated. However, when training data becomes complicated or with multiple class labels, classifiers learned with high dimensional parameters (such as kernel-based SVM and neural networks) will achieve better results.
   (b) Domain shift. An important assumption is that data domains in training and testing phases are the same. However, this is not always true. For example, training data is collected from one city, but testing data may collected from another city in which the population distribution is very different from the first city. Models trained in one data domain may not give a fair and good prediction in another domain. This problem has drawn a lot of attentions by machine learning researcher. Due to space limit, if you are interest, it will be my pleasure to discuss with you regarding this.
3. Time and space complexity. There is no free lunch. We discussed effectiveness of these two models in the first two key points, but learning efficiency is also important.
   (a) Time complexity. In research, when a new algorithm is proposed, time complexity is always required to be discussed. In this work, model 1 applied gradient descent optimizer, but model 2 used Adam. As a classic and traditional optimizer, gradient descent is easy to understand and implement. However, the drawback of it is that when training data is large, gradient descent becomes very slow. This is because, at each iteration, the optimizer will take first-order gradient with respect to model parameters using the entire dataset. Fortunately, Adam is introduced as a simple and computationally efficient algorithm for gradient-based optimization of stochastic objective function [3]. It simplely combines the advantages of AdaGrad and RMSProp and it is aimed towards machine learning problems with large datasets and/or high dimensional parameter spaces.
   (b) Space complexity. Due to high dimensional model parameter and high inference cost in model 2, momdel 1 has significantly less spatial complexity. But as time goes on, space complexity becomes less important and this problem can be solved by the improvement of computer hardwares.

References

[1] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority over-Sampling Technique. 16, 1 (2002), 321–357.

[2] Tusneem Elhassan and Aljurf M. 2016. Classification of Imbalance Data using Tomek Link(T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method:. *Journal of Informatics and Data Mining* 1 (01 2016). https://doi.org/10.21767/2472-1956.100011

[3] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035.