CS 280 Programming Assignment 2A Neural Network & SVM for Multiclass Classification

Helbert Paat

University of the Philippines
hapaat@up.edu.ph

November 17, 2020

I. Neural Networks

Neural networks refer to broad type of non-linear models built with layers. The model can capture high degree of non-linearity in the data. A fully-connected neural network is developed by stacking layers. Let r be the number of layers. Let $W^{[1]}, ..., W^{[r]}, b^{[1]}, ..., b^{[r]}$ be the weight matrices and biases of all the layers. Then a multi-layer neural network can be written as

$$\begin{aligned}
&a^{[1]} = \sigma(W^{[1]}x + b^{[1]}) \\
&a^{[2]} = \sigma(W^{[2]}a^{[1]} + b^{[2]}) \\
&... \\
&a^{[r-1]} = \sigma(W^{[r-1]}a^{[r-2]} + b^{[r-1]}) \\
&h_\theta(x) = W^{[r]}a^{[r-1]} + b^{[r]})
\end{aligned}$$

where σ one-dimensional non-linear function that maps $\mathbb{R}$ to $\mathbb{R}$ often referred to as an activation function. A usual choice of the activation function is the sigmoid activation function. The sigmoid activation function is defined to be the following:

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

In this paper, a neural network with two hidden layers is developed using sigmoid activation for the two hidden layers. Sigmoid activation function is also used for the output layer.

II. Support Vector Machines

Support Vector Machines is used for classification problems. The algorithm finds a hyperplane decision boundary that splits the examples into two classes. The algorithm may allow some observations to be misclassified to achieve greater robustness to individual observations.

The solution has the form $f(x) = \beta_0 + \Sigma_{i \in S} \alpha_i K(x, x_i)$ where β_0 and $\alpha_i' s$ are the parameters and $K(x, x_i)$ is the kernel function. A kernel function measures the similarity of 2 observations. Through the use of kernels, non-linearity is introduced in the support vector classifiers.

III. Methods

The dataset used in this experiment was partitioned into training, validation, and test set. The dataset contains 3486 instances with 354 attributes each. Every instance is categorized into one of the 8 labels.

Data set	# of instances	% of dataset
Training Set	2436	70%
Validation Set	700	20%
Test Set	350	10%

Table 1: *Partition of the Dataset into Training Set, Validation Set, and Test Set*

A. Neural Network for Multiclass Classification

a. Compute the initial parameters for the neural network. Weight matrices are initialized with a random normal distribution centered on zero and with standard deviation 1. Bias vectors are initialized to be zero vectors.

b. Implement the forward layer for the two-hidden layer neural network. For the first and second hidden layers, sigmoid function is used as the activation function. For the output layer, sigmoid function is also used.

c. The loss function used is the mean-squared error loss function defined to be the following:

$$R(\theta) = \frac{1}{N}\Sigma_{k=1}^{K}\Sigma_{i=1}^{N}(y_{ik} - f_k(x_i))^2$$

where $f_k(x) = h_\theta(x)$, K is the number of classes, N is the number of training examples, and y_{ik} is an element of the one-hot encoding of the labels for example i where $y_{ik} = 1$ if the label of training example i is k and 0 otherwise.

d. The corresponding classifier is $G(x) = argmax_k f_k(x)$

e. To minimize $R(\theta)$, mini-batch stochastic gradient descent was used to determine the update in the weights and bias. The following described this algorithm:

```
Mini-batch stochastic gradient descent
1: Hyperparameters: learning rate $\alpha$, batch size B, number of iteration $n_{iter}$
2: Initialize $\theta$ randomly.
3: for i = 1 to $n_{iter}$ do
4:     Sample B examples $j_1, ..., j_B$ (without replacement) uniformly from $\{1, ..., n\}$, and update $\theta$ by
```

$$\theta := \theta - \frac{\alpha}{B}\Sigma_{k=1}^{B} \nabla_\theta J^{j_k}(\theta)$$

f. A method of regularization called weight decay is also applied in order to avoid overfitting. A penalty $\lambda J(\theta)$ is added to the error function where

$$J(\theta) = \sum_i W_i^2 + \sum_i b_i^2$$

where $\lambda \geq 0$ is a tuning parameter.

B. SVM for Multiclass Classification

Support Vector Machines were designed for binary classification. The algorithm is not natively used for classification tasks with more than two classes.

To use SVM for multiclassification problems, the multi-class classification dataset can be split into multiple binary classification datasets. A binary classification model is then fitted on each. There are several heuristic methods to perform this. In this paper, the heuristic method used is the One-vs-One (OvO) method, which is the suggested approach for SVM and other kernel-based algorithms. The OvO approach splits the dataset into one dataset for each class versus every other class.

C. Dealing with the Highly Imbalanced Dataset

Note that the dataset is highly imbalanced. Table 2 shows the number of instances per label. The plot of the training data can be seen in Figure 1.

Label	Number of Instances
1	1154
2	170
3	22
4	341
5	200
6	206
7	36
8	307

Table 2: *Number of Instances per Label in the Training Set*

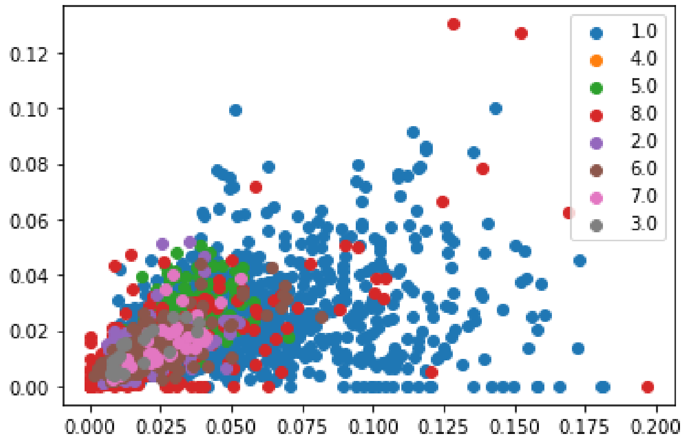

Figure 1: *Plot of Dataset*

SMOTE
Algorithms such as the Support Vector Machine algorithm works well for balanced classification. However, it is not effective on imbalanced datasets. When a machine learning technique like SVM is used for imbalanced datasets, it will not effectively learn the decision boundary because it tends to ignore the classes

with few instances. This results in poor performance.

To solve this problem, Synthetic Minority Oversampling Technique (SMOTE) can be used. It is a type of data augmentation where examples in the minority class are duplicated. However, they don't add new information to the model but only oversample the minority class. SMOTE synthesizes new samples by drawing a new sample at a point along the line between the examples that are close in the feature space. Figure 2 shows the balanced dataset after applying SMOTE with 1154 instances per class.

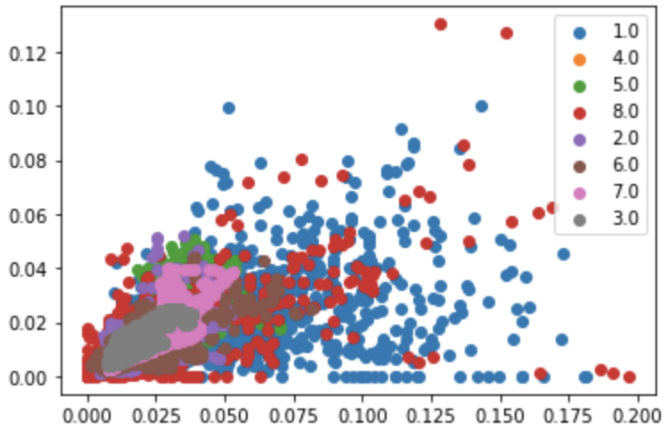

Figure 2: *Plot of Balanced Dataset after Applying SMOTE*

D. Measuring Performance

Accuracy is used to measure the performance of the classifier. Intuitively, accuracy is the number of test instances correctly classified over the total number of test instances.

IV. Results

A. Neural Network Hyperparameter Tuning

In this experiment, there are four hyperparameters that must be determined: the learning rate, the number of units in the first hidden layer, the number of units in the second hidden layer, and the regularization rate.

Learning Rate
After training 30 epochs, the validation set is used to determine the accuracy using the updated parameters. To determine the optimal learning rate, 200 units in the first hidden layer, 200 units in the second hidden layer, and no regularization was used. Table 3 shows the results for testing different values of the learning rate. A learning rate of 0.50 yields the best performance.

Learning Rate	Accuracy
0.01	57.71
0.05	65.43
0.50	69.57
0.75	69.43
0.90	68.29
1.50	69.14
2.50	53.14

Table 3: *Testing different values of the learning rate and the resulting model's prediction accuracy*

Number of Nodes in First Hidden Layer
To choose the optimal number of nodes in the first hidden layer, learning rate of 0.50, a second hidden layer with 200 nodes, and no regularization rate is used. Table 4 shows the results. A first hidden layer with 50 hidden units yields the best result.

# of Nodes in 1st Hidden Layer	Accuracy
50	74.71
100	62.14
200	69.57
250	67.57
300	55.57

Table 4: *Testing different values of the number of nodes in the first hidden layer and the resulting model's prediction accuracy*

Number of Nodes in Second Hidden Layer
To choose the optimal number of nodes in the second hidden layer, learning rate of 0.50, a first hidden layer with 50 nodes, and no regularization rate is used. Table 5 shows the results. A second hidden layer with 50 nodes results in the best accuracy.

Regularization Rate
To choose the optimal regularization rate, learning rate of 0.50, a first hidden layer with 50 nodes, and a second layer with 50 nodes are used. Table 6 shows the results. Very small values of the regularization rate resulted in the best accuracy. For this reason, no regularization rate was used.

The optimal hyperparameters to use are presented in Table 7.

# of Nodes in 2nd Hidden Layer	Accuracy
50	77.57
75	68.57
100	75.71
200	74.71
250	1.43
300	6.57

Table 5: *Testing different values of the number of nodes in the second hidden layer and the resulting model's prediction accuracy*

Regularization Rate	Accuracy
0.000001	77.71
0.00001	77.71
0.0001	77.71
0.001	77.71
0.01	77.57
0.10	75.14

Table 6: *Testing different values of the regularization rate and the model's corresponding prediction accuracy*

Hyperparameter	Value
Learning Rate	0.50
# of nodes in 1st hidden layer	50
# of nodes in 2nd hidden layer	50
Regularization Rate	0

Table 7: *Optimal Hyperparameters for the Neural Network Model*

Plot of Training and Validation Errors
Using the best values for each hyperparameter for the Neural Network model, training and validation errors were plotted using 1000 epochs. Refer to Figure 3 for details.

B. SVM Hyperparameter Tuning

To determine what kernel must be used, several kernel types were tested. The kernel that yields the best accuracy is chosen. Table 8 shows the result.

It is observed that the SVM with polynomial kernel of degree 13 yields the best accuracy. Using this kernel, a testing was also conducted to determine the best value of C. Recall that the value of C can be interpreted as tolerance level to the violations of some observations to the margin. Table 9 shows that C=1 results in the best accuracy.

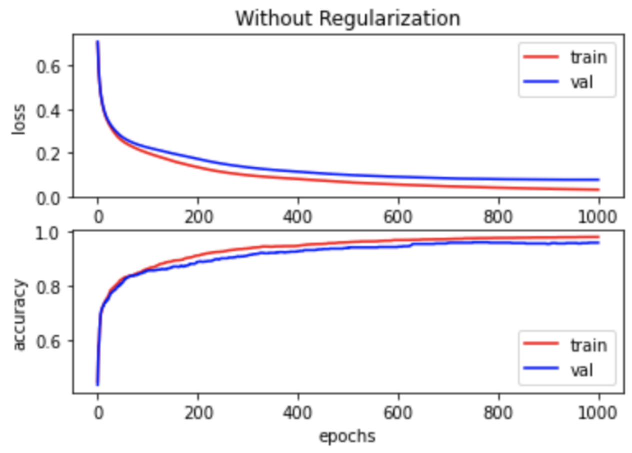

Figure 3: *Plot of Training and Validation Errors using the Best Hyperparameter Values for the Two-Hidden Layer Neural Network Model*

Kernel	Accuracy
linear	96.29
polynomial (degree 5)	95.43
polynomial (degree 9)	96.57
polynomial (degree 11)	96.86
polynomial (degree 13)	97
polynomial (degree 15)	96.43
sigmoid	84.71
radial	93.43

Table 8: *Different Kernels for SVM and their corresponding accuracy*

C	Accuracy
0.25	94.57
0.5	96.14
1	97
2	96.86
3	96.86

Table 9: *Different values of C for the SVM classifier and their corresponding accuracy*

C. Comparing the Classifiers in Terms of Accuracies and Running Times on Unseen Data

We compare the SVM and Neural Network in terms of accuracy and running times on the unseen test data. Note that the Neural Network used the best value of the hyperparameters using 1000 epochs.

Classifier	Accuracy	Running Time
Neural network	94.29	3373.79
SVM (imbalanced dataset)	93.71	4.0578
SVM (balanced dataset)	96	18.1985

Table 10: *Comparison of Support Vector Machine and Neural Networks in terms of Accuracy and Running Time*

V. Conclusions and Recommendation

In this paper, it has been shown that the two-hidden layer neural network that uses sigmoid function in its hidden layers and output layer, and SVM models with polynomial kernel of degree 13 performed well in classification. The neural network model achieves better accuracy than the SVM model using the imbalanced dataset. However, the drawback with the neural network model is the time it takes to train the model in order to achieve better performance. The SVM model gave an improved accuracy using the balanced dataset.

For future studies, neural network can be trained using a more appropriate activation function for the output layer such as the softmax function and more appropriate loss function for classification such as the cross-entropy loss.