CS6375 Machine Learning

Project 1

Cliff Eddings

University of Texas at Dallas

Summer 2019

Keywords and Concepts

1. Neural Network – Artificial Neural Networks are built out of densely interconnected set of simple units where each unit takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output (which may become the input to many other units) (Mitchell, 1997).
2. Data preprocessing – converts the raw data normalizing it into a clean data set. Mostly, data gathering methods are lightly controlled resulting in outliers, impossible data combinations, and missing values etc. … The quality, reliability and availability are some of the factors that may lead to a successful data interpretation by a neural network (Nawi, Atomi, & Rehman, 2013).
3. Layer – cluster of nodes in the same level in a neural network
4. Node – threshold unit in a neural network, a neuron
5. Activation Function – function used in the algorithm to weigh and evaluate the input data, i.e. ReLU, Sigmoid, etc. The purpose of the activation function is to introduce non-linearity into the output of a neuron (Tiwari, n.d.).
6. Regularizer – adding the regularization component will drive the values of the weight matrix down. This will effectively decorrelate the neural network… which in turn decreases the effect of the activation function. Therefore, a less complex function will be fit to the data, effectively reducing overfitting (Peixeiro, n.d.).
7. Drop Out – means the neural network cannot rely on any input node … the neural network will be reluctant to give high weights to certain features, because they might disappear. Consequently, the weights are spread across all features … effectively shrinks the model and regularizes it (Peixeiro, n.d.).
8. SoftMax – at the last step of a neural network the computed real values are converted into probabilities.
9. Optimizer – an algorithm used to optimize the neural network. Types of the algorithm include Adam, Stochastic Gradient Descent, etc.
10. Epoch – number of iterations to train the model.
11. Batch Size – number of samples evaluated before updating the model parameters.

Design Formula

The goal of the project is to design a neural network model which will train on 10% of a set of data then tested on the full set to achieve an accuracy between 90-95%. The provided dataset gives 569 examples with 30 categories and 2 labels (0, 1). In order to design the model, there were several aspects of the model I experimented and tuned. The aspects are preprocessing the data, number of nodes in a layer, the activation function, regularizers, dropout percentage, the optimizer function, epochs, and batch size.

The first step in the process was run the example code as given to get a base accuracy and loss value. The values for the template program was 0.5188 loss and 75.40% accuracy when training on the x and y training files then tested on the x and y test files.

The next step was to determine the contribution of each individual aspect of the model by changing the values and running the program. The testing from this step provided the following observations (reference Table 1 page 6; Figure 1 page 10):

- Preprocessing the data gave the largest improvement in accuracy – 93.5%
- Adding nodes to the layers improved the model better than adding layers – 81.55% average compared to 84.89%
- Using the SGD (stochastic gradient descent) optimizer is better for the beginning model than Adam – 81.72% compared to 75.45%.
- Tanh an eLU were better activation functions on the beginning model than ReLU. Both activation functions gave a 77% accuracy compared to 75%.
- Changing the values and types of the regularizers had little to no effect on the accuracy of the model.

These observations were important in developing and tuning the developed model.

Testing and Evaluation

After the initial testing and evaluation, the first version of the designed model was implemented for testing, evaluation, and refinement. The initial design model:

- Using the scaler.fit_transform() function for preprocessing
- 15 nodes in 3 dense layers
- ReLU as the activation function
- L2 regularizer set at 0.001
- Dropout percentage set at 0.2
- Adam optimizer
- 1,000 epochs
- Batch size of 32.

Training and testing the neural network on the x and y test files with minor adjustments achieved an average accuracy of 92.25% the best variation, version 2.6, produced an accuracy of 93.32% (reference Table 2, page 8, and Figure 2, page 11).

The next step involved training and testing the model on different seeds of the x and y test datasets. The first problem encountered was the scaler.fit_transform function did not provide adequate accuracy on certain seeds and the preprocessing was changed to divide the values by 100. After several rounds of evaluation, the best version achieved an average of 92.27% across 8 different dataset seeds and the provided training files (reference Table 3, page 9, and Figure 3, page 12).

The Final Neural Network Model

The neural network, designed for this project is described as follows:

- Data is preprocessed by dividing the dataset values by 100
- The Neural Network is 3 dense layers consisting of 25 nodes per layer and a top layer of 2 nodes for the output.
- The activation function is ReLU
- The third level uses a L1 and L2 regularizers from the keras module set at 0.015
- Drop out percentage is set to 0.4
- The output is evaluated with the SoftMax function.
- The optimizer algorithm is “Adam”
- The model is trained for 1,000 epochs
- Batch size is 32.

Conclusions

This exercise revealed some intriguing lessons in the development of neural networks. One being adding nodes and layers can be counterproductive when trying to improve accuracy. Another observation is the importance of preprocessing the data and preprocessing it in a way that performs well on different sets of training data. Also, the keras module in TensorFlow simplifies much of the process of designing the neural network.

References

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill.

Nawi, N. M., Atomi, W. H., & Rehman, M. Z. (2013). The Effect of Data Pre-Processing on Optimized Training of Artificial Neural Networks. *Procedia Technology 11*, 32-39.

Peixeiro, M. (n.d.). *How to Improve a Neural Network With Regularization*. Retrieved from Towards Data Science: https://towardsdatascience.com/how-to-improve-a-neural-network-with-regularization-8a18ecda9fe3

Tiwari, S. (n.d.). *Activation Functions in Neural Networks*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/activation-functions-neural-networks/

Tables

Table 1

Version 1 Tracking – Changes in each attribute to show the effects on accuracy and loss, reference for Figure 1.

Version	Preprocess type	Layers	Nodes per layer	activation function	regulizer setting	dropout percentage	optimizer	Epochs	batch size
1.0	none	3	5	relu	l2 = 0.001	0.2	Adam	500	32
1.1	/100								
1.2	scaler								
1.3		2							
1.4		4							
1.5		5							
1.6				sigmoid					
1.7				hard_sigmd					
1.8				tanh					
1.9				elu					
1.10			15						
1.11			20						
1.12			25						
1.13					l1=0.001				
1.14					l2=0.01				
1.15					l1=0.01				
1.16					l1&l2 0.001				
1.17					l1&l2 0.01				
1.18						.05			
1.19						.1			
1.20						.4			
1.21						.6			
1.22							sgd		
1.23							adamax		
1.24							Nadam		
1.25								1000	
1.26								1500	
1.27									10
1.28									20
1.29									40
1.30									50
2.0	scaler		15					1000	
2.1	scaler	4	15					1000	
2.2	scaler	3	25					1000	
2.3	scaler	3	25				sgd	1000	
2.4	scaler	3	25			.1	Adam	1000	

2.5	scaler	3	25		l2=.01	.1	Adam	1000	
2.6	scaler	3	25		l1&l2=.01	.1	Adam	1000	
2.7	scaler	3	25		l1&l2=.01	.1	Adam	1000	40
2.8	scaler	3	25		l1&l2=.01	.1	Adam	1000	20
2.9	scaler	3	25	tanh	l1&l2=.01	.1	Adam	1000	32
2.10	scaler	3	25	elu	l1&l2=.01	.1	Adam	1000	32
2.11	scaler	3	25	hard_sig	l1&l2=.01	.1	Adam	1000	32
2.12	scaler	3	25	relu	l1&l2=.01	.1	Adam	1200	32
2.6a	/100	3	25	relu	l1&l2=.001	.1	Adam	1000	32
2.6b					.01				
2.6c						.4			
2.6d					.0095				
2.6e		4				.075			
2.7a	/100	3	25	relu	l2 = 0.01	0.4	Adam	1000	32
2.7b					l1=.01 l2=.001				
2.7c					l1 & l2 = .015				

Table 2

Version 2 Evaluation. Training and testing the neural network on x_traing.csv and y_train.csv

	Loss	Accuracy
Version 1.0	0.5188	75.40%
Version 2.0	0.3486	91.74%
Version 2.1	0.7726	89.46%
Version 2.2	0.4178	92.44%
Version 2.3	0.2408	91.74%
Version 2.4	0.3980	92.79%
Version 2.5	0.3282	92.97%
Version 2.6	0.2748	93.32%
Version 2.7	0.2710	92.44%
Version 2.8	0.3034	92.62%
Version 2.9	0.2815	91.92%
Version 2.10	0.2688	92.79%
Version 2.12	0.3043	92.79%

Table 3.

Version 2 Fine Tuning – Training and testing the neural network on different seeds of x_test.csv and y_test.csv

	Version 2.6b		Version 2.7a		Version 2.7c	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
x_train	0.2468	92.97%	0.2578	92.44%	0.2962	92.62%
Seed 2	0.2903	91.21%	0.3363	90.69%	0.2956	90.33%
Seed 3	0.2952	92.44%	0.3035	92.62%	0.2798	92.79%
Seed 4	0.2789	91.92%	0.2867	92.09%	0.2819	92.27%
Seed 5	0.3662	90.86%	0.3815	91.04%	0.3495	91.39%
Seed 6	0.2936	92.62%	0.3019	92.44%	0.2962	92.62%
Seed 7	0.2468	92.97%	0.2578	92.44%	0.2576	92.62%
Seed 8	0.2838	91.56%	0.2954	91.04%	0.2656	92.09%
Seed 9	0.2492	91.39%	0.2538	93.32%	0.2356	93.67%
Average	**0.2834**	**91.99%**	**0.2972**	**92.01%**	**0.2842**	**92.27%**
Range	**0.1194**	**2.11%**	**0.1277**	**2.63%**	**0.1139**	**3.34%**
Sdev	**0.0350**	**0.74%**	**0.0392**	**0.83%**	**0.0300**	**0.89%**

Figures

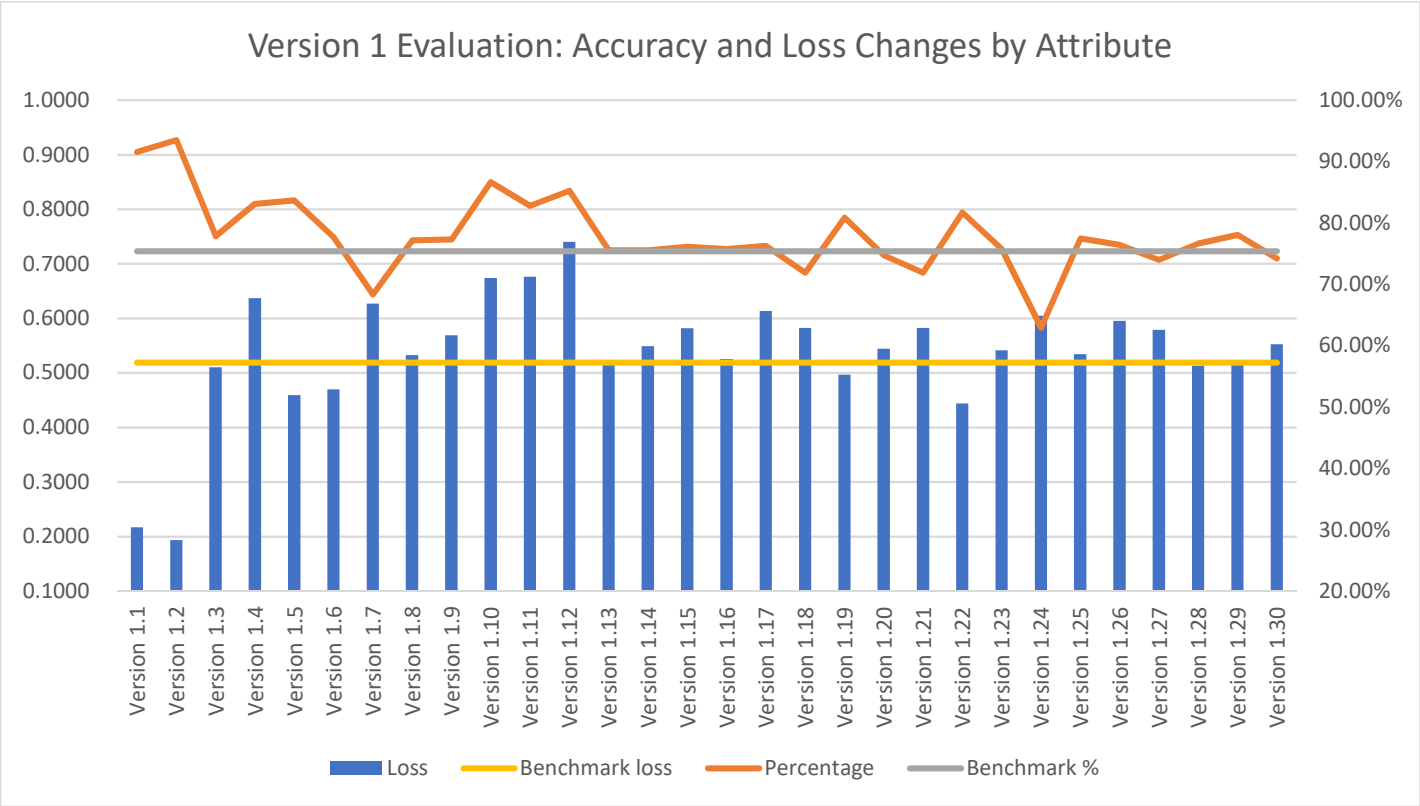

Figure 1. Shows the changes in accuracy and loss achieved by changing one attribute at a time to determine the overall impact of each attribute. Reference Table 1 for version description.

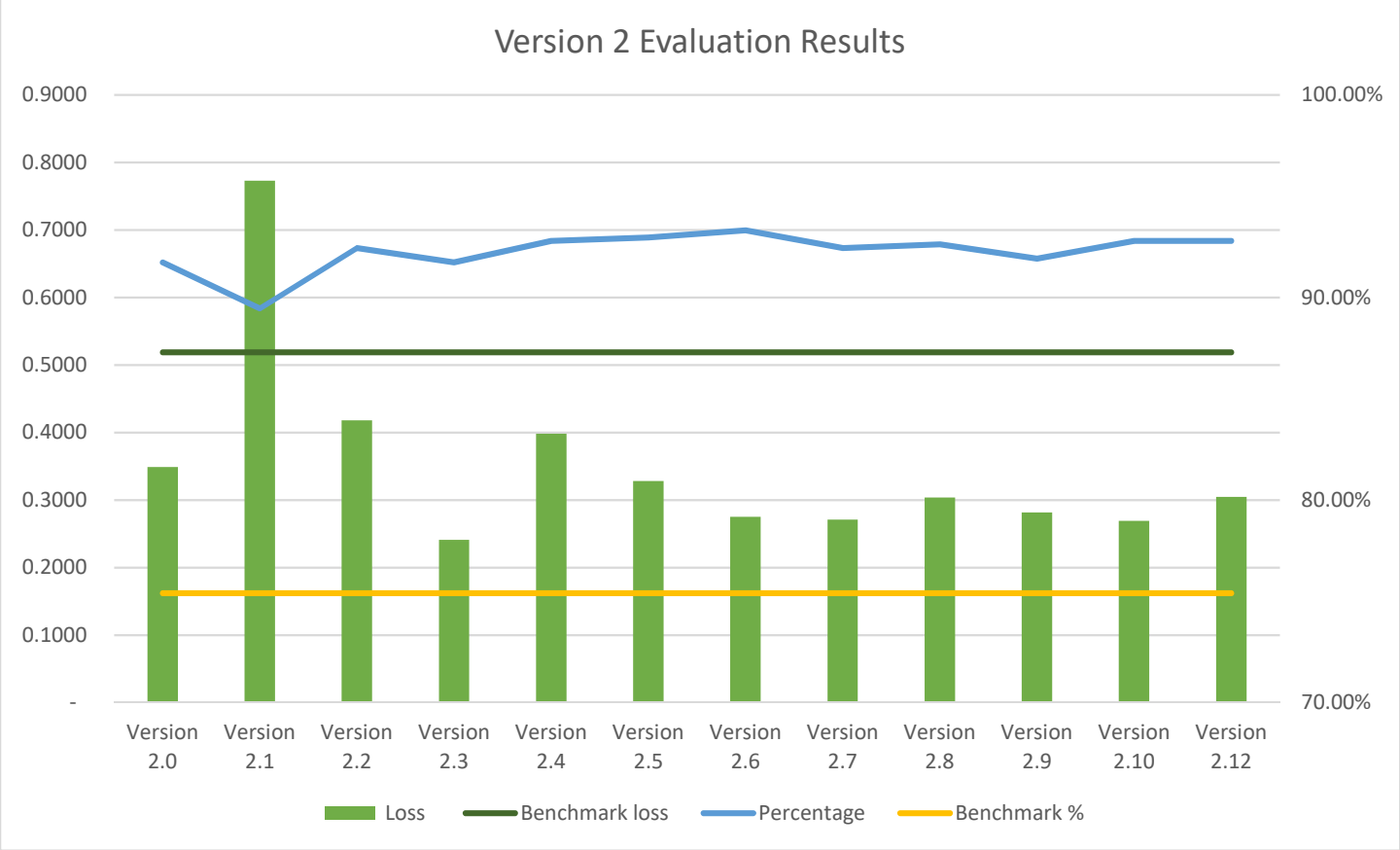

Figure 2: Shows changes in accuracy and loss when tuning the designed model. Reference Table 1 for version descriptions.

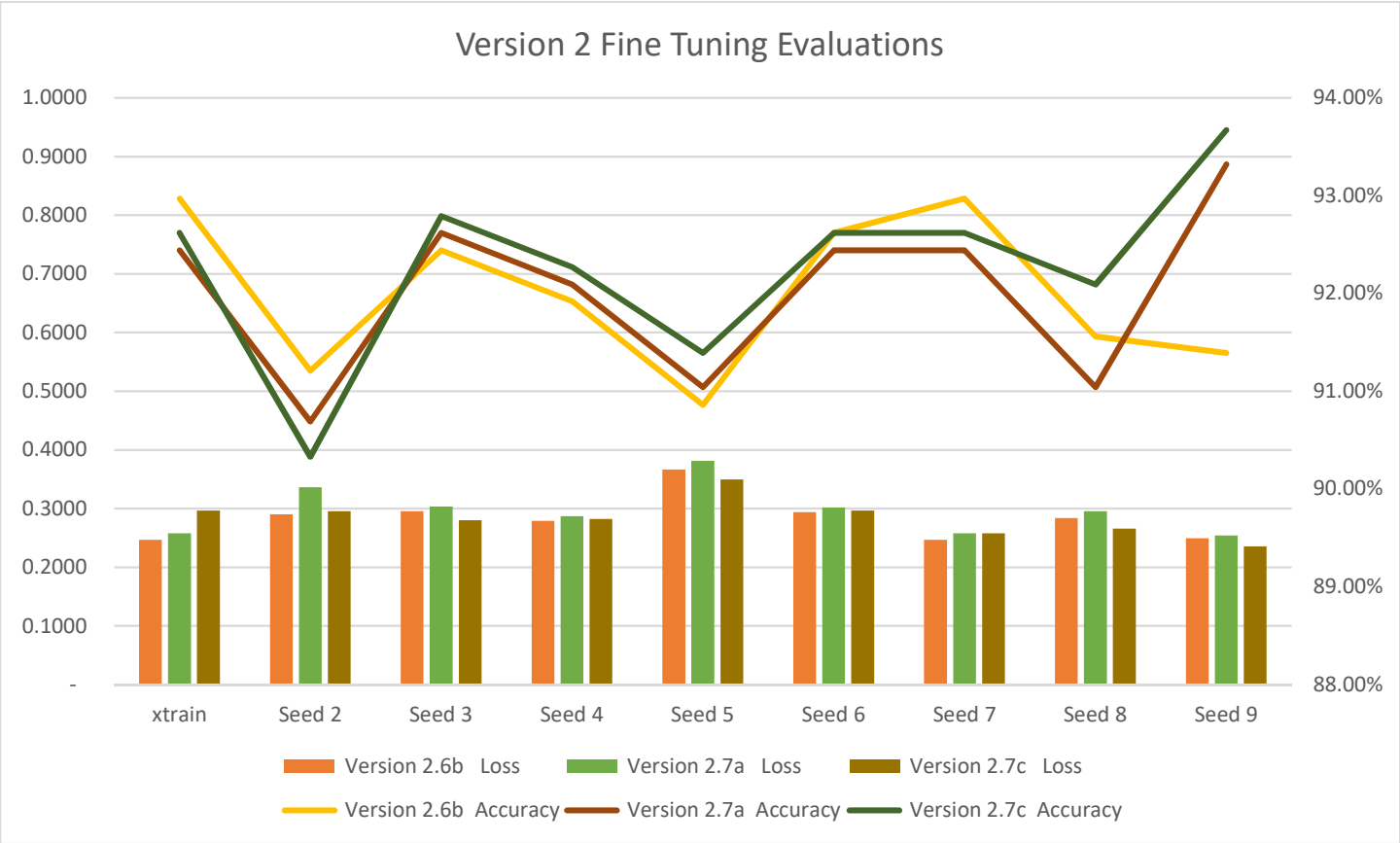

Figure 3. Evaluation results for fine tuning version 2. Results are for the top 3 versions. Reference Table 1 for version descriptions.