FORDHAM UNIVERSITY
THE JESUIT UNIVERSITY OF NEW YORK

CIS 5800 Machine Learning Final Project

Implementations and Experiments

on Neural Network and Support Vector Machine

Jiayin Hu

Fordham University

2019 Fall

Introduction

Neural Network and Support Vector Machine (SVM) are two widely used machine learning methods to solve the classification problems. They both can be applied to linear and non-linear classification.

In this project, I implemented and expanded a 3-layer neural network. Meanwhile, I used both the neural network and support vector machine from the pre-built machine learning library in Python, Scikit-learn, to experiment on a binary classification problem.

The data I used is "Adult" dataset, which provides some personal features to predict whether a person can earn over $50K per year. The best result for neural network and SVM gives 82.00% and 82.54% accuracy in prediction respectively.

Methods

1. Neural Network

I implemented a three layer neural network with input layer, hidden layer, and output layer. The `NeuralNetwork` class achieves forward propagation to calculate the output of each layer and the classification result and back propagation to update the weights connecting neighboring layer. The structures of weights and outputs in each layer are all 1d-arrays or nd-arrays in Numpy.

When initiating the neural network, the constructor accepts the quantity of nodes in each layer and specifies the activation function which will be applied. The number of nodes in the input layer should equal the number of features feed in the neural network. The number of hidden layer nodes is flexible to change. Although the "Adult" dataset has binary target, `NeuralNetwork` can handle multi-classification problems, which include more than 2 results in target. Therefore, for the number of nodes in the output layer, 1 node is easier to deal with a binary classification. When 3 or more classes are involved, the number should be equal to the quantity of classes. With these preset parameters, the weights matrices connecting input layer to hidden layer, hidden layer to output layer are generated with random numbers. For the convenience of comparison, a random seed can be assigned to help generate the same initial weight matrices. Activation functions are built inside with their corresponding derivatives.

Activation Function	**Formula** $f(x)$	**Derivative** $f'(x)$
Sigmoid (default)	$\frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$
Hyperbolic Tangent	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$1-f(x)^2$
Rectify Linear	$\max(0,x)$	$1 \times I(f(x)>0)$

`NeuralNetwork` are bundled with several methods to complete the task of training model and predicting new data.

`forward_feed` is the fundamental step of forward propagation. It computes the output of next layer based on the known data from previous layer and the current weights.

`predict` proceeds all the data samples through a completed forward propagation and get the result in output layer.

`predict_result` provides the final class labels of input data samples. It works for both binary classification and multi-classification.

`accuracy` calculates the ratio that the prediction of the neural network gives correct class.

`fit_model` trains the neural network based on back propagation. The process of back propagation changes when initiated with different activation functions. The method updates weight matrices using the gradient after training each sample. It allows setting learning epochs and learning rate to match the different demands in different datasets. It also provides the function of pruning the neurons in hidden layer after several epochs. The pruning rule is based on the L1 or L2 norm of the weight vector the neuron connecting to the output layers. When the norm is less than the threshold, the pruning will work on the neuron, which means all the weights into and out from the neuron will be masked. Parameter "verbose" can control whether to show the training process output, which contains accuracy and time using after each epoch.

2. Neural Network with Neuron Suppression

`ImprovedNeuralNetwork` incorporates the neuron "competition", which means the previous output of the neurons in the layer will be used when calculating the new output. The class I implemented employs the idea in calculating hidden layer.

`ImprovedNeuralNetwork` differs from `NeuralNetwork` in **`fit_model`** part. At the start of training, the weights connecting the previous hidden layer to the new hidden layer are randomly assigned. When a data sample comes in, the hidden layer outputs are set as zeros. Then, use both input features and existing hidden layer outputs to obtain new hidden layer outputs.

$$X_{\text{hid},t} = f(WX_{\text{in}} + UX_{\text{hid},t-1}), \quad f : \text{activation function}$$

This step repeats several times (default: 10 times) to get a relatively stable result. Through back propagation progress, the first step of updating weights connecting hidden layer and output layer is the same as `NeuralNetwork`, while the cumulative error is propagated through both W and U. So, these two weights need to be updated at the same time.

3. Support Vector Machine

Support Vector Machine is an effective machine learning method in high dimensional space. The hyperplane of the classifier only relies on a part of the training samples (i.e. support vectors). Radial based function (RBF) is one of the most popular kernel methods applied in SVM.

$$K(x, z) = \exp(-\gamma||x - z||^2), \quad \gamma > 0$$

For the SVM classifier with RBF kernel, the two hyper-parameters are regularization parameter (C) and kernel coefficient (γ). C balanced the complexity and ratio of mis-classification in the model. γ is the parameter controls the influence of one sample to the separator hyperplane. When we have greater C and less γ , the model has less support vectors and more complex.

Dataset Introduction

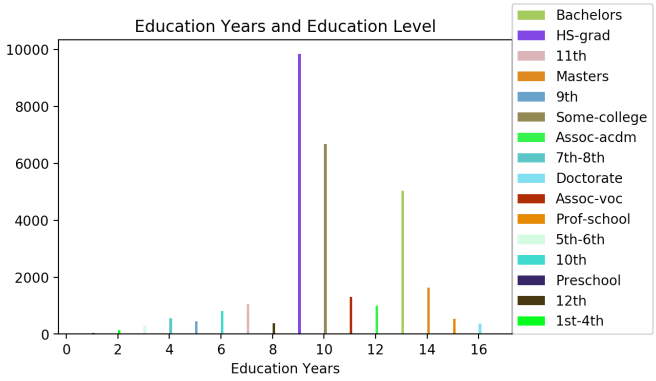
1. Iris Dataset

Iris dataset contains 50 samples from each of three different species of Iris. The features are the length and the width of the sepals and petals of flowers. I did not change any of the dataset. The dataset is used to test if `NeuralNetwork` can be applied in multi-classification problem.

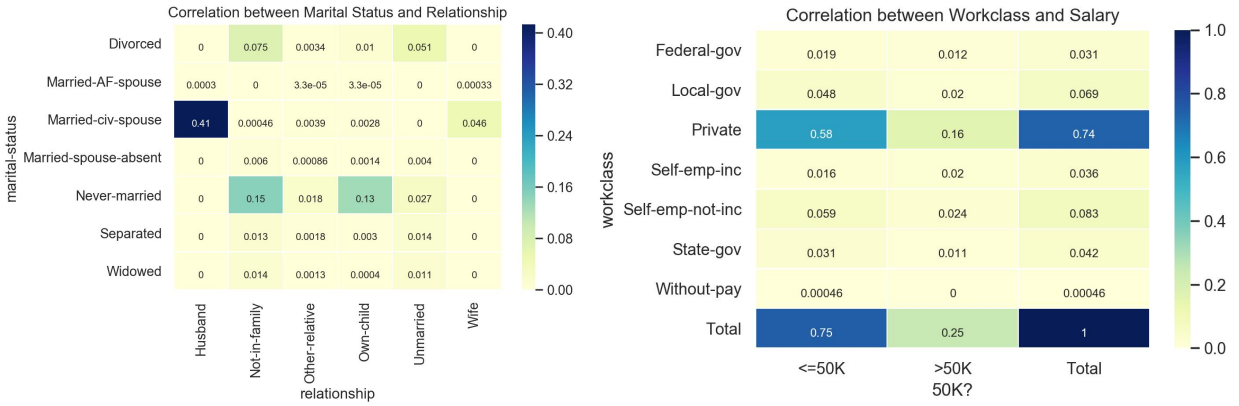
2. Adult Dataset

The original “Adult” dataset contains 32561 records with 14 features and 1 binary target. After removing missing values, 30162 records are kept. I dropped several columns which are not meaningful in classification or represent similar information with the kept columns.

The continuous variable fnlwgt represents final weight, which represents the number of units in the target population that the responding unit represents. I ignored it in my analysis.



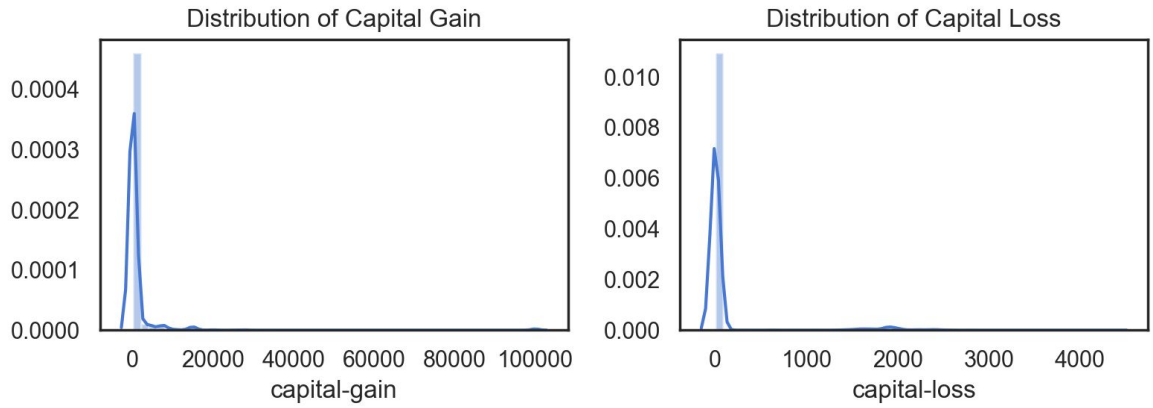
Each education level only corresponds to a unique education year. They contain the same information, therefore I removed education level.



Most of “Married-AF-spouse” and “Married-civ-spouse” are also “Husband” or “Wife” in a family relationship, while other marital statuses are not. So I kept marital status and determine “Married-AF-spouse” and “Married-civ-spouse” as “Married”, others as “Unmarried”.

“Workclass” and “occupation” carry similar information about the personal jobs. “Occupation” represents more specific jobs, which may not be suitable for a classification problem (easy causing overfitting). I kept “workclass” column and combined the class in public sectors, then ordered them by the ratio of earning 50K per year.

For the native country and race, they are both extremely unbalanced data. 91% samples are native American and 86% samples are white. 80% are white American. I only kept native country and combined all other countries except the U.S. as a group.



91.58% data of capital gain and 95.26% of capital loss are 0. At the same time, the range of the values is large, which contains little information. I removed these two columns.

The 7 kept features are “age”, “workclass”, “education year”, “marital status”, “native country”, “sex”, “hours per week”. All these features are converted to numerical scalar.

Features	**Transformation**
age, education year, hours per week	No change
workclass	0: Without-pay; 1: Private; 2: Self-emp-not-inc; 3: State-gov, Federal-gov, Local-gov; 4: Self-emp-inc
marital status	1: Married-AF-spouse, Married-civ-spouse; 0: Others
native country	1: United States; 0: Others
sex	1: Male; 0: Female

After the preprocess, all non-binary data are normalized before feeding in the classifier.

Results from Experiments

1. Neural Network Experiment on Multi-classification Problem (Iris Dataset)

`NeuralNetwork` is preset as 4 input neurons, 7 hidden neurons and 3 output neurons. The activation function is default, sigmoid function. The training process repeats 500 epochs and the step size for update is 0.2 (learning_epochs=500, learning_rate=0.2). After shuffling the data, I applied 5-fold cross validation to test the classifier.

Round	**1**	**2**	**3**	**4**	**5**

Result Row: Prediction Column: True Label															
	11			11			9			11			8		
		6			8			10			9			14	
			3		1	10		2	9			10			8
Test Accuracy	1.0			0.967			0.933			1.0			1.0		
Average Accuracy	0.980														

2. Neural Network Experiments on Adult Dataset

In experiments on "Adult" dataset, learning epoch is set as 20 for the size of the dataset is relatively large. Other parameters and settings may be changed in the following experiments. To reduce the impact of initiated random weights, the random seed for all the experiments remains the same to keep the weights consistent when hidden nodes are the same. Also, 5-fold cross validation is applied in each test. For convenience, I only present average accuracy and average running time here.

Tuning parameters for learning rate and number of neurons in hidden layer

The activation function is default as sigmoid function in tuning learning rate and number of neurons. For fixed learning rate 0.01, more neurons in hidden layer increase the complexity of the model and increase the accuracy in training set. Meanwhile, higher accuracy in training set may cause overfitting problem. When the number of neurons is fixed, tuning learning rate is more tricky. Too small learning rate leads to slow training process (need more training epoch to converge). Too large learning rate makes it hard to converge.

#	Learning Rate	Neurons in Hidden Layer	Avg. Accuracy (Training)	Avg. Accuracy (Test)	Time (s)
1	0.01	2	81.51%	81.45%	44.90
2	0.01	5	81.63%	81.57%	48.73
3	0.01	10	**81.76%**	**81.71%**	59.28

4	0.01	20	81.93%	81.85%	72.74
5	0.01	30	82.07%	**82.00%**	85.37
6	0.01	50	**82.14%**	81.13%	109.39
7	0.001	10	74.05%	73.54%	59.93
8	0.1	10	81.22%	80.88%	59.29
9	1	10	79.42%	79.16%	60.49
10	5	10	76.44%	76.18%	54.89

Changing activation function

The learning rate is 0.01 and number of neurons in hidden layer is 10. Sigmoid function and hyperbolic tangent give similar relatively high accuracy.

#	Activation Function	Avg. Accuracy (Training)	Avg. Accuracy (Test)	Time (s)
1	Sigmoid	81.76%	81.71%	59.28
2	Hyperbolic Tangent	**81.82%**	**81.75%**	51.83
3	Rectify Linear	76.46%	76.29%	50.50

Pruning

For the pruning experiment, the neural network is initiated with 10 neurons in hidden layer and hyperbolic tangent as activation function. The pruning rule is to deactivate the neuron when the L1 norm of its out weights (the absolute value of the weight in our experiment) is less than 0.2. The pruning rule begins to apply after 10 epochs. Although part of neural network is pruned, the accuracy does not change so much.

Round	1	2	3	4	5
Deactivated Node	4, 6, 7, 10	4, 6	4,6	4, 6	4, 6
Train	81.57%	81.86%	81.66%	81.43%	82.31%

Accuracy					
Test Accuracy	81.58%	81.42%	81.60%	82.71%	80.95%
Average Test Accuracy			**81.65%**		

Neural Network with Neuron Suppression

For the pruning experiment, the neural network is initiated with 10 neurons in hidden layer. Hyperbolic tangent and sigmoid function are tested. The training results are not stable.

Activation Function	**Round**	**Accuracy (Training)**	**Accuracy (Test)**	**Time (s)**
Hyperbolic Tangent	1	29.89%	30.89%	143.52
	2	66.35%	66.58%	146.85
	3	71.28%	71.27%	144.43
	4	27.74%	27.08%	147.58
	5	69.97%	68.78%	142.05
	Average	53.05%	52.92%	144.88
Sigmoid	1	73.06%	72.27%	144.67
	2	75.18%	75.48%	148.58
	3	69.81%	70.63%	139.36
	4	77.82%	79.01%	142.58
	5	75.13%	74.48%	145.94
	Average	74.20%	74.47%	144.22

3. Support Vector Machine Experiments on Adult Dataset

Tuning parameters for regularization parameter (C) and kernel coefficient (gamma)

I tried tuning the two important parameters (C and gamma) for the SVM using radial basis function as the kernel. The default C and gamma are 1 and 1/7 (7 comes from the number of features).

#	C	Gamma	Avg. Accuracy (Training)	Avg. Accuracy (Test)	Time (s)
1	0.001	Auto (0.1428)	75.10%	75.10%	7.37
2	0.01	Auto (0.1428)	75.64%	75.59%	7.31
3	0.1	Auto (0.1428)	81.92%	81.89%	6.84
4	1	Auto (0.1428)	82.06%	82.00%	6.58
5	10	Auto (0.1428)	**82.14%**	**82.05%**	7.33
6	1	0.001	75.11%	75.11%	7.65
7	1	0.01	81.91%	81.86%	6.70
8	1	0.1	82.05%	81.99%	6.49
9	1	1	82.22%	82.03%	7.24
10	1	10	**83.21%**	**82.54%**	8.87

Conclusion

`NeuralNetwork` can be applied in both binary classification and multi-classification problems. More neurons in hidden layer make the model more complex and increase the parameters need to be calculated, which results in longer time for training. Sigmoid and hyperbolic function both demonstrate better prediction result in "Adult" dataset. Pruning decreases the number of neurons in the model, but still has a stable and decent performance. Neural network with neuron suppression is not good enough for the dataset in our experiment. The performance of SVM

significantly rely on the parameters. For finding a better machine learning methods, tuning parameters is crucial and indispensable.

References

1. How the backpropagation algorithm works: neuralnetworksanddeeplearning.com/chap2.html
2. Backpropagation Through Time and Vanishing Gradients: www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/
3. Zhihua Zhou (2016). Machine Learning (Chinese Edition). China: Tsinghua University Press (p.97-p.115)
4. Scikit-learn SVM: scikit-learn.org/stable/modules/svm.html