CS 280: Intelligent Systems (TueG-A)

Programming Assignment 2

Multilayer Perceptron vs SVM

Meryll Angelica J. Viernes

2010-42676

1. Introduction

Works on artificial neural networks originated from the recognition that the human brain computes in entirely different way from the conventional digital computer. It has the ability to arrange its structural elements known as neurons to execute computations such as pattern recognition, perception, etc. faster that the fastest current digital computer. Specifically, perceptual recognition tasks such as recognizing familiar face in an unfamiliar scene can be routinely accomplished much faster than a powerful computer.

How does a human brain do it? A brain has considerable structure, the nervous system, at birth; it also has the ability to build up its own rules of behavior through "experience". Experience is built up over time with the development (learning to write) of the human brain taking place. A developing nervous system is similar to a plastic brain. Plasticity allows the developing nervous system to familiarize with the varying surrounding environment. Similar to the importance of plasticity to the functions of neurons as information-processing units in the human brain, artificial neurons are essential to neural networks.

A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: knowledge is acquired by the network from its environment through a learning process; interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The perceptron built around a single neuron is limited to performing pattern classification with only two classes. By expanding the output layer of the perceptron to include more than one neuron, we may correspondingly perform classification with more than two classes. However, the classes have to be linearly separable for the perceptron to work properly.

To overcome the practical limitations of the perceptron, we look to a neural network structure known as the multilayer perceptron. The following three points highlight the basic features of multilayer perceptron: (i) The model of each neuron in the network includes a nonlinear activation function that is differentiable, (ii) The network contains one or more layers that are hidden from both the input and output nodes, (iii) The network exhibits a high degree of connectivity, the extent of which is determined by synaptic weights of the network.

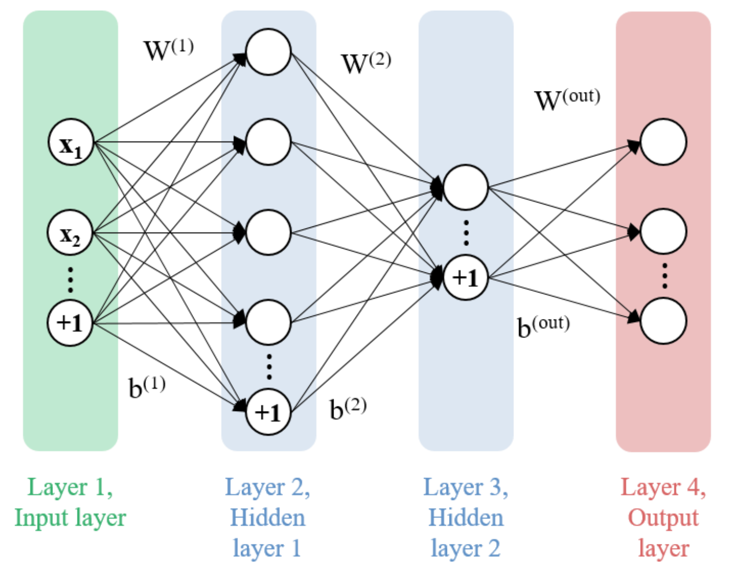


Figure 1. Neural Network Architecture

The hidden units are not part of the output or input of the network—hence their designation as "hidden." The first hidden layer is fed from the input layer. The resulting outputs of the first hidden layer are then applied to the next hidden layer; and so on for the rest of the network. Each hidden or output neuron of a multilayer perceptron is designed to perform two computations: the computation of the function signal, continuous nonlinear function of the input signal and corresponding synaptic weights; 2. the computation of an estimate of the gradient vector which is needed for the backward pass through the network.

A popular method for the training of multilayer perceptron is the back-propagation algorithm which has two phases: the forward phase and the backward phase.

In the forward phase, the synaptic weights of the network are fixed and the input signal is propagated through the network, layer by layer, until it reaches the output. Thus, in this phase, changes are confined to the activation potentials and outputs of the neurons in the network.

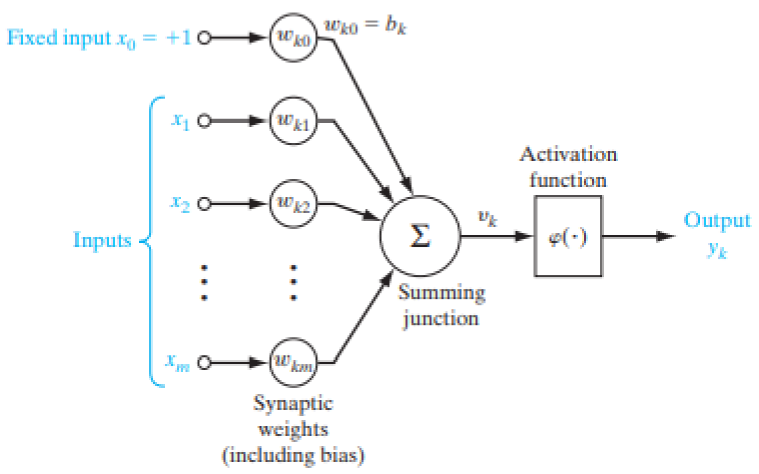


Figure 2. The model of each neuron in the network includes a nonlinear activation function that is differentiable.

Every input, x_i, including the bias with value fixed to 1, are multiplied with the corresponding synaptic weights, w_{ki}. The products are then summed to get v_k:

$$v_k = \sum_{i=0}^{m} w_{ki} x_i.$$

v_k then goes through a nonlinear device that implements the activation function to give an output with possible values between 0 and 1 or -1 and +1. Since we want a value between 0 and 1, the activation function used is the logistic function:

$$\varphi(v) = \frac{1}{1+e^{-\alpha v}}.$$

The predicted output, y_k, can then be compared to the desired output, d_k.

$$e_j(n) = d_k - y_k,$$

to compute for the pattern error:

$$\varepsilon(n) = \frac{1}{2}\sum_k e_j^2(n),$$

which would yield a network error:

$$\varepsilon_{ave} = \frac{1}{N}\sum \varepsilon\,(n)$$

that should be minimized in the backward phase to accurately predict the output.

In the backward phase, the resulting error signal is propagated through the network, again layer by layer, but this time the propagation is performed in the backward direction. In this second phase, successive adjustments are made to the synaptic weights of the network:

$$w_{ji}(n)_{new} = w_{ji}(n)_{old} - \eta \frac{\partial\varepsilon(n)}{\partial w},$$

which could be calculated through chain rule as:

$$w_{ji}(n)_{new} = w_{ji}(n)_{old} - \eta\delta_j(n) y_i(n),$$

where $y_i(n)$ is the input (or output of the previous layer), and $\delta_j(n)$ for output neuron, $o_j(n)$ is:

$$\delta_j(n) = e_j(n) o_j(n)\left[1 - o_j(n)\right],$$

and $\delta_j(n)$ for hidden neuron is:

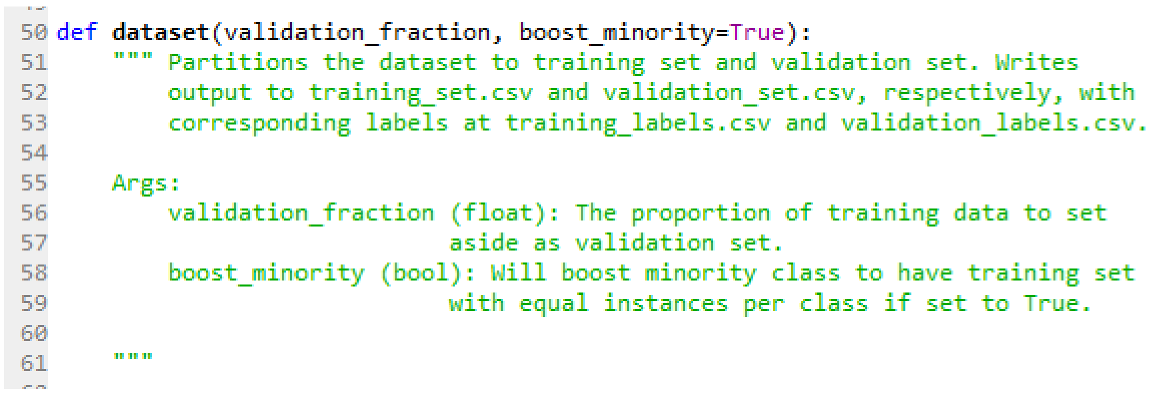
$$\delta_j(n) = y_j(n)[1 - y_i(n)] \sum \delta_k(n) w_{kj}(n),$$

where $\delta_k(n)$ is equivalent to $\varphi'(v)$ that could be calculated as $\varphi(v)[1 - \varphi(v)\,]$ for logistic function.

In this paper, a neural network with two hidden layers was created to predict the class label of an input with 354 attributes or features.

2. Methodology

An internal validation of the neural network that uses dataset that weren't used in computing the weights of the neurons is necessary to monitor prediction errors and verify the predictive performance of the neural network on an unbiased unseen data which is used as criterion as to when to stop the training. The weights of the network is not adjusted with this data set, it is just used to minimize overfitting and verify that an increase in accuracy over the training data actually yields an increase in accuracy over the data that the network hasn't trained on. Thus, the dataset was partitioned into training set and validation set.



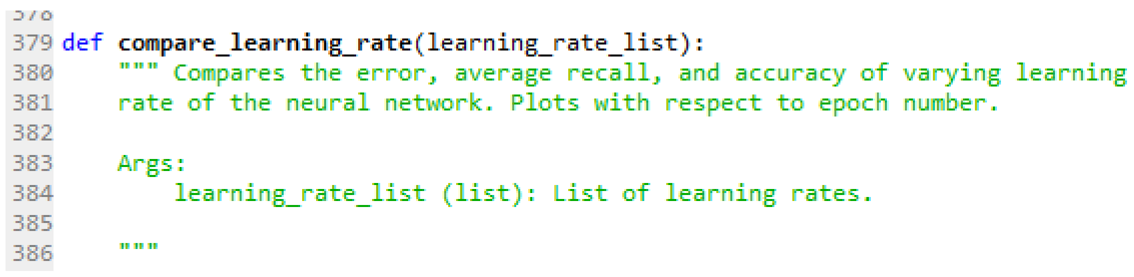
```python
def dataset(validation_fraction, boost_minority=True):
    """ Partitions the dataset to training set and validation set. Writes
        output to training_set.csv and validation_set.csv, respectively, with
        corresponding labels at training_labels.csv and validation_labels.csv.

    Args:
        validation_fraction (float): The proportion of training data to set
                                     aside as validation set.
        boost_minority (bool): Will boost minority class to have training set
                                     with equal instances per class if set to True.

    """
```

Figure 3. Function used in partitioning the data to training set and validation set.

Furthermore, since the dataset is highly imbalanced, there are huge differences in the number of instances for each class, the minority class was boosted by artificially creating new samples from the existing samples. This would ensure that there are equal instances over all classes. Training over imbalanced dataset would result to biased models since it tends to maximize the accuracy or minimize the overall error only which tends to focus on the most populous classes since that yield better accuracy.

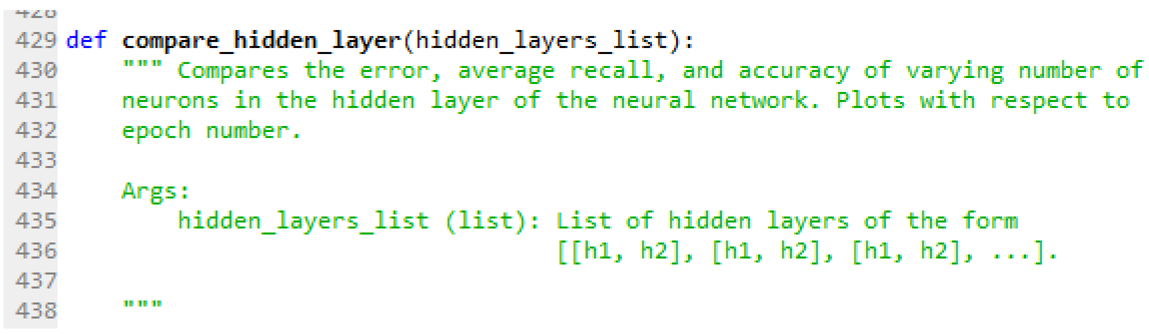
However, since highly imbalanced dataset is typical in real life data, the validation dataset was left imbalanced. Since checking for accuracy of an imbalanced dataset is not practical since it tends to neglect the minority class, average recall of prediction of neural network over the validation dataset was used as stopping criterion. Checking the average recall of the validation dataset would both ensure accurate prediction on all classes and accurate prediction on unseen data.

Decreasing the learning rate would increase the chance of accurately computing the weights that would give accurate prediction; however, this would increase the time to arrive at the said weights. Similarly, an increase in number of neurons would increase the accuracy of prediction though it also increases the runtime. Thus, the number of neurons in each hidden layer and the learning rate was varied to check which neural network architecture is efficient and accurate.



```python
def compare_learning_rate(learning_rate_list):
    """ Compares the error, average recall, and accuracy of varying learning
    rate of the neural network. Plots with respect to epoch number.

    Args:
        learning_rate_list (list): List of learning rates.

    """
```

Figure 4. Function used in comparing the performance of neural networks with different learning rate.



```python
def compare_hidden_layer(hidden_layers_list):
    """ Compares the error, average recall, and accuracy of varying number of
    neurons in the hidden layer of the neural network. Plots with respect to
    epoch number.

    Args:
        hidden_layers_list (list): List of hidden layers of the form
                                   [[h1, h2], [h1, h2], [h1, h2], ...].

    """
```

Figure 5. Function used in comparing the performance of neural networks with different number of neurons in the hidden layer.

Lastly, the performance in terms of accuracy and runtime of the created neural network was compared to prediction performance of a support vector machine (SVM).

3. Results and Discussion

As stated earlier, the dataset was balanced by boosting the minority class through artificially adding samples from the existing samples of the said class. The performance of neural network with balanced training data was compared to the imbalanced training data. As expected, the training using imbalanced dataset with stopping criterion of 95% average prediction recall ended at max epoch since the neural network was not able to predict the minority class; shown in Figure 6. As seen in Table 1, although 92.53% was predicted correctly using highly imbalanced training dataset, none of the least occurring class (class 3) was predicted correctly, and only 12.5% of the second least occurring class (class 7) was predicted correctly.

Table 1. Confusion matrix of neural network with imbalanced training dataset.

	1	2	3	4	5	6	7	8
Predicted 1	262	1	0	0	0	0	0	0
Predicted 2	2	37	2	0	1	0	0	3
Predicted 3	0	0	0	0	0	0	0	0
Predicted 4	0	0	0	76	0	0	0	2
Predicted 5	0	0	0	0	45	1	2	0
Predicted 6	1	0	0	2	0	44	4	3
Predicted 7	1	0	1	0	1	0	2	2
Predicted 8	4	0	2	2	0	6	0	67

The neural network training with balanced dataset was able to predict even the minority class as shown in Table 2.

Table 2. Confusion matrix of neural network with boosted minority class.

	1	2	3	4	5	6	7	8
Predicted 1	261	0	0	0	0	0	0	1
Predicted 2	2	38	0	0	0	0	0	2
Predicted 3	0	0	5	0	0	0	0	0
Predicted 4	1	0	0	78	0	0	0	0
Predicted 5	0	0	0	0	47	0	0	0
Predicted 6	0	0	0	2	0	47	1	4
Predicted 7	0	0	0	0	0	1	7	2
Predicted 8	6	0	0	0	0	3	0	68

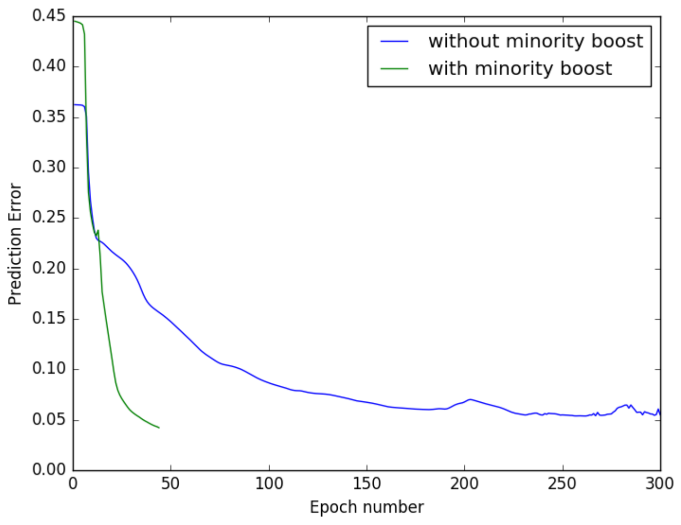


Figure 6. Prediction error over epoch number of balanced and imbalanced training set.

At every input instance in every epoch weights are adjusted through back propagation; thus, as the epoch number increases, the prediction error, which is minimized in back propagation, decreases.

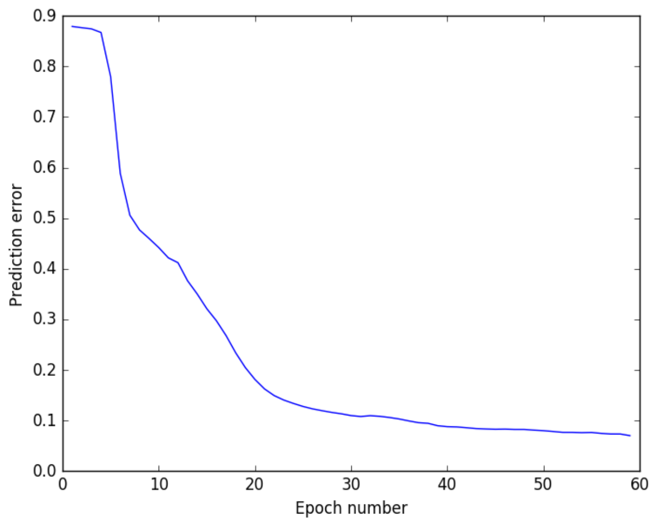


Figure 7. Prediction error over epoch number

As shown in Figure 8 and Table 3, increasing the learning rate decreases the training time or increases the rate at which the values for the synaptic weights that could give a reasonable prediction accuracy. However, training with high learning rate does not guarantee converging to an optimal or reasonable prediction accuracy or average recall. With learning rate 1.00, the prediction error fluctuates towards the end before reaching the desired stopping criterion, 95% average recall. Similarly, learning rete lower or equal to 0.1 has significantly increased the runtime; thus, 0.50 was chosen as the value of the learning rate since it both has reasonable runtime and relatively lower fluctuations in reaching the stopping criterion.

Table 3. Runtime of various learning rate.

learning rate	runtime
0.01	29.68 min
0.05	13.94 min
0.10	11.91 min
0.50	3.90 min
1.00	2.58 min

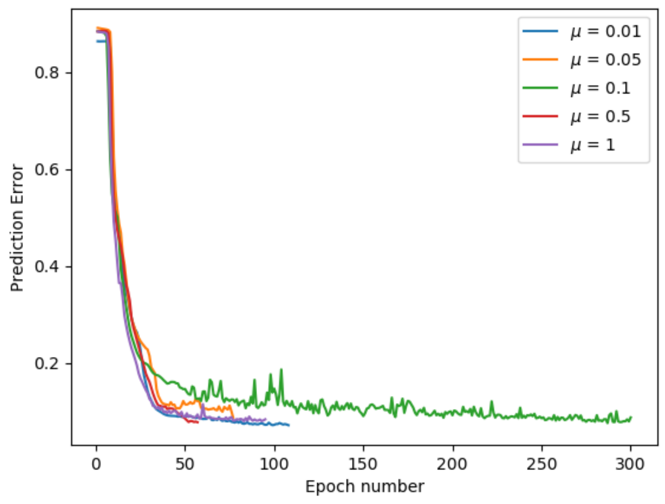


Figure 8. Prediction error over epoch number with various learning rate.

As seen in Figure 9, (i) increasing the number of neurons only in the first hidden layer worsen the training runtime and prediction accuracy of the neural network, (ii) increasing the number of neurons only in the second hidden layer does little in predictive accuracy, (iii) increasing number of neurons in both hidden layers improves the rate at which the prediction error converges to attain the weight that would give correct predicted outputs. However, an increase in the number of neurons also increases the runtime at which the training was done. Thus, similar to choosing the value for learning rate, the number of neurons with reasonable average prediction and runtime was chosen, specifically the 21-15.

Table 4. Runtime of various number of neurons in the hidden layer.

hidden layer	runtime
7-5	9.92 min
14-10	10.20 min
21-15	12.88 min
28-20	18.53 min

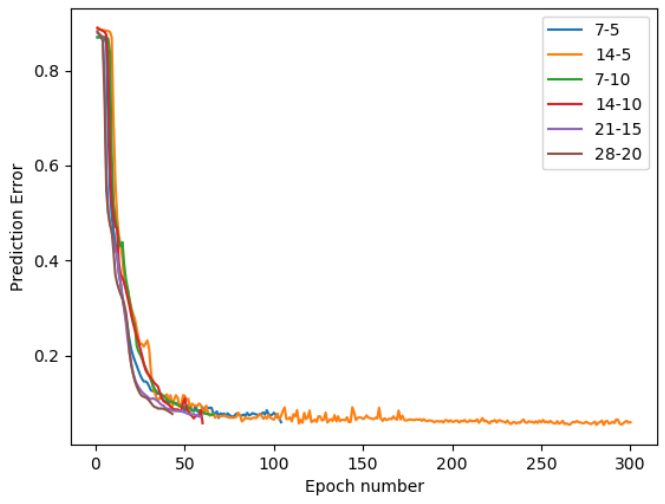


Figure 9. Prediction error over epoch number with various number of neurons in the hidden layer.

The prediction performance of the created neural network was compared to the prediction using SVM. It uses a linear kernel and it has more flexibility in the choice of penalties and loss functions and scales better to large numbers of samples. This class also supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme. It is computational efficient since it only has classifiers equivalent to number of classes needed. For each classifier, the class is fitted against all the other classes.

As shown in Table 5, SVM has higher accuracy. This is due to the nature of computation of the algorithms. A neural network guarantees to find a classifier that would satisfy the user-identified condition, which is used as stopping criterion in the continuous training to minimize the error; it only guarantees to find a local minima, while SVM guarantees to find the global minima. Thus, it is natural to get higher accuracy using SVM. However, a significant advantage of using MLP over SVM in prediction is that for SVM, you'll have to guess the kernel function to be used; you'll have to know the pattern of the data which is not commonly known in real life analysis, while for MLP, you'll only have to check which neural network architecture best complements your data.

Table 5. Performance comparison of MLP and SVM.

	MLP	SVM
Accuracy	94.62%	95.49%
Average Recall	96.00%	96.31%
Runtime	10.01 min	0.191 min

4. Conclusion

Prediction error decreases as epoch number increases by undergoing several adjustments on synaptic weights through back propagation. Also, balancing the dataset also increases prediction accuracy in the minority class. Decreasing the learning rate and increasing the number of neurons in the hidden layer is a trade-off between accuracy and runtime. An increase in learning rate and decrease in number of neurons in the hidden layer could significantly decrease the training time but could also decrease the accuracy of prediction.

References

Haykin, S. (2009) Neural Networks and Learning Machines. 3rd Edition, Prentice- Hall, Englewood Cliffs, NJ.