CS 445/545: Machine Learning
Programming Assignment#1

Dataset: The dataset used for this assignment is MNIST dataset
https://www.kaggle.com/oddrationale/mnist-in-csv

EXPERIMENT 1: Vary number of hidden inputs
N = 20,50,100

Q & A

1) How does the number of hidden units affect the final accuracy of the test data?

   The number of hidden units doesn't have a significant effect on the accuracy. The training of the network significantly improves with the increase in hidden units, resulting in better accuracy.

2) How does it affect the number of epochs needed for training to converge?

   The training set can converge with fewer iterations. Less number of epochs are required with the increase in the number of hidden units.

3) Is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?
   There is no overfit to the training data for my network as I used the given parameters.

4) How do your results compare to the results obtained by your perceptron in HW 1?

The accuracy of perceptron in HW1 was 85%, whereas the accuracy of perceptron in this case is over 95% which is more accurate that HW 1.

REPORT:

We are using MNIST dataset for this experiment. The dataset consists of 60000 training examples and labels, 10000 test examples, and labels. Here I am implementing a two-layered neural network that is used for handwritten digit recognition. A neural network structure is designed with 784 inputs, 1 hidden layer with n hidden units, and 10 output units. Each output unit should correspond to one of the 10 classes (0 to 9). In the program, I have used backpropagation with stochastic gradient descent to train the network.

First, we have to load the data and visualize it. The MNIST dataset consists of handwritten digits. For the first experiment, we must vary the number of hidden inputs(n). Initialize the values of weights and scale the data values between 0 and 1, divide it by 255. Then consider momentum(m) = 0.9, learning rate= 0.1, bias = 1. The value n changes from 20,50,100 while training the data. I used 50 epochs while training the network. By comparing the training data with the test data, I determined the accuracy. When n=20, accuracy is over 95%, n=50, accuracy is over 96%, n = 100, accuracy was over 96.3% by the end of 49th epoch. The accuracy values for test and train data are then saved in the CSV files which are used to plot the graph.

Experiment 2: Vary the momentum value

M = 0,0.25,0.5

Q & A

1) How does the momentum value affect the final accuracy on the test data?
   Accuracy of the data when m=0 is 97.71% and m= 0.5 is 97.43%. When the momentum value is increased, the accuracy drops slightly.

2) How does it affect the number of epochs needed for the training to converge?
   We can observe that the accuracy was quite low in the initial epochs, but that it improved after a few iterations. With the increase in epochs, the network gets trained and more accurate. The training set converges faster when we increase the momentum value.

3) Again, is there evidence that any of your network has overfit to the training data? If so, what is the evidence?
   There is no overfitting in the training data with the given parameters.

Report:

We are using MNIST dataset for this experiment. The dataset consists of 60000 training examples and labels, 10000 test examples, and labels. Here I am implementing a two-layered neural network that is used for handwritten digit recognition. A neural network structure is designed with 784 inputs, 1 hidden layer with n hidden units, and 10 output

units. Each output unit should correspond to one of the 10 classes (0 to 9). In the program, I have used backpropagation with stochastic gradient descent to train the network.

First, we have to load the data and visualize it. The MNIST dataset consists of handwritten digits. For the first experiment, we must vary the number of hidden inputs(n). Initialize the values of weights and scale the data values between 0 and 1, divide it by 255. Then consider n = 100, learning rate= 0.1, bias = 1. In this experiment, the momentum(m) varies, so consider m = 0,0.25,0.5. I used 50 epochs while training the network. By comparing the training data with the test data, I determined the accuracy. When m= 0, accuracy is 97.1%, m =0.25, accuracy is
97.6%, m = 0.5, accuracy is 97.4%. The accuracy values for test and train data are then saved in the CSV files which are used to plot the graph.

Experiment 3: Vary the number of training examples

Training examples = 30000,15000

Q & A

1) How does the size of the training data effect the final accuracy on the test data?
   A dataset is more efficient if it has several training examples. There is a possibility that large datasets can reduce the accuracy which results in increase of the errors.

2) How does it affect the number of epochs needed for the training to converge?

If we have several training examples (large training data), training data doesn't need many epochs to converge.

3) Again, is there evidence that any of the networks has overfit to the training data? If so, what is that evidence?
   There is no overfitting in the training data

Report:

We are using MNIST dataset for this experiment. The dataset consists of 60000 training examples and labels, 10000 test examples, and labels. Here I am implementing a two-layered neural network that is used for handwritten digit recognition. A neural network structure is designed with 784 inputs, 1 hidden layer with n hidden units, and 10 output units. Each output unit should correspond to one of the 10 classes (0 to 9). In the program, I have used backpropagation with stochastic gradient descent to train the network.

First, we have to load the data and visualize it. The MNIST dataset consists of handwritten digits. For the first experiment, we must vary the number of hidden inputs(n). Initialize the values of weights and scale the data values between 0 and 1, divide it by 255. Then consider n = 100, learning rate= 0.1, bias = 1, momentum (m) = 0.9. In this experiment, we need to use different training examples. I considered 15000 (1/4th the original training examples) and 30000(1/2th of the original training examples). I used 50 epochs while training the network. By comparing the training data with the test data, I determined the accuracy. When training examples are 15000, the accuracy obtained was over 95%, when the examples are 30000, the

accuracy obtained is 96.1%. The accuracy values for test and train data are then saved in the CSV files which are used to plot the graph.