CS446: Machine Learning **Spring 2017**

Problem Set 5

Zhiyu Zhu (zzhu24) *Handed In: April 13, 2017*

1. [**Neural Networks**]

   (a) According to the deciation of the learning rule:

$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$
$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$

$$Err_d(\vec{w}) = \frac{1}{2}\sum_{k\in K}(t_k - o_k)^2$$
$$\frac{\partial E_d}{\partial o_j} = -(t_j - o_j)$$

$$o_j = \frac{1}{1+e^{-(net_j - T_j)}}$$
$$\frac{\partial o_j}{\partial net_j} = o_j(1-o_j) = \begin{cases} 0, & \text{if } net_j \leq 0 \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

$$net_j = \sum w_{ij}\cdot x_{ij}$$
$$\frac{\partial net_j}{\partial w_{ij}} = x_{ij}$$

$$\triangle w_{ij} = R(t_j - o_j)o_j(1-o_j)x_{ij} = R\delta_j x_{ij}$$
$$\delta_j = (t_j - o_j)o_j(1-o_j)$$

$$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}}$$
$$\frac{\partial E_d}{\partial w_{ij}} = \sum_{k\in downstream(j)}\frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial net_j}x_{ij}$$

$$\frac{\partial E_d}{\partial w_{ij}} = \sum_{k \in downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j} x_{ij}$$

$$\frac{\partial E_d}{\partial w_{ij}} = \sum_{k \in downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} x_{ij}$$

$$\frac{\partial E_d}{\partial w_{ij}} = \sum_{k \in downstream(j)} -\delta_k w_{jk}$$

As we deducted above:

$$\frac{\partial o_j}{\partial net_j} = o_j(1 - o_j) = \begin{cases} 0, & \text{if } net_j \leq 0 \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

$$\frac{\partial E_d}{\partial w_{ij}} = \begin{cases} 0, & \text{if } net_j \leq 0 \\ \sum_{k \in downstream(j)} -\delta_k w_{jk}, & \text{otherwise} \end{cases} \tag{3}$$

Therefore:

$$\delta_j = \begin{cases} 0, & \text{if } net_j \leq 0 \\ \sum_{k \in downstream(j)} -\delta_k w_{jk}, & \text{otherwise} \end{cases} \tag{4}$$

a i. **Data for domain= circles**
domain= circles , batch size= 10 , learning rate= 0.1 , activation function= relu , hidden layer width= 10 , accuracy= 100.0%
domain= circles , batch size= 10 , learning rate= 0.1 , activation function= relu , hidden layer width= 50 , accuracy= 100.0%
domain= circles , batch size= 10 , learning rate= 0.1 , activation function= tanh , hidden layer width= 10 , accuracy= 100.0%
domain= circles , batch size= 10 , learning rate= 0.1 , activation function= tanh , hidden layer width= 50 , accuracy= 100.0%
domain= circles , batch size= 10 , learning rate= 0.01 , activation function= relu , hidden layer width= 10 , accuracy= 98.875%
domain= circles , batch size= 10 , learning rate= 0.01 , activation function= relu , hidden layer width= 50 , accuracy= 100.0%
domain= circles , batch size= 10 , learning rate= 0.01 , activation function= tanh , hidden layer width= 10 , accuracy= 50.0%
domain= circles , batch size= 10 , learning rate= 0.01 , activation function= tanh , hidden layer width= 50 , accuracy= 51.5%
domain= circles , batch size= 50 , learning rate= 0.1 , activation function= relu , hidden layer width= 10 , accuracy= 100.0%
domain= circles , batch size= 50 , learning rate= 0.1 , activation function= relu , hidden layer width= 50 , accuracy= 100.0%

domain= circles , batch size= 50 , learning rate= 0.1 , activation function= tanh , hidden layer width= 10 , accuracy= 56.875%
domain= circles , batch size= 50 , learning rate= 0.1 , activation function= tanh , hidden layer width= 50 , accuracy= 59.625%
domain= circles , batch size= 50 , learning rate= 0.01 , activation function= relu , hidden layer width= 10 , accuracy= 62.125%
domain= circles , batch size= 50 , learning rate= 0.01 , activation function= relu , hidden layer width= 50 , accuracy= 95.125%
domain= circles , batch size= 50 , learning rate= 0.01 , activation function= tanh , hidden layer width= 10 , accuracy= 48.875%
domain= circles , batch size= 50 , learning rate= 0.01 , activation function= tanh , hidden layer width= 50 , accuracy= 50.0%
domain= circles , batch size= 100 , learning rate= 0.1 , activation function= relu , hidden layer width= 10 , accuracy= 100.0%
domain= circles , batch size= 100 , learning rate= 0.1 , activation function= relu , hidden layer width= 50 , accuracy= 100.0%
domain= circles , batch size= 100 , learning rate= 0.1 , activation function= tanh , hidden layer width= 10 , accuracy= 50.375%
domain= circles , batch size= 100 , learning rate= 0.1 , activation function= tanh , hidden layer width= 50 , accuracy= 49.0%
domain= circles , batch size= 100 , learning rate= 0.01 , activation function= relu , hidden layer width= 10 , accuracy= 66.25%
domain= circles , batch size= 100 , learning rate= 0.01 , activation function= relu , hidden layer width= 50 , accuracy= 83.0%
domain= circles , batch size= 100 , learning rate= 0.01 , activation function= tanh , hidden layer width= 10 , accuracy= 48.75%
domain= circles , batch size= 100 , learning rate= 0.01 , activation function= tanh , hidden layer width= 50 , accuracy= 49.875%

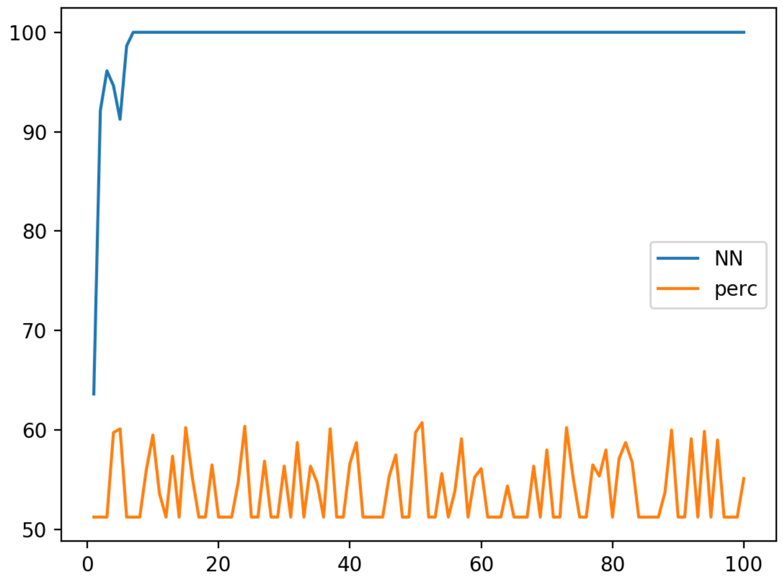
best parameter and accuracy:
domain= circles , batch size= 10 , learning rate= 0.1 , activation function= relu , hidden layer width= 10 , accuracy= 100.0%

ii. **Data for domain= mnist**
domain= 'mnist' , batch size= 10 , learning rate= 0.1 , activation function= 'relu', hidden layer width=10 , accuracy= 95.8018576972%
domain= 'mnist' , batch size= 10 , learning rate= 0.1 , activation function= 'relu', hidden layer width=50] accuracy= 95.9270872639%
domain= 'mnist' , batch size= 10 , learning rate= 0.1 , activation function= 'tanh', hidden layer width=10] : accuracy= 96.9119647564%
domain= 'mnist' , batch size= 10 , learning rate= 0.1 , activation function= 'tanh', hidden layer width=50] : accuracy= 96.9286801327%
domain= 'mnist' , batch size= 10 , learning rate= 0.01 , activation function=

'relu', hidden layer width=10] : accuracy= 96.5698042748%
domain= 'mnist' , batch size= 10 , learning rate= 0.01 , activation function= 'relu' , hidden layer width=50] : accuracy= 96.3694494995%
domain= 'mnist' , batch size= 10 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=10] : accuracy= 96.7700685463%
domain= 'mnist' , batch size=10 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=50] : accuracy= 96.235893573%
domain= 'mnist' , batch size=50 , learning rate= 0.1 , activation function= 'relu' , hidden layer width=10] : accuracy= 96.4863387826%
domain= 'mnist' , batch size=50 , learning rate= 0.1 , activation function= 'relu' , hidden layer width=50] : accuracy= 96.0440809745%
domain= 'mnist' , batch size=50 , learning rate= 0.1 , activation function= 'tanh' , hidden layer width=10] : accuracy= 96.886929982%
domain= 'mnist' , batch size=50 , learning rate= 0.1 , activation function= 'tanh' , hidden layer width=50] : accuracy= 96.5614222202%
domain= 'mnist' , batch size=50 , learning rate= 0.01 , activation function= 'relu' , hidden layer width=10] : accuracy= 96.6949990323%
domain= 'mnist' , batch size=50 , learning rate= 0.01 , activation function= 'relu' , hidden layer width=50] : accuracy= 96.6365752763%
domain= 'mnist' , batch size=50 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=10] : accuracy= 96.3360674797%
domain= 'mnist' , batch size=50 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=50] : accuracy= 96.1274350774%
domain= 'mnist' , batch size=100 , learning rate= 0.1 , activation function= 'relu' , hidden layer width=10] : accuracy= 96.6198598999%
domain= 'mnist' , batch size=100 , learning rate= 0.1 , activation function= 'relu' , hidden layer width=50] : accuracy= 96.44458167%
domain= 'mnist' , batch size=100 , learning rate= 0.1 , activation function= 'tanh' , hidden layer width=10] : accuracy= 96.7700615844%
domain= 'mnist' , batch size=100 , learning rate= 0.1 , activation function= 'tanh' , hidden layer width=50] : accuracy= 96.2442477802%
domain='mnist' , batch size=100 , learning rate= 0.01 , activation function= 'relu' , hidden layer width=10] : accuracy= 96.6782906178%
domain='mnist' , batch size=100 , learning rate= 0.01 , activation function= 'relu' , hidden layer width=50] : accuracy= 96.6449155599%
domain='mnist' , batch size=100 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=10] : accuracy= 96.2526159111%
domain='mnist' , batch size=100 , learning rate= 0.01 , activation function= 'tanh' , hidden layer width=50] : accuracy= 96.1942269643%

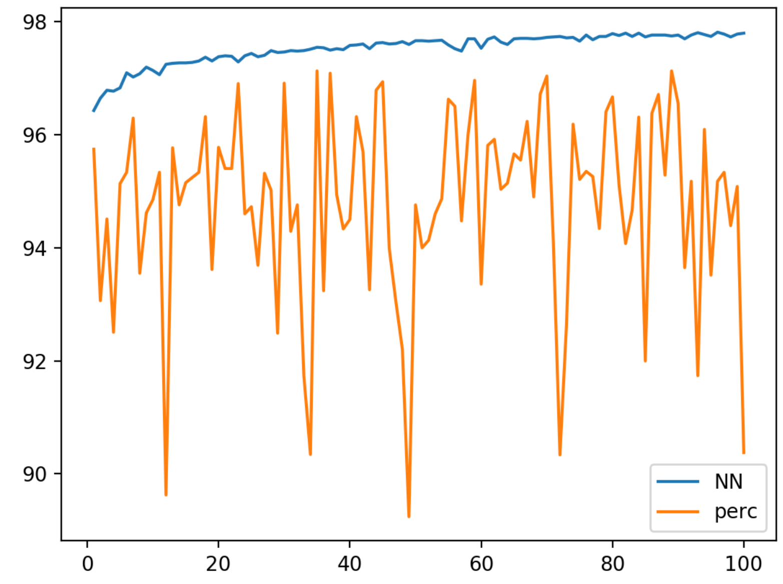
best parameter and accuracy:
domain='mnist', batch size=10, learning rate=0.1, activation function='tanh', hidden layer width=50] : accuracy=96.1942269643%

Data for domain= circles



Accuracy of NN:100.0%
Accuracy of Perceptron: 46.875%

Data for domain= mnist



Accuracy of NN:93.823%
Accuracy of Perceptron: 88.71%

(b) **Analysis**

i. Circles:
In the case that we chose circles as the domain, the neural network gives a lot more better algorithm than the perceptron. The perceptron is a linear separators so it cannot generate a fitting separator for the circle data. However, the neural network is perfect on learning non linear functions. So we can see from the graph and the result that the neural network gives a high and stable accuracy while the perceptron gives a low accuracy.

ii. Mnist:
In this case, the neural network is still preferred. The accuracy is not only very high but also very stable. Since the neural network works on any non linear data set, it is fit for the mnist database. Though the perceptron also has a high accuracy, it is quite unstable.

2. [**Multi-class classification**]

(a) i. One vs ALL: k separators
All vs ALL: k^k separators

ii. One vs ALL: k separators each used m examples
All vs ALL: k^k separators each used $\frac{2m}{k}$ examples

iii. One vs ALL: the final label is decided by Winner Takes All rules: $f(x) = argmax_i(\vec{w_i}^T \cdot \vec{x})$
All vs ALL: the final label is decided by the most output label chosen by separators

iv. One vs ALL: complexity is O(km)
All vs ALL: complexity is O(km)

(b) In this case, One vs All is preferred. First of all, in multi-class cases, if k is large, the data sized used for training in All vs All is pretty small compared to the One vs All. With more iteration of training with more data, the algorithm will be much more precise when tested on the testing data. Then, since One vs All needs much less separators, it is more easy to implement in the multi-class cases that k is large.

(c) In this case, in contrast, All vs all is preferred. Since the dual space in the kernel perceptron increases the efficiency, the complexity of the All to All benefits it to generate the algorithm faster.

(d) One vs All : O(kdmm)
All vs All : O(dm)
All vs All preferred

(e) One vs All : O(kddm)
All vs All : O(kddm)

(f) Counting : O(kk)
Knock Out : O(k)

3. [**Probability Review**]

(a) i. Town A:

$$E(A) = \frac{1+1}{2} = 1$$

Town B:

$$E(A) = \lim_{x\to\infty} \sum_{i=1}^{x} i \cdot (\frac{1}{2})^i = 2$$

ii. The ratio is same as the boy to girl ratio as 1:1 at the end of a generation for both Town A and Town B.

(b) i.

$$P(A,B) = P(A) \cdot P(B|A)$$
$$P(B,A) = P(B) \cdot P(A|B)$$
$$P(A,B) = P(B,A)$$
$$P(A) \cdot P(B|A) = P(B) \cdot P(A|B)$$
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

ii.

$$P(A,B,C) = P(C|B,A) \cdot P(B|A) \cdot P(A)$$

(c)

$$E(X) = 1 \cdot P(A) + 0 \cdot (1 - P(A))$$
$$E(X) = P(A)$$

(d) i. They are not dependent, according to b(i) and the chart:

$$P(X = 0) = \frac{1}{15} + \frac{1}{10} + \frac{4}{15} + \frac{8}{45} = \frac{11}{18}$$
$$P(Y = 0) = \frac{1}{15} + \frac{1}{15} + \frac{4}{15} + \frac{2}{15} = \frac{8}{15}$$
$$P(X = 0, Y = 0) = \frac{1}{15} + \frac{4}{15} = \frac{1}{3}$$
$$P(X = 0) \cdot P(Y = 0) \neq P(X = 0, Y = 0)$$
$$P(X) \neq P(X|Y)$$

so, they are not dependent.

ii. They are conditionally dependent, according to b(ii) and the chart:

$$P(X = 0, Y = 0, Z = 0) = \frac{1}{15}$$
$$P(Z = 0) = \frac{1}{15} + \frac{1}{15} + \frac{1}{10} + \frac{1}{10} = \frac{1}{3}$$
$$P(X = 0, Z = 0) = \frac{1}{15} + \frac{1}{10} = \frac{1}{6}$$
$$P(Y = 0, Z = 0) = \frac{1}{15} + \frac{1}{15} = \frac{2}{15}$$
$$P(X = 0, Z = 0) \cdot P(Y = 0, Z = 0) = \frac{1}{45}$$
$$P(X = 0, Y = 0, Z = 0) = \frac{P(X = 0, Z = 0) \cdot P(Y = 0, Z = 0)}{P(Z = 0)}$$

so, they are conditionally dependent

iii.

$$P(X + Y > 0) = P(X = 1, Y = 0) + P(X = 1, Y = 1) + P(X = 0, Y = 1)$$
$$P(X + Y > 0) = \frac{1}{15} + \frac{1}{10} + \frac{1}{10} + \frac{2}{15} + \frac{8}{45} + \frac{4}{45} = \frac{2}{3}$$
$$P(X = 0, X + Y > 0) = P(X = 0, Y = 1) = \frac{1}{10} + \frac{8}{45} = \frac{5}{18}$$
$$P(X = 0|X + Y > 0) = \frac{P(X = 0, X + Y > 0)}{P(X + Y > 0)} = \frac{5}{12}$$