CS446: Machine Learning **Spring 2017**

Problem Set 5

Kexin Hui *Handed In: April 12, 2017*

1. Answer to problem 1 - Neural Networks

   a. We need to calculate some derivatives beforehand in order to derive the backpropogation weight update rules.

$$
\begin{aligned}
Err_d(x_i, w) = \frac{1}{2}\sum_{k\in K}(t_k - o_k)^2 \Rightarrow \frac{\partial E_d}{\partial o_j} &= -(t_j - o_j) \\
o_j = max(0, net_j)) \Rightarrow \frac{\partial o_j}{\partial net_j} &= \begin{cases} 1 & \text{if } net_j > 0 \\ 0 & \text{if } net_j \leq 0 \end{cases} \\
net_j = \sum w_{ij}x_i \Rightarrow \frac{\partial net_j}{\partial w_{ij}} &= x_i
\end{aligned} \tag{1}
$$

   For the connection between the hidden layer and the output layer (influences output j through net):

$$
\begin{aligned}
\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} &= \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \\
&= \begin{cases} -(t_j - o_j)x_i & \text{if } net_j > 0 \\ 0 & \text{if } net_j \leq 0 \end{cases}
\end{aligned} \tag{2}
$$

   For the connection between the hidden layer and the input layer

$$
\begin{aligned}
\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} &= \sum_{k\in downstream(j)} \frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \\
&= \sum_{k\in downstream(j)} \frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \\
&= \sum_{k\in downstream(j)} \frac{\partial E_d}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \\
&= \begin{cases} \sum_{k\in downstream(j)} -(t_k - o_k)w_{jk}x_i & \text{if } net_j > 0 \text{ and } net_k > 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{3}
$$

   where $\frac{\partial net_k}{\partial o_j} = w_{ij}$.
   To conclude, based on the backpropogation algorithm, we calculate the error terms for each output unit and each hidden unit respectively as above, then update weights by $\Delta w_{ij} = -R\frac{\partial E_d}{\partial w_{ij}}x_i$ where R is the learning rate.

b. i. $squared_loss_gradient$ function is quite straightforward based on $\frac{\partial E_d}{\partial o_j}$ in Equation (1). Simply return $output\_activations - y$.
$relu_derivative$ equals to 1 when the data value is greater than 0 according to $\frac{\partial o_j}{\partial net_j}$ in Equation (1). That is, return $1.0 * (z > 0/0)$.

ii. I used five fold cross validation to try all the combinations of parameters. The results are shown below.
For the dataset consisting of two concentric circles:

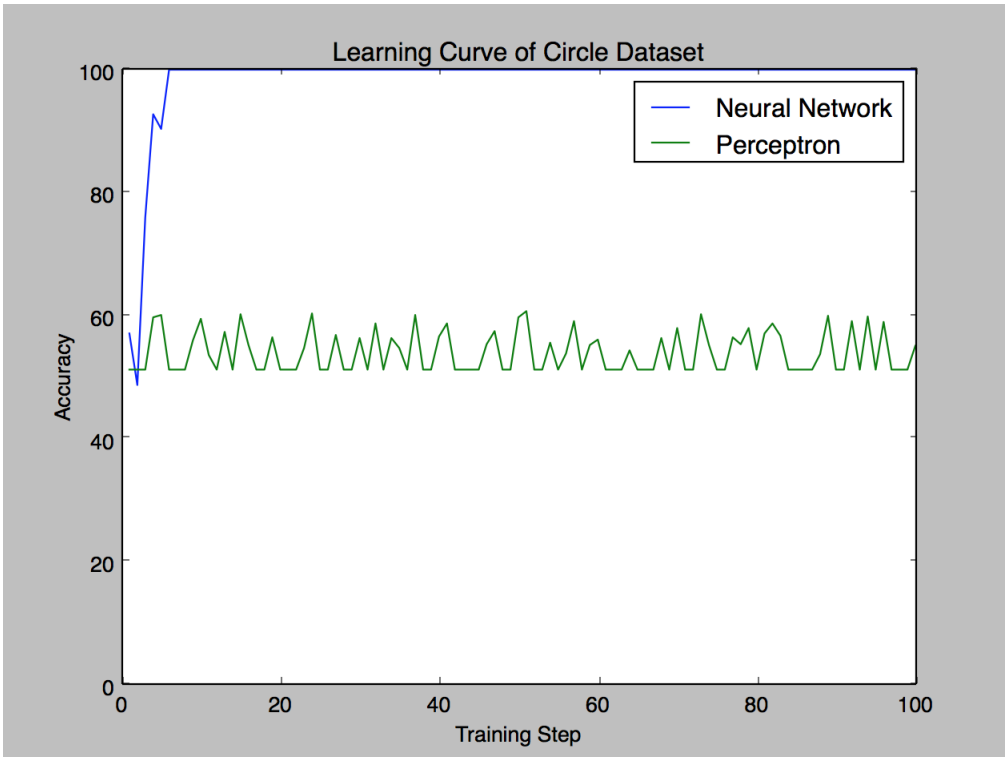
```
dataset: circle
Batch Size: 10 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 52.46875
Batch Size: 10 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 52.0
Batch Size: 10 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 77.3125
Batch Size: 10 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 90.46875
Batch Size: 10 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 62.96875
Batch Size: 10 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 71.71875
Batch Size: 10 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 99.6875
Batch Size: 10 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 100.0
Batch Size: 50 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 51.21875
Batch Size: 50 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 50.78125
Batch Size: 50 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 55.09375
Batch Size: 50 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 60.4375
Batch Size: 50 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 50.4375
Batch Size: 50 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 50.4375
Batch Size: 50 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 92.34375
Batch Size: 50 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 97.6875
Batch Size: 100 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 50.8125
Batch Size: 100 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 52.5
Batch Size: 100 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 52.3125
Batch Size: 100 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 56.5
Batch Size: 100 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 51.625
Batch Size: 100 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 50.65625
Batch Size: 100 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 75.125
Batch Size: 100 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 88.5625
```

I notice that one set of parameters giving 100% accuracy. Therefore, that should be our best performing parameters: Batch Size = 10, Learning Rate = 0.1, Activation Function = 'relu', Hidden Layer Width = 50.
For the dataset consisting of a subset of the mnist digit dataset:

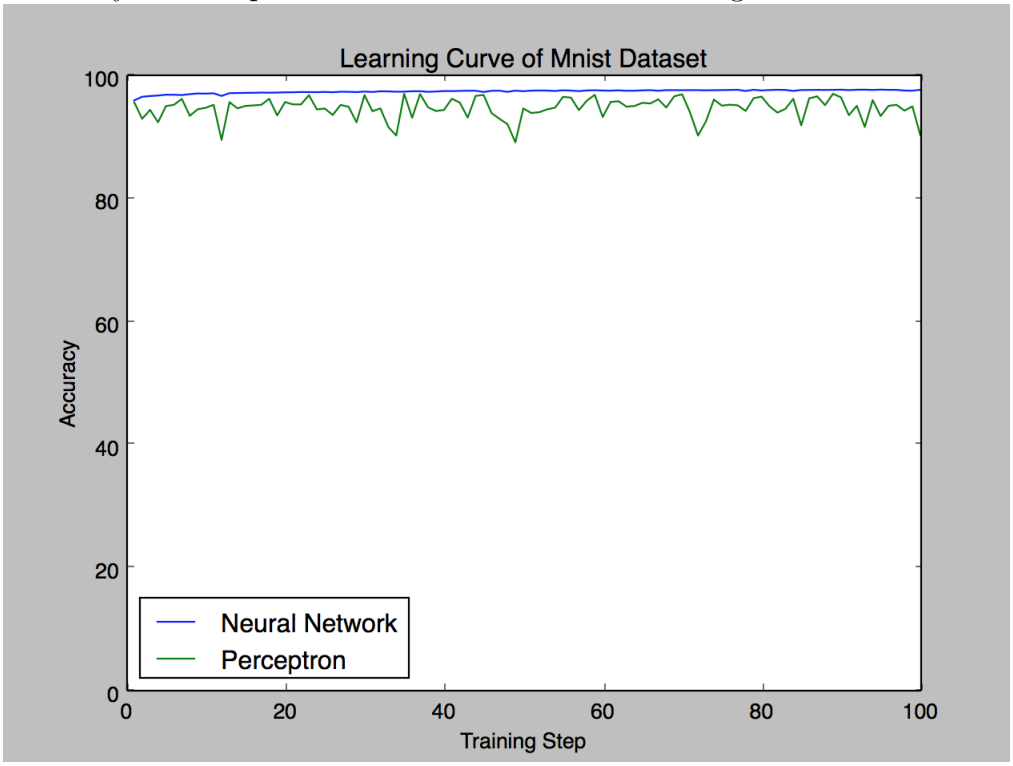
```
dataset: mnist
Batch Size: 10 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 96.2799916545
Batch Size: 10 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 95.6353014813
Batch Size: 10 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 95.7291883997
Batch Size: 10 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 95.7437930315
Batch Size: 10 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 96.4427289798
Batch Size: 10 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 96.3488420613
Batch Size: 10 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 94.7694554559
Batch Size: 10 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 95.2493219278
Batch Size: 50 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 96.0337992906
Batch Size: 50 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 96.0025036512
Batch Size: 50 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 96.2925099103
Batch Size: 50 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 96.3279783017
Batch Size: 50 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 96.3989150845
Batch Size: 50 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 95.6958063843
Batch Size: 50 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 95.6415606092
Batch Size: 50 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 95.5643646985
Batch Size: 100 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 95.8877529731
Batch Size: 100 Learning Rate: 0.01 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 95.8814938452
Batch Size: 100 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 10  Accuracy: 96.2799916545
Batch Size: 100 Learning Rate: 0.01 Activation Function: relu Hidden Layer Width: 50  Accuracy: 96.2883371584
Batch Size: 100 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 10  Accuracy: 96.2695597747
Batch Size: 100 Learning Rate: 0.1 Activation Function: tanh Hidden Layer Width: 50  Accuracy: 95.7437930315
Batch Size: 100 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 10  Accuracy: 94.7673690799
Batch Size: 100 Learning Rate: 0.1 Activation Function: relu Hidden Layer Width: 50  Accuracy: 95.8773210933
```

Overall, the accuracies look good for all parameter settings. The top performance is 96.4427289798% for accuracy. And the corresponding parameters are: Batch Size = 10, Learning Rate = 0.1, Activation Function = 'tanh', Hidden Layer Width = 10.

iii. I used to the best parameters solved by previous question. The learning curves for both datasets and the testing accuracy results are attached below.



Accuracy of Neural Network is 100% for the circles testing data.
Accuracy of Perceptron is 49.5% for the circles testing data.



Accuracy of Neural Network is 96.9254% for the circles testing data.
Accuracy of Perceptron is 88.8609% for the circles testing data.
In short, based on the two results on two different datasets, it is clear and

straightforward to see that Neural Network has better performance than a simple Perceptron. For the first dataset which is not linearly separable, Perceptron is not expressive and the accuracy is really bad. However, using hidden layers in Neural Network, we can easily achieve a much better expressiveness.

2. Answer to problem 2 - Multi-class classification

   a. i. For One-vs-All, we learn k classifiers for k classes.
   For All-vs-All, we learn k choose 2 classifiers for k choose 2 pairs of labels.

   ii. For One-vs-All, we need to use all m examples to learn each classifier.
   For All-vs-All, we need $2m/k$ examples since each type of class labels has m/k examples.

   iii. For One-vs-All, decision is made based on Winner Takes All. That is, $f(x) = argmax_i \ {w_i}^T x$, predict the one with the highest score.
   For All-vs-All, each pair of classifier v_{ij} will predict class i if $v_{ij}x > 0$, and predict class j if $v_{ij}x < 0$. Thus, at decision time, a voting scheme is usually applied: all k choose 2 classifiers are applied to the test examples and the class label gets the highest votes will get predicted.

   iv. Based on (i) and (ii), for One-vs-All, the computational complexity is $O(mk)$.
   For All-vs-All, the computational complexity is $k^2 \times \frac{2m}{k} = O(mk)$.

   b. Personally, I would prefer All-vs-All scheme to do the multi-class classification. One-vs-All has linear number of classifiers but All-vs-All requires a quadratic number of classifier. Both schemes have the same computational complexity since All-vs-All uses less examples to learn from than One-vs-All. The main point here is that All-vs-All is more expressive since if a certain label is not linearly separable from the other, One-vs-All cannot be used to give prediction.

   c. If Kernel Perceptron is used for the classification, All-vs-All is undoubtedly preferred due to its improved efficiency. Because One-vs-All has a computational complexity of $O(m^2k)$ and All-vs-All has a computational complexity of $O(m^2)$.

   d. For One-vs-All, we need to learn k classifiers, and m examples are needed to learn each classifier. Therefore, the overall training time complexity is $k \cdot d \cdot m^2 = O(kdm^2)$.
   For All-vs-All, we have in total $k(k-1)/2$ classifiers, and m/k examples needed to learn per positive and per negative example respectively. Therefore, the overall training time complexity is $\frac{k(k-1)}{2} \cdot d \cdot (\frac{2m}{k})^2 = O(dm^2)$.
   In short, All-vs-All is more time-efficient in training.

   e. For One-vs-All, we need to learn k classifiers, and m examples are needed to learn each classifier. Therefore, the overall training time complexity is $k \cdot d^2 \cdot m = O(kd^2m)$.
   For All-vs-All, we have in total $k(k-1)/2$ classifiers, and m/k examples needed to learn per positive and per negative example respectively. Therefore, the overall training time complexity is $\frac{k(k-1)}{2} \cdot d^2 \cdot \frac{2m}{k} = O(kd^2m)$.
   In short, both schemes have the same training time complexity.

f. For Counting, the evaluation time complexity would be $m(m-1)/2 \cdot d = O(m^2 d)$ because we need to evaluate every pair of labels to give the majority vote and there are m choose 2 pairs and each classifier has dimensionality of d.
For Knockout, the evaluation time complexity would be $(m-1) \cdot d = O(md)$ since for m examples we need to have one label beat all other $m-1$ labels.

3. Answer to problem 3 - Probability Review

a. i. Expected number of children
Let X denote the number of the children in one family. For town A, each family will have exactly one child, either a boy or a girl, so that the expected number of children in a family is given by $E[X] = 1 \times \frac{1}{2} + 1 \times \frac{1}{2} = 1$.
For town B, each family will have as many children as it wants until a boy is born. The problem becomes a series of geometric distribution where $P(X = k) = (1-p)^{k-1}p$ where k is the number of children in one family and p equals to 0.5. Therefore, the expected number of children in a family is given by

$$\begin{aligned} E[X] &= 1 \times P(X=1) + 2 \times P(X=2) + 3 \times P(X=3) + ... \\ &= 1 \times 0.5 + 2 \times 0.5^2 + 3 \times 0.5^3 + ... \\ &= \sum_{k=1}^{\infty} k \times 0.5^k \\ &= \frac{1}{0.5} = 2 \end{aligned} \tag{4}$$

which follows the mean of the geometric distribution.

ii. Boy to girl ratio at each generation
Let X and Y denote the number of boys and girls in one family respectively, and m and n be the total number of families in town A and B.
For town A, $E[X] = 0.5 \times m = \frac{m}{2}$ and $E[Y] = 0.5 \times m = \frac{m}{2}$ since the probability of having either a boy or a girl in each one-child family is the same. Therefore, the boy to girl ratio in town A is $E[X] : E[Y] = \frac{m}{2} : \frac{m}{2} = 1 : 1$.
For town B, since each family will have as many as it wants until a boy is born, each family will have exactly one boy so that the number of boys in town B is $E[X] = n$. In addition, each family will have all but one girls, so that the expected number of girls in town B is given by

$$\begin{aligned} E[Y] &= n \times E[number\ of\ girls\ in\ one\ family] \\ &= n \times (1 \times P(x=1) + 2 \times P(x=2) + 3 \times P(x=3) + ...) \\ &= n \times \sum_{k=1}^{\infty} k \times 0.5^{k+1} \\ &= n \end{aligned} \tag{5}$$

Therefore, the boy to girl ratio in town B is $E[X] : E[Y] = n : n = 1 : 1$, the same as that of town A.

b. i. For the left side of the equation,

$$P(A|B) = \frac{P(AB)}{P(B)} \tag{6}$$

For the right side of the equation,

$$\begin{aligned} \frac{P(B|A)P(A)}{P(B)} &= \frac{\frac{P(AB)}{P(A)}P(A)}{P(B)} \\ &= \frac{P(AB)}{P(B)} \end{aligned} \tag{7}$$

Thus, we see that the left side of the equation equals to the right side. We can conclude that

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{8}$$

ii.

$$P(A, B, C) = P(A|B, C) \times P(B, C) = P(A|B, C) \times P(B|C) \times P(C) \tag{9}$$

c.

$$\begin{aligned} E[X] &= 0 \times P(X = 0) + 1 \times P(X = 1) \\ &= 0 \times P(\bar{A}) + 1 \times P(A) = P(A) \end{aligned} \tag{10}$$

d. i. In order to show whether X is independent of Y or not, we need to show that $P(X, Y) \stackrel{?}{=} P(X)P(Y)$. In this case, we know from the table that

$$\begin{aligned} &P(X = 0) = 1/15 + 4/15 + 1/10 + 8/45 = 11/18 \\ &P(Y = 0) = 1/15 + 1/15 + 4/15 + 2/15 = 8/15 \\ &P(X = 0, Y = 0) = 1/15 + 4/15 = 1/3 \\ &P(X = 0, Y = 0) \neq P(X = 0)P(Y = 0) \end{aligned} \tag{11}$$

Therefore, X is not independent of Y.

ii. In order to show whether X is independent of Y given Z, we need to show that $P(X = x|Y = y, Z = z) \stackrel{?}{=} P(X = x|Z = z)$. In this case, we know from the table that

$$\begin{aligned} P(X = 0|Y = 0, Z = 0) &= \frac{1/15}{1/15 + 1/15} = 1/2 \\ P(X = 0|Y = 0, Z = 1) &= \frac{4/15}{4/15 + 2/15} = 2/3 \\ P(X = 0|Y = 1, Z = 0) &= \frac{1/10}{1/10 + 1/10} = 1/2 \\ P(X = 0|Y = 1, Z = 1) &= \frac{8/45}{8/45 + 4/45} = 2/3 \\ P(X = 0|Z = 0) &= \frac{1/15 + 1/10}{1/15 + 1/10 + 1/15 + 1/10} = 1/2 \\ P(X = 0|Z = 1) &= \frac{4/15 + 8/45}{4/15 + 8/45 + 2/15 + 4/45} = 2/3 \end{aligned} \tag{12}$$

Therefore, we can see that

$$\begin{aligned} &P(X=0|Z=0) = P(X=0|Y=0,Z=0) = P(X=0|Y=1,Z=0) = 1/2 \\ &P(X=0|Z=1) = P(X=0|Y=0,Z=1) = P(X=0|Y=1,Z=1) = 2/3 \end{aligned} \tag{13}$$

In short, X is conditionally independent of Y given Z.

iii.

$$\begin{aligned} P(X=0|X+Y>1) &= P(X=0|X+Y\neq 0) \\ &= \frac{1/10+8/45}{1-1/15-4/15} \\ &= 5/12 \end{aligned} \tag{14}$$