10601 Machine Learning HW4

Mingyang Song

(1) Did you *receive* any help whatsoever from anyone in solving this assignment? Yes.

Q3 part c, I discussed with Shangqing Zhang about the meaning and solution.

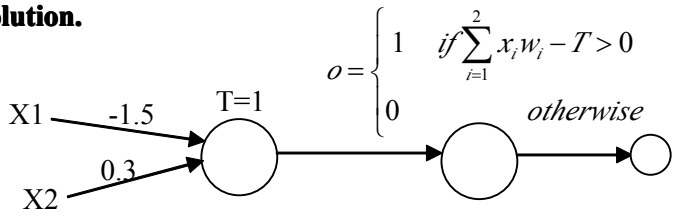
(2) Did you *give* any help whatsoever to anyone in solving this assignment? Yes.

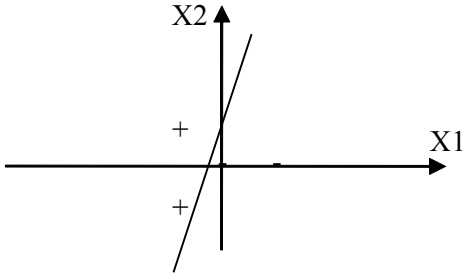
Q3 part c, I discussed with Shangqing Zhang about the meaning and solution.

Q1. Perceptron Networks (32 pt)

(a) Design a single perceptron to correctly classify the above patterns. Specify the weight vector w and the threshold by T. Geometrically illustrate how you derived your solution.

$$o = \begin{cases} 1 & if \sum_{i=1}^{2} x_i w_i - T > 0 \\ 0 & otherwise \end{cases}$$





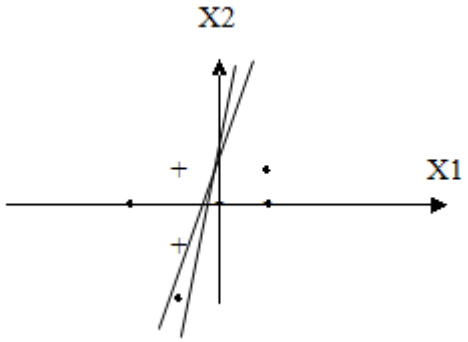
(b) Using the preceptron you derived in Q1(a), classify the following input patterns:

$$\left\{x_5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, y_5 = 1\right\},\ \left\{x_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, y_6 = 0\right\},\ \left\{x_7 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, y_7 = 0\right\},\ \left\{x_8 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, y_8 = 1\right\}$$

(c) Since there are many correct answers to Q1(a), the answers to Q1(b) may in general also differ from student to student. Are there any input patterns in Q1(b) for which only one answer is possible? If so, which and why? If not, why not?

x_5, x_6, x_7 have only one possible answer. x_8 may have different answers.

Geometrically: There can be two lines classifying Q1 data correctly however leaving x_8 different answers. For other 3 input patterns, they must be classified into one answer.



Mathematically:

To correct classify Q1(a), we have

$$-w_1 + w_2 > T$$
$$-w_1 - w_2 > T$$
$$0 < T$$
$$w_1 < T$$

For x_5, $\begin{matrix} -w_1 + w_2 > T \\ -w_1 - w_2 > T \\ -2w_1 > 2T > T \end{matrix}$, y=1; for x_6, $\begin{matrix} 0 < T \\ w_1 + w_2 < 0 < T \end{matrix}$, y=0; for x_7, $w_1 < T$, y=0;

However, we cannot decide x_8.

(d)If we modify our training set to, is it still possible to perform correct classification using a single neuron perceptron? If so, give such a perceptron. If not, explain why not. Answer this problem mathematically (not graphically).

If we can perform a correct classification using single neuron perceptron, we will have the following conclusions:

$$\begin{matrix} -w_1 + w_2 > T \\ -w_1 - w_2 \le T \\ w_1 - w_2 > T \\ w_1 + w_2 \le T \end{matrix} \Rightarrow \begin{matrix} -w_1 + w_2 > T \\ -w_1 - w_2 \le T \\ 2w_2 > 0 \\ w_2 > 0 \end{matrix} \Rightarrow \begin{matrix} w_1 - w_2 > T \\ w_1 + w_2 \le T \\ 2w_2 < 0 \\ w_2 < 0 \end{matrix} \Rightarrow \text{contradiction.}$$

When we calculate w1 or T, we also have contradictions. So that we cannot perform a correct classification using single perceptron.

Q2. Gradient-based Training (20 pt)

Consider the Back-Propagation algorithm operating on neurons which use the transfer function log sigmoid instead of the usual sigmoid function. That is, assume that the output of a single neuron is $o = \log\sigma(w^T x)$. Give the weight update rules for output layer weights and for hidden layer weights. Remember $(\sigma(x))' = \sigma(x)\cdot(1-\sigma(x))$. (20 pt)

The update rule is $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = -\eta \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = -\eta \frac{\partial E_d}{\partial net_j} x_{ij}$.

(1) For output layer,

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j} = -(t_j - o_j)\frac{\partial o_j}{\partial net_j}$$

Since $o = \log\sigma(w^T x)$,

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \log(\sigma(net_j))}{\partial net_j} = \frac{1}{\sigma(net_j)\ln 2}\sigma(net_j)(1-\sigma(net_j)) = \frac{(1-\sigma(net_j))}{\ln 2}$$

Therefore, $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ij} = \eta(t_j - o_j)(1-\sigma(net_j))x_{ij}$ (ln2 can be added into η)

(2) For hidden layer,

$$\frac{\partial E_d}{\partial net_j} = \sum_{k\in Downstream(j)} \frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial net_j} = \frac{-(t_j - o_j)(1-\sigma(net_k))}{\ln 2}\frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$= \frac{-(t_j - o_j)(1-\sigma(net_k))}{\ln 2} w_{kj}\frac{\partial o_j}{\partial net_j} = \frac{-(t_j - o_j)(1-\sigma(net_k))}{\ln 2} w_{kj}\frac{(1-\sigma(net_k))}{\ln 2}$$

$$= \frac{-(t_j - o_j)(1-\sigma(net_k))^2}{(\ln 2)^2} w_{kj}$$

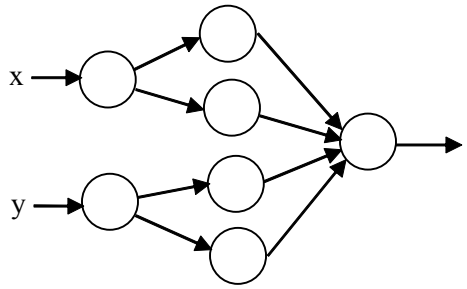
So $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ij} = \eta(t_j - o_j)(1-\sigma(net_j))^2 w_{kj}x_{ij}$ ($(\ln 2)^2$ can be added into η)

Q3.Error functions (28 pt)

Consider the task of learning target concepts that are rectangles in the x, y plane. Each hypothesis is described by the x, y coordinates of the lower left and upper right corners of the rectangle - llx, lly, urx, ury respectively. An instance x, y is labeled positive by the hypothesis < llx, lly, urx, ury > if and only if the point x, y lies inside the rectangle. The goal is to minimize the classification error rate.

(a) Define the appropriate error function E for this setup. (8 pt)

We can build an ANN as follows:



$E = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$ where o_d is the output of the ANN. So when we input x and y, we need to train a hidden layer let it learn the concept that (x, y) in the rectangle(x>llx, x<urx, y>lly, y<ury) gives a positive sigma (net) and this sigma as the input of output layer gives positive result.

For the hidden layer, we must let the model learn "AND" concept.

(b) Can a gradient descent algorithm be used with this error function? Why or why not? (10 pt)

No. As we use output of ANN which is an noncontinuous function, the error function will also be noncontinuous. At this time, we cannot use gradient descent algorithm according to its property.

(c) If the answer to part b is yes, briefly describe the procedure of performing gradient descent. If the answer to part b is no, sketch an approximation to the error function in part a that will make gradient descent possible. (10 pt)

To use gradient descent, we nee to change the perceptron unit into Sigmoid unit.

$o_d = \sigma(net) = \frac{1}{1+e^{-net}}$.

To produce a positive result, we have let x>llx, x<urx, y>lly, y<ury with AND. Then they produce a net>0 for the output of hidden layer. As a result, the o_d >0 and result is positive.

To learn AND concept, we have to give each unit of the for units on hidden layer a same weight and let them adding up to T(>0). Net-T will go into final unit. Than each hidden unit should have a 1/4*T. So that any unit gives a negative result will lead to a negative. And to keep continuous, we still need hidden units to be Sigmoid. And the largest output of Sigmoid is 1, and we can set T to be 1.

From input layer to hidden layer, we can simply use x-lls or urx-x to define the net.

$$o_h = \sigma(net) = \frac{1}{4}\frac{1}{1+e^{-net}}$$

$$net_h = \frac{1}{4}\frac{1}{1+e^{-(x-llx)}} + \frac{1}{4}\frac{1}{1+e^{-(y-lly)}} + \frac{1}{4}\frac{1}{1+e^{-(urx-x)}} + \frac{1}{4}\frac{1}{1+e^{-(urx-x)}} - 1$$

$$o_d = \sigma(net) = \frac{1}{1+e^{-net_h}}$$

$$E = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

So exists $\frac{\partial E}{\partial llx} = (t_d - o_d)\frac{\partial(t_d - o_d)}{\partial llx} = (t_d - o_d)\frac{e^{-net_h}}{(1+e^{-net_h})^2}\frac{\partial(\frac{1}{4}\frac{1}{1+e^{-(x-llx)}})}{\partial llx}$

$$= \frac{1}{4}(t_d - o_d)\frac{e^{-net_h}}{(1+e^{-net_h})^2}\frac{e^{llx-x}}{\left(1+e^{llx-x}\right)^2}$$

Others are the same.

Done.

Q4.Simple Recurrent Network (20 pt)

Consider the sigmoid neuron illustrated below. It has only two inputs x1, x2. x1 is a "feed-back" from the output. Such a neuron can model signal evolution over time, in discrete steps, by implementing a recurrent relation. Specifically, x1 at a given time is set to be the neuron's output from the previous time step, namely:

$$\mathbf{x_1(t+1) = O(t).}$$

Assume that x_2 is constant, namely, it does not change over time.

1. Calculate O(t + 1) as a function of the weights w1, w2 of the neuron (you may assume there is no threshold) (8pt)

As there is no threshold, $O(t+1) = \frac{1}{1+e^{-(x_1(t+1)w_1 + x_2 w_2)}} = \frac{1}{1+e^{-(O(t)w_1 + x_2 w_2)}}$.

2. Derive $\frac{\partial O(t+1)}{\partial w_1}, \frac{\partial O(t+1)}{\partial w_2}$. (12 pt)

Remember $x_1(t+1)$ is the function of w_1, w_2.

$$\frac{\partial O(t+1)}{\partial w_1} = -\frac{1}{\left(1+e^{-(O(t)w_1+x_2w_2)}\right)^2}\cdot(-e^{-(O(t)w_1+x_2w_2)})(\frac{\partial O(t)}{\partial w_1}w_1+O(t)) = \frac{e^{-(O(t)w_1+x_2w_2)}(w_1\frac{\partial O(t)}{\partial w_1}+O(t))}{\left(1+e^{-(O(t)w_1+x_2w_2)}\right)^2}$$

$$\frac{\partial O(t+1)}{\partial w_2} = -\frac{1}{\left(1+e^{-(O(t)w_1+x_2w_2)}\right)^2}\cdot(-e^{-(O(t)w_1+x_2w_2)})(w_1\frac{\partial O(t)}{\partial w_2}+x_2) = \frac{e^{-(O(t)w_1+x_2w_2)}(w_1\frac{\partial O(t)}{\partial w_2}+x_2)}{\left(1+e^{-(O(t)w_1+x_2w_2)}\right)^2}$$

As $x_1(t+1) = O(t)$ is the function of w_1, w_2, we can not remove the $\frac{\partial O(t)}{\partial w}$ factor.