Solutions to Assignment 4: Neural Networks

10601: Machine Learning (Spring 2012)

TA : Antonio Juarez *(ajuarez@andrew.cmu.edu)*

OUT: Feb 22, 2012

DUE: Feb 29, 2012

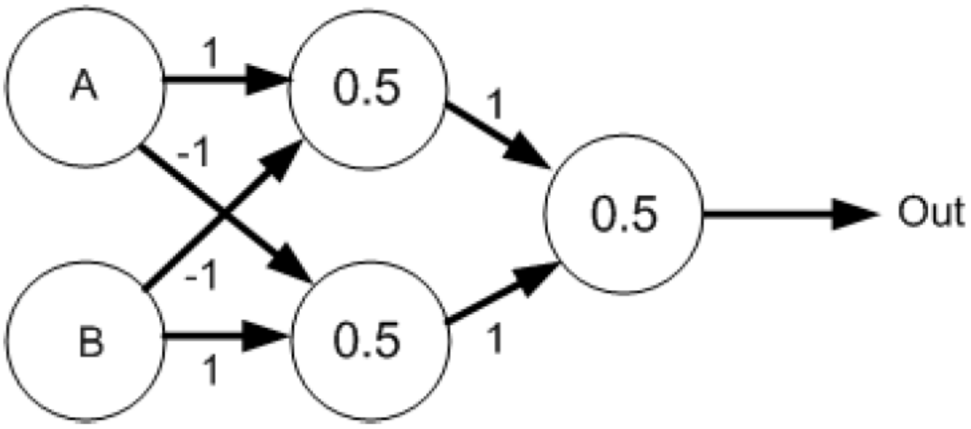
- We prefer typed solutions, but if you cannot do that, please make sure your handwriting is neat and legible. We cannot give credit to any part of the solution that is not legible.
- Answer the questions in the order they are stated.
- Policy on collaboration: Please refer to course website: page 'Policies'.
- Policy on late homework: Please refer to course website: page 'Policies'. If you would like to use one or more of your discretionary extension days, you should mail the TA in charge before the deadline, to let them know.
- For questions and clarifications, contact Antonio (ajuarez@andrew.cmu.edu).
- Be sure to write your Andrew ID and name on the top of **every page**.
- Antonio has regular office hours on Wednesdays, 11:00am-12:00md. He will also be holding extra office hours on Monday Feb 27th, 11:00am-12:00md.

Perceptron Networks (32 pt)

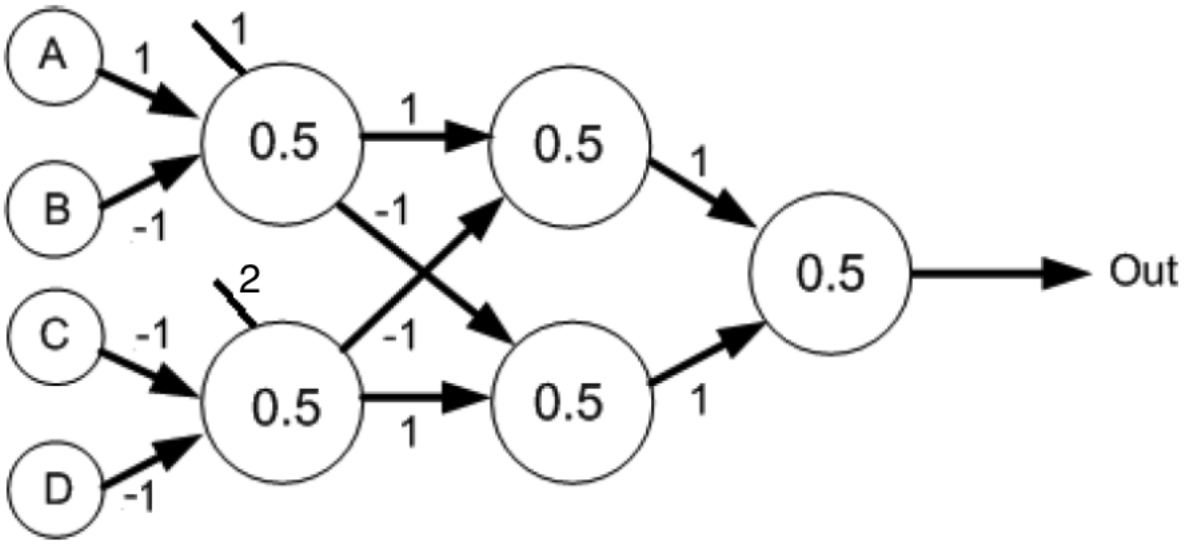
1. For every neural network in the following questions, there should be one input node for every boolean input, and one boolean output node. A boolean is 0 if false and 1 if true. All non-input nodes, including the output node, should have simple 0-1 step-functions as transfer functions. In other words, you should have a perceptron network. Try to keep your weights integer and small though they can be negative. Also try to keep your networks as symmetric as possible - doing so in earlier questions may help with the later questions. Do not draw edges with weight 0. In case you are not familiar with logic notation: $\vee$ means "OR", $\wedge$ means "AND", $\neg$ means "NOT", and $\oplus$ means "XOR".

   In each diagram below, the value inside the node is the threshold of the step function.

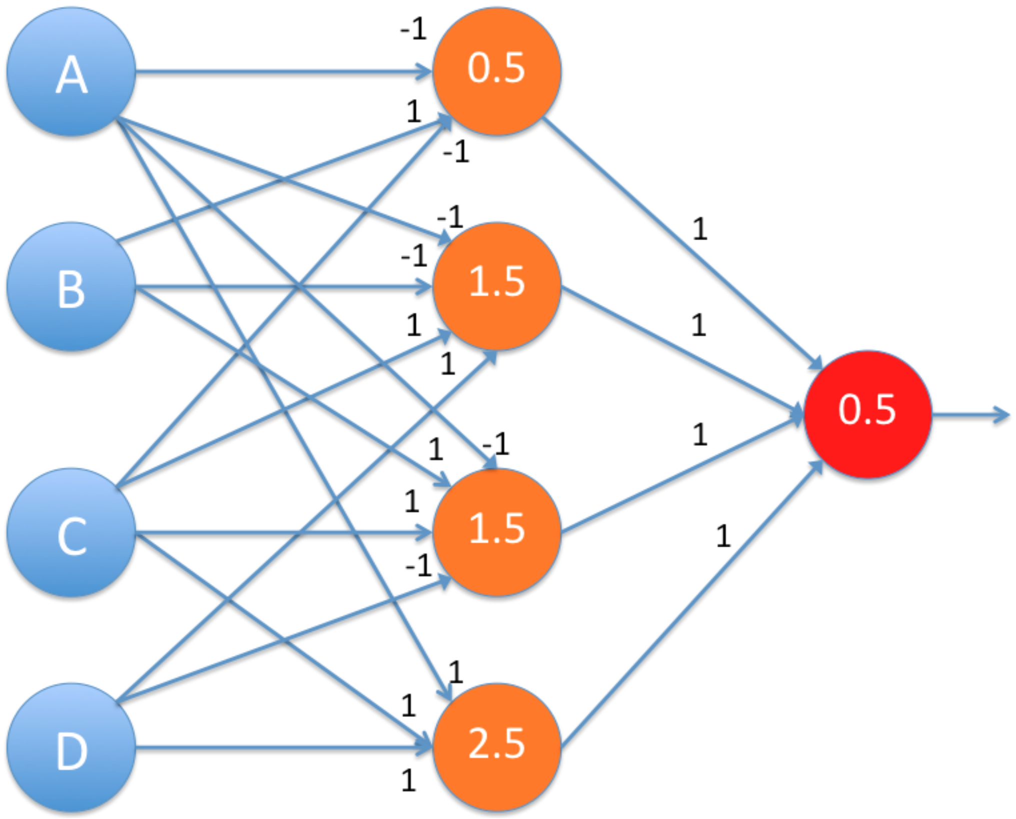
   (a) Create (draw) a neural network with one hidden layer of two nodes that implements $A \oplus B$. (8 pt)
   Solution



   (b) Using your neural network from $Q1(A)$, create (draw) a neural network with two hidden layers that implements $(A \vee \neg B) \oplus (\neg C \vee \neg D)$. (8 pt)
   Solution



   (c) Create (draw) a neural network with only one hidden layer (of any number of nodes) that implements $(A \vee \neg B) \oplus (\neg C \vee \neg D)$. (8 pt)
   Solution



The function $(A \vee \neg B) \oplus (\neg C \vee \neg D)$ can be expressed as: $ACD \vee \overline{A}B\overline{C} \vee \bar{A}\bar{B}CD \vee \overline{A}BC\overline{D}$. Each of the nodes in the hidden layer represents one of these conjunctions, and the disjunction of all of these is computed in the single output node.

(d) Say you do not care how many nodes are in your hidden layer, but you do not want to have more than one hidden layer. Can you represent ANY Boolean function (n Boolean inputs, 1 boolean output) this way? Please explain. (8 pt)

Solution *Yes, any boolean function can be represented as a disjunction of conjunctions (or a conjunction of disjunctions). And this can be represented in an ANN using one hidden node for each conjunction/disjunction, with the output node implementing a conjunction/disjunction of the hidden layer outputs (as in the previous problem).*

Gradient-based Training (40 pt)

1. Imagine the Back-Propagation algorithm operates on units using the transfer function *tanh* instead of the sigmoid function. That is, assume that the output of a single unit is $o = tanh\left(\overrightarrow{w}.x\right)$. Give the weight update rules for output layer weights and hidden layer weights. Remember $tanh'(x) = 1 - tanh^2(x)$. (20 pt)

   Solution

   Output Layer

$$E_d\left(\overrightarrow{w}\right) = \frac{1}{2} \sum_{k \in outputs} \left(t_k - o_k\right)^2$$

$$\Delta w_{ji} = -\eta \frac{\delta E_d\left(\overrightarrow{w}\right)}{w_{ji}}$$

$$\frac{\delta E_d\left(\overrightarrow{w}\right)}{w_{ji}} = \frac{\delta E_d\left(\overrightarrow{w}\right)}{\delta o_j} \frac{\delta o_j}{\delta net_j} \frac{\delta net_j}{\delta w_{ji}}$$

$$\frac{\delta E_d\left(\overrightarrow{w}\right)}{w_{ji}} = -\left(t_k - o_k\right)\left(1 - tanh^2\left(net_j\right)\right) x_{ji}$$

   Therefore $\Delta w_{ji} = \eta\left(t_k - o_k\right)\left(1 - tanh^2\left(net_j\right)\right) x_{ji}$

Hidden Layer

$$\begin{aligned}
\frac{E_d\left(\overrightarrow{w}\right)}{w_{ji}} &= \left(\sum_{k\in Downstream(j)} \frac{\delta E_d\left(\overrightarrow{w}\right)}{\delta net_k}\frac{\delta net_k}{\delta net_j}\right)\frac{\delta net_j}{\delta w_{ji}} \\
&= \left(\sum_{k\in Downstream(j)} -\delta_k \frac{\delta net_k}{\delta net_j}\right) x_{ji} \\
&= \left(\sum_{k\in Downstream(j)} -\delta_k \frac{\delta net_k}{\delta o_j}\frac{\delta o_j}{\delta net_j}\right) x_{ji} \\
&= \left(\sum_{k\in Downstream(j)} -\delta_k w_{kj}\frac{\delta o_j}{\delta net_j}\right) x_{ji} \\
&= \left(\sum_{k\in Downstream(j)} -\delta_k w_{kj}\left(1 - tanh^2\left(net_j\right)\right)\right) x_{ji}
\end{aligned}$$

Therefore $\Delta w_{ji} = \eta\left(1 - tanh^2\left(net_j\right)\right) x_{ji} \sum_k \delta_k w_{kj}$.

2. Consider the alternative error function for training a network:

$$E(\overrightarrow{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}\left(t_{kd} - o_{kd}\right)^2 + \gamma\sum_{i,j} w_{ji}^2$$

Derive the gradient descent update rule for this definition of E. Use the sigmoid as the transfer function. (20 pt)

Solution

Output layer

$$\begin{aligned}
E_d\left(\overrightarrow{w}\right) &= \frac{1}{2}\sum_{k\in outputs}\left(t_k - o_k\right)^2 + \gamma\sum_{i,j} w_{ji}^2 \\
\frac{\delta E_d\left(\overrightarrow{w}\right)}{w_{ji}} &= -\left(t_k - o_k\right) o_j\left(1 - o_j\right) x_{ji} + 2\gamma w_{ji}
\end{aligned}$$

Therefore:

$$\Delta w_{ji} = \eta\left(t_k - o_k\right) o_j\left(1 - o_j\right) x_{ji} - 2\gamma\eta w_{ji}$$

Similarly for the hidden layer:

$$\Delta w_{ji} = \eta o_j\left(1 - o_j\right) x_{ji} \textstyle\sum_k \delta_k w_{kj} - w\gamma\eta w_{ji}$$

Error functions (28 pt)

1. 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)
   Solution

$$E_{dataset}\left(llx, lly, urx, ury\right) = \frac{1}{2}\sum_{x_i, y_i\in dataset}\left(t\left(x_i, y_i\right) - o\left(x_i, y_i\right)\right)^2$$

where $t(x, y)$ *is the correct label for the point* (1 *if positive,* 0 *if negative), and*

$$o(x, y) = g_s(x_i - llx)\, g_s(y_i - lly)\, g_s(urx - x_i)\, g_s(ury - y_i)$$

which will be 1 *if inside the hypothesis rectangle, and 0 otherwise.* $g_s(x) = 0$ *is the step function:*

$$g(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

The product of the step functions is 1 if the point is inside the rectangle and 0 otherwise.

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

Solution *No, the function is not differentiable because of the step functions. Gradient descent algorithms cannot therefore be used to optimize this error function.*

(c) If the answer to part *b* is yes, briefly describe the procedure of performing gradient descent. If the answer of part *b* is no, can you suggest an approximation to the error function in part *a* so that you can apply gradient descent with your new error function? (10 pt)

Solution *We can use a sigmoid function to approximate the step function* g_s.