The University of Texas at Dallas

CS 6375.001

Machine Learning

Assignment 1:

Neural Networks

Student:
Reg Gonzalez
rdg170330

Instructor:
Anurag Nagar

Due Date: March 20, 2024

Assignment 1 - Theoretical Part

1.1 Revisiting Backpropagation Algorithm

a) Error function: $E_d(w) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$

Weight delta: $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$, where η represents the learning rate

Now let's try to compute $\frac{\partial E_d}{\partial w_{ji}} \rightarrow \frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$.

This talks about two things: 1.) how the net of node *j*, which is the input *before* the activation function (i.e., $\sum_i w_{ji} x_{ji}$), affects the error ($\frac{\partial E_d}{\partial net_j}$) and 2.) how the weight of the connection between nodes *i* and *j* affects the net of node j ($\frac{\partial net_j}{\partial w_{ji}}$).

To compute $\frac{\partial net_j}{\partial w_{ji}}$ (the second term) is easy: $\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$. To compute $\frac{\partial E_d}{\partial net_j}$ (the first term) is more challenging because there are two cases you need to keep in mind: node *j* could either be an output unit or a hidden unit.

Let's assume that *j* is an output unit: $\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$. Let's focus on the first term in that equation first (i.e., $\frac{\partial E_d}{\partial o_j}$). We know the Error function (defined above), so when we differentiate E_d with respect to o_j, we get: $\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} [\frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2]$. This can be simplified to $\frac{\partial E_d}{\partial o_j} = -(t_j - o_j)$. Great, now we've got the first term! Now let's move on to the second term in the equation (i.e., $\frac{\partial o_j}{\partial net_{ji}}$). Because we are using the **Tanh activation function** here, the derivative of that function is $f'(x) = 1 - tanh(x)^2$. That derivative is the second term. So, when we put it all together, if node *j* is an output unit, then $\frac{\partial E_d}{\partial net_{ji}} = -(t_j - o_j)(1 - tanh(x)^2)$. To simplify this going further, we can just say that $-(t_j - o_j)(1 - tanh(x)^2) = -\delta_j$, so $\frac{\partial E_d}{\partial net_{ji}} = -\delta_j$.

Assignment 1 - Theoretical Part

Now, let's plug this into the equation for weight delta: $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \rightarrow$

$\Delta w_{ji} = \eta(t_j - o_j)(1 - tanh(x)^2)x_{ji} = \eta\delta_j x_{ji}$.

Remember, that was just the first case (when node *j* is an output unit). Now we have to consider when *j* is a part of the hidden layer. Again, we're looking to find $\frac{\partial E_d}{\partial net_j}$. However, since *j* is a part of the hidden layer, the equation to solve is a bit different. Now it's $\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$. Look at the 2nd term (i.e., $\frac{\partial net_k}{\partial o_j}$). That's easy to compute: it's just w_{kj}. Look at the 3rd term (i.e., $\frac{\partial o_j}{\partial net_j}$). That was already done above. It's the derivative of the Tanh(x) activation function: $f'(x) = 1 - tanh(x)^2$. We can rewrite the equation as $\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k w_{ji}(1 - tanh(x)^2)$. This can be rewritten as $\delta_j = (1 - tanh(x)^2) \sum_{k \in Downstream(j)} \delta_k w_{ji}$. Like before, we'll just simplify all of that to $\frac{\partial E_d}{\partial net_j} = \delta_j$.

Again, let's plug this into the equation for weight delta: $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta\delta_j x_{ji}$

To summarize:

- If node *j* is an output node: $\frac{\partial E_d}{\partial net_j} = -\delta_j = -(t_j - o_j)(1 - tanh(x)^2)$
- If node *j* is a hidden node: $\frac{\partial E_d}{\partial net_j} = \delta_j = (1 - tanh(x)^2) \sum_{k \in Downstream(j)} \delta_k w_{ji}$
- The weight delta (also called weight change) is: $\Delta w_{ji} = \eta\delta_j x_{ji}$

b) The derivation of the backpropagation algorithm is the same as part a), the only difference here is that we're using the ReLu activation function instead of the Tanh activation function.

The derivation of the ReLu activation function is $f'(x) = \begin{cases} 0, \, for \, x < 0 \\ 1, \, for \, x > 0 \end{cases}$

Assignment 1 - Theoretical Part

So for simplicity, I'll just write the final equations:

- If node *j* is an output node: $\frac{\partial E_d}{\partial net_j} = \delta_j = -(t_j - o_j)\frac{\partial o_j}{\partial net_j}$, where $\frac{\partial o_j}{\partial net_j} = \begin{cases} 1, \ if \ net_j > 0 \\ 0, \ otherwise \end{cases}$

- If node *j* is a hidden node: $\frac{\partial E_d}{\partial net_j} = \delta_j = \sum_{k \in Downstream(j)} \delta_k w_{ji} \frac{\partial o_j}{\partial net_j}$, where $\frac{\partial o_j}{\partial net_j} = \begin{cases} 1, \ if \ net_j > 0 \\ 0, \ otherwise \end{cases}$

- The weight delta (also called weight change) is: $\Delta w_{ji} = \eta \delta_j x_{ji}$

Assignment 1 - Theoretical Part

1.2 Gradient Descent

Output *o* is defined as: $o = w_0 + w_1(x_1 + x_i^2) + \ldots + w_n(x_n + x_n^2)$

The goal is to learn w_i's that minimize the squared error function, which is defined as:

$E[\vec{w}] = \frac{1}{2}\sum_{d\epsilon D}(t_d - o_d)^2$, where *D* is a set of training examples provided to the neural network.

The gradient is defined as: $\nabla E[\vec{w}] = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}]$

The training rule for updating the weights vector is: $\Delta\vec{w} = -\eta\nabla E[\vec{w}]$ or $\Delta w_i = -\eta\frac{\partial E}{\partial w_i}$

The derivation of gradient descent proceeds as follows:

$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \rightarrow$

$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \rightarrow$

$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \rightarrow$

$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot(\vec{x}_d + \vec{x}_d^{\,2})) \rightarrow$

$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-(\vec{x}_d + \vec{x}_d^{\,2}))$

This derivation follows the same way as the regular gradient descent derivation; the difference is how *o* is defined.

When we plug $\frac{\partial E}{\partial w_i}$ into the equation for Δw_i, we get:

$\Delta w_i = -\eta\frac{\partial E}{\partial w_i} = \eta\sum_{d\epsilon D}(t_d - o_d)(x_{id} + x_{id}^2)$,

Assignment 1 - Theoretical Part

where η is the learning rate, *D* is the set of training examples, t_d is the target output for training example *d*, o_d is the neural network's output for training example *d*, and x_{id} is the value of the ith attribute for training example *d*.

Since Δw_i just represents the change in the weights vector, that's not the final result of the weight update. The final result would be adding this change in the weights vector to the old weight to get the new weight. Mathematically, that means:

$$w_i = w_i + \Delta w_i = w_i + \eta \sum_{d \epsilon D} (t_d - o_d)(x_{id} + x_{id}^2)$$

Assignment 1 - Theoretical Part

1.3 Comparing Activation Functions

a) Let's denote $y_1, y_2, y_3, y_4, and\ y_5$ as the nodes 1, 2, 3, 4, and 5 respectively in Figure 1. $x_1\ and\ x_2$ are the inputs to the neural network while $h(x)$ represents the activation function—as per the problem description.

The output of $y_3 = h(w_{31}x_1 + w_{32}x_2)$ and the output of $y_4 = h(w_{41}x_1 + w_{42}x_2)$.

Similarly, the output of $y_5 = h(w_{53}y_3 + w_{54}y_4)$. If we substitute $y_3\ and\ y_4$ using their outputs defined above, we get: $y = h(w_{53} * h(w_{31}x_1 + w_{32}x_2) + w_{54} * h(w_{41}x_1 + w_{42}x_2))$

b) As per the problem description, we are now using vector notation to represent the output of the neural network. The vectors are defined below:

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$W^{(1)} = \begin{pmatrix} w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{5,3} & w_{5,4} \end{pmatrix}$$

To represent the output of the hidden layer, we'll denote that with $HL^{(1)}$. Using vector notation, the output of the hidden layer can be represented as: $HL^{(1)} = h(W^{(1)}X)$. This is the same as combining the outputs of $y_3\ and\ y_4$ from PART 1.3a. This time we're multiplying the vectors of $X\ and\ W^{(1)}$ together instead of multiplying them out separately.

Likewise, let's denote the output of the output layer as Y. Using vector notation, the output of the output layer can be represented as: $Y = h(W^{(2)}HL^{(1)})$. Again, we're doing the same thing we did in PART 1.3a, just with vector notation instead. If we substitute $HL^{(1)}$ using its output defined earlier, we get $Y = h(W^{(2)}h(W^{(1)}X))$.

Assignment 1 - Theoretical Part

c) We have two choices for activation functions $h(x)$: Sigmoid and Tanh. The two activation functions are defined as:

Sigmoid:

$$h_s(x) = \frac{1}{1+e^{-x}}$$

Tanh:

$$h_t(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

We want to show that the neural networks created using the above activation functions generate the same function.

To show this, we want to compute the relationship between $h_s(x)$ and $h_t(x)$, showing that they differ only by linear transformations and constants.

First, let's compute e^{-x} using the Sigmoid function:

$h_s(x) = \frac{1}{1+e^{-x}} \rightarrow 1 + e^{-x} = \frac{1}{h_s(x)} \rightarrow e^{-x} = \frac{1}{h_s(x)} - 1$

Using e^{-x}, let's compute e^{x}:

$e^{-x} = \frac{1}{h_s(x)} - 1 \rightarrow \frac{1}{e^x} = \frac{1}{h_s(x)} - 1 \rightarrow 1 = (\frac{1}{h_s(x)} - 1) * e^x \rightarrow$

$e^x = \frac{1}{\frac{1}{h_s(x)}-1}$

Now that we have the values for e^{-x} and e^{x}, let's plug them into the Tanh activation function formula:

$h_t(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{(\frac{1}{1/h_s(x)-1}) - (1/h_s(x) - 1)}{(\frac{1}{1/h_s(x)-1}) + (1/h_s(x) - 1)} \rightarrow$ To simplify this, let $a = \frac{1}{h_s(x)-1} \rightarrow$

$h_t(x) = \frac{\frac{1}{a} - a}{\frac{1}{a} + a} = \frac{1-a^2}{1+a^2} \rightarrow$ Substitute a back with the original value $\rightarrow$

$h_t(x) = \frac{1 - (\frac{1}{h_s(x)} - 1)^2}{1 + (\frac{1}{h_s(x)} - 1)^2}.$

Here, we've shown the relationship between $h_s(x)$ *and* $h_t(x)$. One can be written as a function of another—differing by linear linear transformations and constants.

Assignment 1 - Theoretical Part

Going back to the original problem, we want to show that neural networks using the two different activation functions generate the same function.

As a reminder from PART 1.3b, the output of the output layer is defined as:
$Y = h(W^{(2)}h(W^{(1)}X))$.

If we want to provide the specific activation functions (i.e., Sigmoid and Tanh), then we could denote the output as the following:
$Y_s = h_s(W^{(2)}h_s(W^{(1)}X))$ and $Y_t = h_t(W^{(2)}h_t(W^{(1)}X))$,
where Y_s represents the output using Sigmoid and Y_t represents the output using Tanh.

Because we've shown that there is a relationship between $h_s(x)\ and\ h_t(x)$, more specifically that $h_t(x)$ can be written as a function of $h_s(x)$, we can say that $Y_s\ and\ Y_t$ are essentially the same. Y_t can just be written using a function of Y_s.