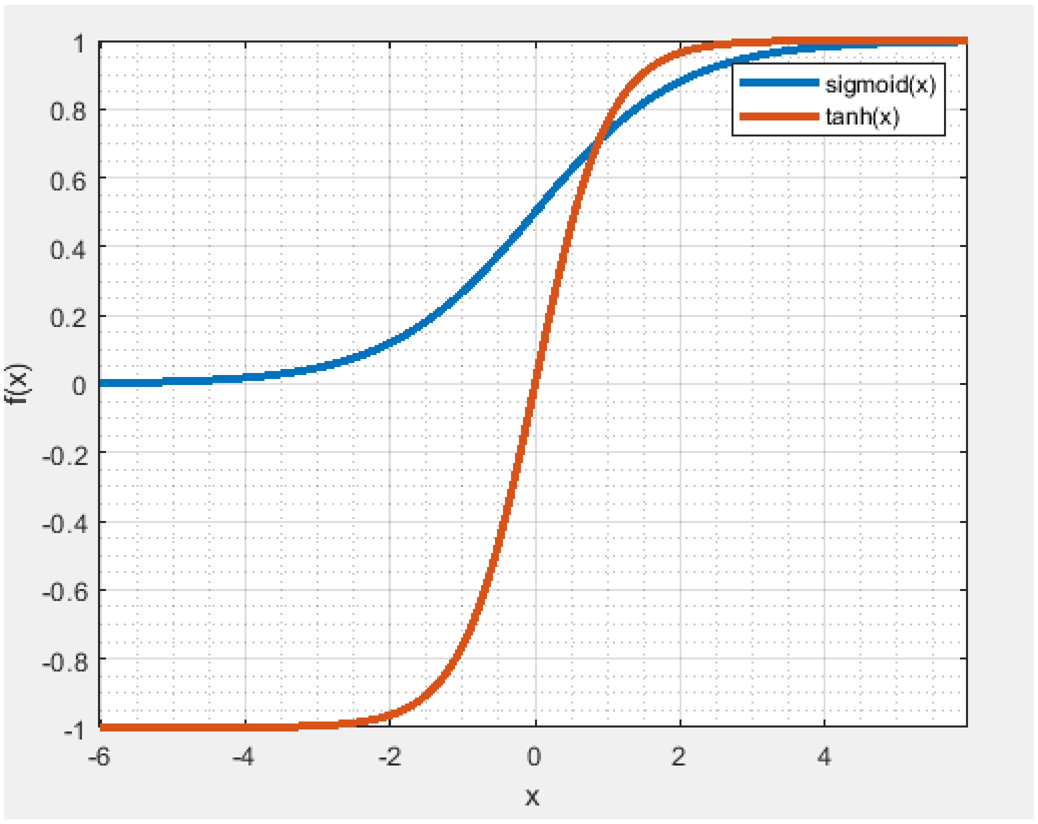
Fall 2020: CSCI 4/5588 Written Homework # 1

Name: Tharani Maaneeivaannan and ID:2575198

PART (A)

Task #1: Draw the graphs of a sigmoid function and a hyperbolic tangent function, overlapping each other, with an *x*-axis ranging from -6 to +6. Describe the differences between them.

The graph of sigmoid function and hyperbolic tangent function overlapping each other with x-axis ranging from -6 to +6 is given below,



Program (Matlab) for generating sigmoid and hyperbolic function:

```
>>x=-6.0:0.1:6.0;

>> y=(1./(1+exp(-x)));

>> plot(x,y,'Linewidth',3)
>> hold on;

>> x=-6.0:0.1:6.0;

>> y=((exp(x)-exp(-x))./((exp(x)+exp(-x))));

>> plot(x,y,'Linewidth',3)

>> grid on

>>grid minor
```

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve.

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

Sigmoid functions has a return value (y axis) in the range 0 to 1. A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point and exactly one inflection point. It is monotonic and has a first derivative which is bell shaped. Sigmoid function is especially used for models where to predict the probability as an output because probability exists between the range of 0 and 1. Sigmoid functions have been used as the activation function of artificial neurons.

tanh function is also like logistic sigmoid and the range of the tanh function is from (-1 to 1). tanh is a scaled sigmoid function. tanh is also sigmoidal (s - shaped). tanh is monotonic and differentiable.

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

While a sigmoid function will map input values between 0 and 1, tanh will map values between -1 and 1. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. In sigmoid function, towards either end of the sigmoid function, the y values tend to respond very less to changes in x values. This means the gradient in that region is small.

Both tanh and logistic sigmoid activation functions are used in feed-forward nets. Sigmoid function is still very popular in classification problems. But the gradient is stronger for tanh function than sigmoid function. Depending on the requirement of gradient strength, the sigmoid function or tanh function is chosen.

PART (B)

Task#2: In chapter 04 (ANN) from pages 10 to 13, we have derived equations from (i) to (vi) for backpropagation where we assumed our activation function was a sigmoid function. Now, in this assignment, similarly, workout the detailed derivation and show the new form of the equations: (i) to (vi) (if any) when the activation function is assumed to be a hyperbolic tangent function, where x is a variable. Note again, your detailed answer is expected.

The output error is defined as,

$$\mathrm{E}(\beta) = \frac{1}{2}\sum_{k=1}^{K}\left(Y_k - Z_k^{(L)}\right)^2$$

layer =1, 2, …,L and L = Output Layer. Y_k is the k^{th}target/original output and $Z_k^{(L)}$ is the k^{th} predicted/network output.
The back-propagation learning rule is based on gradient descent. This is because we want to minimize the error by changing the weights. The weights are initialized with random values, and then they are changed in a direction that will reduce the error:

$$\Delta\beta = -\alpha\frac{\partial \mathrm{E}}{\partial \beta} \quad \ldots\ldots\ldots\ldots\text{(i)}$$

where $\beta(\mathrm{t}+1) = \beta(\mathrm{t}) - \alpha\frac{\partial \mathrm{E}}{\partial \beta}$ or $\Delta\beta = [\beta(\mathrm{t}+1) - \beta(\mathrm{t})] = -\alpha\frac{\partial \mathrm{E}}{\partial \beta}$ and α is the learning rate. The overall idea of the error back-propagation will be to compute the discrepancy of the targeted output versus the predicted output from output node(s) first and then to propagate backward to compute the discrepancy for each of the nodes in the hidden layer except the input layer – because we do not want to change the input layer. Then update the weight or the parameter β for the network to minimize the error. First, we will compute the rate of changes of error with respect to the weight connected to the output node(s): Differentiating $\mathrm{E}(\beta)$ with respect to β,

$$\frac{\partial \mathrm{E}}{\partial \beta^{(L-1)}} = \frac{\partial}{\partial \beta^{(L-1)}}\left[\frac{1}{2}\sum_{k=1}^{K}\left(Y_k - Z_k^{(L)}\right)^2\right]$$

Using chain rule,

$$= \frac{\partial}{\partial\left(Y_k - Z_k^{(L)}\right)}\left[\frac{1}{2}\sum_{k=1}^{K}\left(Y_k - Z_k^{(L)}\right)^2\right]\frac{\partial}{\partial\beta^{(L-1)}}\left(Y_k - Z_k^{(L)}\right)$$

[Removing $\sum$ because all (L-1)th weights may not be connected to all the output nodes.]

$$= \frac{1}{2} \times 2\left(Y_k - Z_k^{(L)}\right).\frac{\partial}{\partial\beta^{(L-1)}}(-Z_k^{(L)})$$

$$= \left(Z_k^{(L)} - Y_k\right).\frac{\partial}{\partial\beta^{(L-1)}}(\sigma T_k^{(L)})$$

Substituting $Z_k^{(L)} = \sigma T_k^{(L)}$

Using chain rule,

$$= \left(Z_k^{(L)} - Y_k\right).\frac{\partial}{\partial T_k^{(L)}}(\sigma T_k^{(L)}) \times \frac{\partial T_k^{(L)}}{\partial\beta^{(L-1)}}$$

Note:

$\mathrm{Z} = \sigma T = f_{tanh}(\mathrm{T}) = \dfrac{e^T - e^{-T}}{e^T + e^{-T}}$

$Z' = \dfrac{d(\sigma T)}{dT} = \dfrac{d}{dT}(\dfrac{e^T - e^{-T}}{e^T + e^{-T}})$

Using chain rule,

$$= \frac{(e^T + e^{-T}).(e^T - (-e^{-T}) - (e^T + (-e^{-T}).(e^T - e^{-T})}{(e^T + e^{-T})^2}$$

$$= \frac{(e^T + e^{-T})^2 - (e^T - e^{-T})^2}{(e^T + e^{-T})^2}$$

$$= \frac{(e^T + e^{-T})^2}{(e^T + e^{-T})^2} - \frac{(e^T - e^{-T})^2}{(e^T + e^{-T})^2}$$

$$= 1 - \left[\frac{e^T - e^{-T}}{e^T + e^{-T}}\right]^2$$

$$Z' = 1 - Z^2 = 1 - (\sigma T)^2$$

Therefore, $\frac{\partial}{\partial T^{(L)}}(\sigma T^{(L)}) = Z^{(L)'} = 1 - (Z^{(L)})^2$ ………………………………(ii)

$$= \left(Z_k^{(L)} - Y_k\right) Z_k{}^{(L)'} \times \frac{\partial T_k^{(L)}}{\partial \beta^{(L-1)}}$$

$$= \left(Z_k^{(L)} - Y_k\right) Z_k{}^{(L)'} \times \frac{\partial}{\partial \beta^{(L-1)}} \left(\beta^{(L-1)^T} Z^{(L-1)}\right) \; [\because T_k^{(L)} = \beta^{(L-1)^T} Z^{(L-1)}]$$

$$= \left(Z_k^{(L)} - Y_k\right) Z_k{}^{(L)'} Z^{(L-1)}$$

Assume the error term for the output layer, $\delta^{(L)} = \left(Z_k^{(L)} - Y_k\right) Z_k{}^{(L)'} = \left(Z_k^{(L)} - Y_k\right)(1 - (Z_k{}^{(L)})^2) = \delta^{(L)} Z^{(L-1)}$
Finally, for output layer we have,

$$\frac{\partial \mathrm{E}}{\partial \beta^{(L-1)}} = \delta^{(L)} Z^{(L-1)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iii)}$$

where $\delta^{(L)} = \left(Z_k^{(L)} - Y_k\right)(1 - (Z_k{}^{(L)})^2$
Let us now compute the rate of changes of error with respect to the weight for the hidden layer(s) similarly:
Differentiating $\mathrm{E}(\beta)$ with respect to $\beta^{(L-2)}$,

$$\frac{\partial \mathrm{E}}{\partial \beta^{(L-2)}} = \frac{\partial}{\partial \beta^{(L-2)}} \left[\frac{1}{2} \sum_{k=1}^{K} \left(Y_k - Z_k^{(L)}\right)^2 \right]$$

$$= \sum_{k=1}^{K} \left(Y_k - Z_k^{(L)}\right) \times \frac{\partial}{\partial \beta^{(L-2)}} \left(-Z_k^{(L)}\right)$$

Substituting $Z_k^{(L)} = \sigma T_k^{(L)}$

$$= \sum_{k=1}^{K} \left(Z_k^{(L)} - Y_k\right) \times \frac{\partial}{\partial \beta^{(L-2)}} \left(\sigma T_k^{(L)}\right)$$

Using chain rule,

$$= \sum_{k=1}^{K} \left(Z_k^{(L)} - Y_k\right) \times \frac{\partial}{\partial T_k^{(L)}} \left(\sigma T_k^{(L)}\right) \times \frac{\partial T_k^{(L)}}{\partial \beta^{(L-2)}}$$

$$= \sum_{k=1}^{K} \left(Z_k^{(L)} - Y_k\right) \times Z_k{}^{(L)'} \times \frac{\partial T_k^{(L)}}{\partial \beta^{(L-2)}}$$

Using chain rule,

$$= \sum_{k=1}^{K} \left(Z_k^{(L)} - Y_k\right) \times Z_k{}^{(L)'} \times \frac{\partial T_k^{(L)}}{\partial Z^{(L-1)}} \times \frac{\partial Z^{(L-1)}}{\partial \beta^{(L-2)}}$$

$$\left[\because T_k^{(L)} = \beta^{(L-1)^T} Z^{(L-1)} \;\; \therefore \frac{\partial T_k^{(L)}}{\partial Z^{(L-1)}} = \beta^{(L-1)}\right]$$

$$= \sum_{k=1}^{K} \left(Z_k^{(L)} - Y_k\right) \times {Z_k}^{(L)'} \times \beta^{(L-1)} \times \frac{\partial Z^{(L-1)}}{\partial \beta^{(L-2)}}$$

$$= \frac{\partial Z^{(L-1)}}{\partial \beta^{(L-2)}} \times \sum_{k=1}^{K} \delta^{(L)} \; \beta^{(L-1)} \left[\therefore \delta^{(L)} = \left(Z_k^{(L)} - Y_k\right) {Z_k}^{(L)'}\right]$$

To find $\frac{\partial Z^{(L-1)}}{\partial \beta^{(L-2)}}$,

$$\frac{\partial Z^{(L-1)}}{\partial \beta^{(L-2)}} = \frac{\partial}{\partial \beta^{(L-2)}} (\sigma T^{(L-1)}) = \frac{\partial (\sigma T^{(L-1)})}{\partial T^{(L-1)}} \times \frac{\partial T^{(L-1)}}{\partial \beta^{(L-2)}}$$

$${Z_k}^{(L-1)'} = \frac{\partial (\sigma T^{(L-1)})}{\partial T^{(L-1)}} = 1 - (Z^{(L-1)})^2$$

$$= 1 - (Z^{(L-1)})^2 \times \frac{\partial T^{(L-1)}}{\partial \beta^{(L-2)}}$$

$$= 1 - (Z^{(L-1)})^2 \times \; Z^{(L-2)} \left[\because \frac{\partial T^{(L-1)}}{\partial \beta^{(L-2)}} = \frac{\partial}{\partial \beta^{(L-2)}} \left(\beta^{(L-2)^T} Z^{(L-2)}\right) = \; Z^{(L-2)}\right]$$

$$= \; 1 - (Z^{(L-1)})^2 \times \; Z^{(L-2)} \times \sum_{k=1}^{K} \delta^{(L)} \times \; \beta^{(L-1)}$$

$$= \; Z^{(L-2)} \times 1 - (Z^{(L-1)})^2 \times \sum_{k=1}^{K} \delta^{(L)} \times \; \beta^{(L-1)}$$

Finally, for the hidden layer we have,

$$\frac{\partial \mathrm{E}}{\partial \beta^{(L-2)}} = Z^{(L-2)} \times 1 - (Z^{(L-1)})^2 \times \sum_{k=1}^{K} \delta^{(L)} \; \times \beta^{(L-1)}$$

Or we can write it as

$$\frac{\partial \mathrm{E}}{\partial \beta^{(L-2)}} = \delta^{(L-1)} Z^{(L-2)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iv)}$$

where $\delta^{(L-1)} = 1 - (Z^{(L-1)})^2 \times \sum_{k=1}^{K} \delta^{(L)} \times \; \beta^{(L-1)}$

Involvements of the Bias term:

If we use Equation (iii) for the bias term at layer (L-1), we can rewrite equation (iii) as:

$$\frac{\partial \mathrm{E}}{\partial {\beta_0}^{(L-1)}} = \delta^{(L)} Z^{(L-1)} = \delta^{(L)} [\because {Z_0}^{(L-1)} = 1] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(v)}$$

Similarly, for the other hidden layers following equation (iv) and the idea of equation (v), we can write:

$$\frac{\partial \mathrm{E}}{\partial {\beta_0}^{(L-2)}} = \delta^{(L-1)} {Z_0}^{(L-2)} = \delta^{(L-1)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(vi)}$$

Now, we have generated the required equation to describe the back-propagation algorithm.

Algorithm 1: Error Backpropagation

```
BEGIN
    1. From a data point (x_i, y_i), apply an input vector x_i to the network and forward
       propagate through the network (as we computed the logistic regression).
    2. For each of the output node/unit compute the error term $\delta^{(L)}$:
           $\delta^{(L)} = \left(Z_k^{(L)} - Y_k\right)(1 - ({Z_k}^{(L)})^2)$
    3. For each of the hidden layer node /unit compute the error term $\delta^{(L-1)}$:
           $\delta^{(L-1)} = 1 - (Z^{(L-1)})^2 \times \sum_{k=1}^{K} \delta^{(L)} \times \beta^{(L-1)}$
       And Compute $\delta^{(L-2)}, \delta^{(L-3)} \ldots.$ until $\delta^{(2)}. Except\ \delta^{(1)}$ because it is the input layer
       and we do not want to change the original input data.
    4. Update the weights ( $\beta$ ) of the network
           $\beta^{(i)}(t+1) = \beta^{(i)}(t) - \alpha\delta^{(i+1)} Z^{(i)}$ [For non-bias term]
           ${\beta_0}^{(i)}(t+1) = {\beta_0}^{(i)}(t) - \alpha\delta^{(i+1)}$ [For bias terms]
       where, i=1, 2, …, (L-1)
END
```