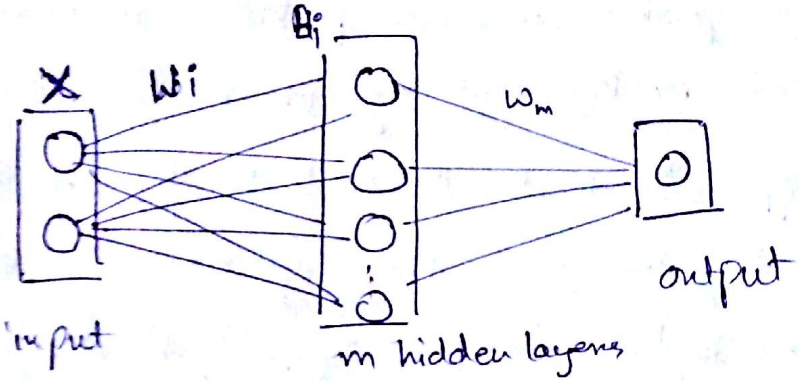
Theory Questions

Submitted by: Himanshu Aggarwal
(MT17015)

(1.) A neural net of arbitrary depth using just linear activation functions can't be used to model the XOR table. ~~since~~ XOR function is a non-linear function, and a neural net with linear activation function is nothing but a linear classifier, which can't be used to model XOR table.



Let W_i & B_i denote weight & bias associated with the hidden layers in the above network with m hidden layers. Let f_i be the linear activation functions with each hidden layer.

Input to first hidden layer is, $= X \cdot W_1$

$$f_1(X \cdot W_1) = k \cdot X \cdot W_1 + B_1$$

where k is some constant.

Now this above value is the input to the next layer. Similar as above, the final output of the neural network will be of the form, $(k' X \cdot W^* + B^*)$

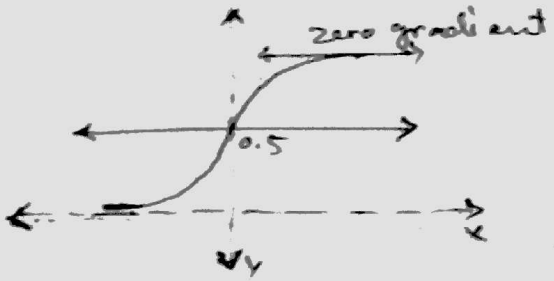
where W^* is the product of weights over the layers & B^* is the product of weights & biases over the layers.

So, it can be seen through the output that this neural net with linear activation function produces a linear product of input & and weights.

This result is similar to the results produced by any linear classifier covered during lectures, like linear regression or SVM.

(2.) The reason possible for such problem is the Vanishing Gradient problem of sigmoid function.

What happens is that the constant slop of sigmoid function is reached during the training (due to large values going as input in the sigmoid function. This leads to zero gradient & and hence no updates in the weights is possible thereafter.



Using ReLU will not garantee the correct traing of the model, since even if it doesn't show vanishing gradient problem, it may lead to data explosion. And in case of negative weights, there will be no update in the weights.

Preprocessing Technique for Sigmoid

→ Scaling the input by a some factor.
(As can be seen in this assignment)

Preprocessing technique for ReLU

→ Usage of Leaky ReLU
→ Normalization of data.

(3.) The cost function (Qudratic) is, $C = \frac{(y-a)^2}{2}$

Partial Derivative of cost function wrt w & b is as

$$\frac{\partial C}{\partial w} = (a-y)\,\sigma'(z)\,x$$

$$\frac{\partial C}{\partial b} = (a-y)\,\sigma'(z)$$

Where $\sigma'(z)$ is the derivative of sigmoid function.
When z, neurons output, is close to 1,
$\sigma'(z)$ gets very small.
So equation ① & ② become very small.
This is vanishing gradient problem.

Cross entropy solves this issue by changing the cost function to $C = y\ln(a) + (1-y)\ln(1-a)$

Partial derivative of C wrt w gives,

$$\frac{\partial C}{\partial w} = \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w}$$

$$= \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right) \sigma'(z) \cdot x$$

$$= \frac{\sigma'(z) \cdot x}{\sigma(z)(1-\sigma(z))} (\sigma(z) - y)$$

$$\frac{\partial C}{\partial w} = (\sigma(z) - y) x$$

Similarly $\frac{\partial C}{\partial b} = (\sigma(z) - y)$

Since the term $\sigma'(z)$ is getting canceled in the derivate of cross entropy cost function wrt w and b, learning slowdown problem is no more.