①

ASSIGNMENT 2: DEEP LEARNING

Pranjali Pathre
2019112002.

Question 1: Neural Networks.

1. When training a deep neural network with gradient descent, the partial derivatives traversing the network from the final layer to initial layer, using chain rule, undergo matrix multiplications to compute their derivatives.

Due to matrix multiplications, if the derivatives are small, then the gradients will decrease exponentially as we propage through the model until it eventually vanishes, which is vanishing gradient problem.

Similarly, in exploding gradient problem, if the derivatives are large, then the gradient will increase exponentially as we propagate down the model until they eventually explode, which might cause model to be unstable due to large change in model weights.

2.

ReLU is defined as $h = max(0, a)$.

Advantages of using ReLU:-

1. Reduced likelihood of gradient to vanish:
In RELU gradients have a constant value whereas sigmoids becomes increasingly small as the absolute value of x increases. The constant gradient of ReLUs result in faster learning.

2. Sparsity:
It arises when $a < 0$. ReLU generates sparse representations. On the other hand sigmoid are always likely to generate some non-zero values resulting in dense representations.

3. Slower runtime:
ReLU has much slower runtime than sigmoid function. ReLU converges faster than sigmoid function.

3.

No. of input units = 2.

Hidden layer = 1

Hidden units = 2.

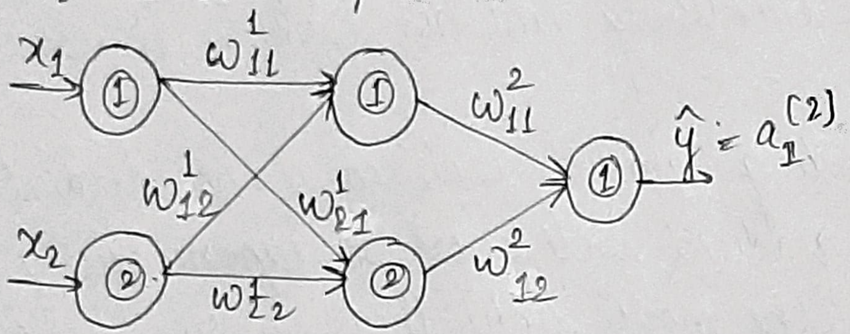
Output unit = 1.

Activation function for hidden units = Tanh.

Activation function for output unit = sigmoid.

Assuming bias terms to be zero.

Let x_1 & x_2 be the inputs.



Let $W^{[1]} = \begin{bmatrix} \\ \end{bmatrix}$

We have,

$z_1^{[1]} = w_1^{[1]T} x$, Here x is the input vector, $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ & $w_1^{[1]} = \begin{bmatrix} w_{11}^{[1]} \\ w_{12}^{[1]} \end{bmatrix}$

$= [w_{11}^{[1]} \;\; w_{12}^{[1]}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$z_1^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2$.

$z_2^{[1]} = [w_{21}^{[1]} \;\; w_{22}^{[1]}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = w_2^{T[1]} X.$, $w_2^{[1]} = \begin{bmatrix} w_{21}^{[1]} \\ w_{22}^{[1]} \end{bmatrix}$

$= w_{21}^{[1]} x_1 + w_{22}^{[1]} x_2$.

$a_1^{[1]} = g(z_1^{[1]}) = \dfrac{e^{z_1^{[1]}} - e^{-z_1^{[1]}}}{e^{z_1^{[1]}} + e^{-z_1^{[1]}}}$, $a_2^{[1]} = \sigma(z_2^{[1]}) = \dfrac{e^{z_2^{[1]}} - e^{-z_2^{[1]}}}{e^{z_2^{[1]}} - e^{-z_2^{[1]}}}$

③

$$z_2^{[2]} = w_1^{[2]^T} a^{[1]}$$

So, $z_1^{[2]} = w_1^{[2]^T} a^{[1]}$, Here $w_1^{[2]} = \begin{bmatrix} w_{11}^{[2]} \\ w_{12}^{[2]} \end{bmatrix}$ & $a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix}$.

$$\therefore z_1^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix}$$

$$= w_{11}^{[2]} a_1^{[1]} + w_{12}^{[2]} a_2^{[1]}.$$

$$a_1^{[2]} = \sigma(z_1^{[2]}) = \frac{1}{1+e^{-z_1^{[2]}}}.$$

& $\hat{y} = a_1^{[2]}$.

For sigmoid activation function, $g(z) = \frac{1}{1+e^{-z}}$.

$$\therefore g'(z) = \frac{1}{1+e^{-z}}\left[1 - \frac{1}{1+e^{-z}}\right] = g(z)[1-g(z)].$$

For tanh activation function, $g(z) = \frac{e^{+z} - e^{-z}}{e^{z}+e^{-z}}$.

$$g'(z) = 1 - \left(\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\right)^2 = 1 - g^2(z)$$

Cost function: $J = \frac{1}{m} \sum_{i=1}^{n} L(\hat{y}_i, y_i)$, $L(\hat{y}_i, y_i) = (y_i - \hat{y}_i)^2$

Gradient Descent:

$dW^{[1]} = \frac{\partial J}{\partial W^{[1]}}$, $W^{[1]} := W^{[1]} - \alpha\, dW^{[1]}$, α is the learning rate.

$dW^{[2]} = \frac{\partial J}{\partial W^{[2]}}$, $W^{[2]} := W^{[2]} - \alpha\, dW^{[2]}$

Forward Propagation:

$Z^{[1]} = W^{[1]}X$, $Z_2 = W^{[2]}A^{[1]}$

$A^{[1]} = g^{[1]}(Z^{[1]})$, $A^{[2]} = g^{[2]}(Z_2^{[2]})$

Back Propagation.

$$\frac{\partial \mathcal{L}}{\partial a_1^{[2]}} = \frac{\partial}{\partial a}[y-a_1^{[2]}]^2 = -2[y-a_1^{[2]}].$$

$$a_1^{[2]} = \frac{1}{1+e^{-z_1[2]}} \quad , \quad \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} = a_1^{[2]}(1-a_1^{[2]}).$$

$$\therefore \frac{\partial \mathcal{L}}{\partial a_1^{[2]}} = -2[y-a_1^{[2]}] \;\&\; \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} = a_1^{[2]}(1-a_1^{[2]}).$$

$$\therefore \frac{\partial L}{\partial z_1^{[2]}} = \frac{\partial L}{\partial a_1^{[2]}} \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} = -2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]}).$$

$$\therefore dz_1^{[2]} = -2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]}).$$

$$\frac{\partial L}{\partial w_{11}^{[2]}} = \frac{\partial L}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} = -2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]}).\,a_1^{[1]}.$$

$$\frac{\partial L}{\partial w_{12}^{[2]}} = \frac{\partial L}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial w_{12}^{[2]}} = -2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]}).\,a_2^{[1]}$$

$$\therefore dw_{11}^{[2]} = -2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]})a_1^{[1]}$$
$$dw_{12}^{[2]} = -2(y-a_1^{[2]}).a_1^{[2]}(1-a_1^{[2]})a_2^{[1]}.$$

$\left\{ dw_{11}^{[2]} \,\&\, dw_{12}^{[2]} \right.$ are averaged over all the training examples $\left.\right\}$

$$\therefore w_{11}^{[2]} = w_{11}^{[2]} - \alpha dw_{11}^{[2]} = w_{11}^{[2]} + \alpha(2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]})a_1^{[1]}).$$
$$w_{12}^{[2]} = w_{12}^{[2]} - \alpha dw_{12}^{[2]} = w_{11}^{[2]} + \alpha[2(y-a_1^{[2]})a_1^{[2]}(1-a_1^{[2]})a_2^{[1]}).$$

$$\frac{\partial L}{\partial z^{[1]}} = dz^{[1]} = w^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]}).$$

$$\therefore dz_1^{[1]} = \frac{\partial L}{\partial z_1^{[1]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial a_1^{[1]}} \cdot \frac{\partial a_1^{[1]}}{\partial z_1^{[1]}} ,$$
$$= dz_1^{[2]}.w_{11}^{[2]}.(1-a_1^{2[1]}) \qquad a_1^{[2]} = \tanh^2(z_1^{[1]}).$$

$$dz_2^{[1]} = \frac{\partial L}{\partial z_2^{[1]}} = \frac{\partial L}{\partial a_1^{[2]}} \cdot \frac{\partial a_1^{[2]}}{\partial z_1^{[2]}} \cdot \frac{\partial z_1^{[2]}}{\partial a_2^{[1]}} \cdot \frac{\partial a_2^{[1]}}{\partial z_2^{[1]}}$$
$$= dz_1^{[2]}.w_{12}^{[2]}(1-a_2^{2[1]}) \quad , \quad a_2^{2}[1] = \tanh^2(z_2^{[1]})$$

$$\therefore dw_{11}^{[1]} = \frac{\partial \mathcal{L}}{\partial z_1^{[1]}} \cdot \frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}} = dz_1^{[1]} \cdot x_1.$$

$$dw_{12}^{[1]} = dz_p^{[1]} \cdot x_2.$$

$$dw_{21}^{[1]} = dz_2^{[1]} \cdot x_1$$

$$dw_{22}^{[1]} = dz_2^{[1]} \cdot x_2.$$

Here $dw^{[1]}$ are averaged over all the training examples.

$$\therefore w_{11}^{[1]} := w_{11}^{[1]} - \alpha dw_{11}^{[1]} = w_{11}^{[1]} - \alpha dz_1^{[1]} \cdot x_1.$$

$$w_{12}^{[1]} := w_{12}^{[1]} - \alpha dw_{12}^{[1]} = w_{12}^{[1]} - \alpha dz_1^{[1]} x_2.$$

$$w_{21}^{[1]} := w_{21}^{[1]} - \alpha dw_{21}^{[1]} = w_{21}^{[1]} - \alpha dz_2^{[1]} \cdot x_1$$

$$w_{22}^{[1]} := w_{22}^{[1]} - \alpha dw_{22}^{[1]} = w_{22}^{[1]} - \alpha dz_2^{[1]} \cdot x_2.$$

4. The gradient of loss function with respect to weights & bias in each layer can be written as,

$$\frac{\partial \mathcal{L}}{\partial w_{ik}^{[l]}} = \delta_i^{[l]} a_k^{[l-1]}, \qquad \delta_i^{[l]} = \sum_{k=1}^{n^{[l+1]}} w_{ki}^{[l+1]} \delta_k^{[l+1]} (g^{[l]})' (z_i^{[l]})$$

$$\delta_i^{[L]} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial z_i^{[L]}}.$$

$$\frac{\partial \mathcal{L}}{\partial b_i^{[l]}} = \delta_i^{[l]}.$$

gradient descent,

$$w_{ij}^{[l]} := w_{ij}^{[l]} - \alpha \frac{\partial J}{\partial w_{ij}^{[l]}}, \quad b_i^{[l]} := b_i^{[l]} - \alpha \frac{\partial J}{\partial b_j^{[l]}}.$$

Case 1: If we initialize all weights to zero.

We have, $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$ (In vectorised implementation)

& $\frac{\partial \mathcal{L}}{\partial w_{ik}^{[l]}}$ will have same values for zero initialization of weights.

and therefore same update will be performed for all the weights in a particular which ~~can~~ is the equivalent to learning same features by all the neurons/nodes which might cause the network model to saturate.

Case 2: Random Initialization of weights to one

Now, if we initialize all the weights to 1 in a layer, $w_{ij}^{[l]} = 1$, each step of gradient descent, the weights in each layer will be same for all neurons and this network will behave similar to one with having only one hidden unit in each layer which will have limited learning capacity. as we also use same activation function for all the neurons in a layer.

Case 3: Random initialization of weights.

Random initialization of weights will help to overcome the symmetry and no update issue faced when we initialize weights to one & zero respectively.

Question 2: Convolutional Neural Network.

1. Size of previous layer = J×K.

Filter size = M×N.

Stride length = S.

Padding = P.

Let the dimension of the new layer be X×Y.

$$X = \frac{n_H + 2p - f}{S} + 1 = \frac{J+2P-M}{S} + 1$$

$$Y = \frac{K+2P-N}{S} + 1.$$

∴ Size of resulting convolutional layer will be $\left(\frac{J+2P-M}{S}+1\right) \times \left(\frac{K+2P-N}{S}+1\right)$

2. Max pooling filter size = F×F

Stride = S.

Input layer size = J×K.

Let the output layer size be X×Y.

$$\therefore X = \frac{J-F}{S}+1 \text{ \& } Y = \frac{K-F}{S}+1.$$

∴ The size of the downsampled layer will be $\left(\frac{J-F}{S}+1\right) \times \left(\frac{K-F}{S}+1\right)$.

3. A CNN can be converted to a fully connecting by setting number of neurons on output layer to be equal to no. of filters ~~using~~ used in CNN, where the each filter size is equal to that of input.