Deep Learning Home Work 1

Meet Oswal(mo2532)

19 February 2024

Q1.

a)

$$f(x) = \begin{cases} h & \text{if } 0 < x < \delta \\ 0 & \text{otherwise} \end{cases}$$

A Simple Neural Network with 2 hidden neurons can realize the above function using a step activation on the hidden neurons. Below is the network architecture: where X is the input

w1, w2, w3 and w4 are the weights connecting input to hidden neurons and output neuron.

b1, b2, b3 are the 3 biases.

$z1 = \sigma(w1.X + b1)$

$z2 = \sigma(w2.X + b2)$

$z3 = w3.z1 + w4.z2 + b3$

in above equations σ represents unit step function.

The values for all variables which when input to the Neural Networks can realize the orignal box function are:

b3 = -h

w3 = w4 = h

w1 = 1, w2 = -1

b1 = -1, b2 = (δ - 1)

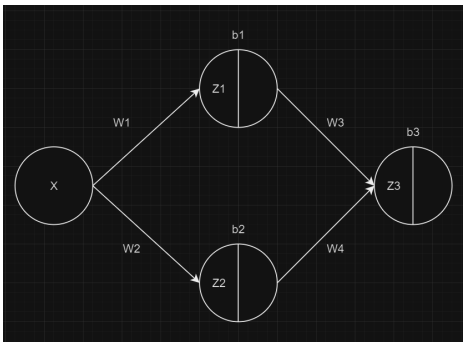

Figure 1: 1.a NN Architecture

For any given value of X we can solve the box function using these above values as parameters.

b) For any function f(x) defined over interval [-B, B] can be approximated by a neural network with one hidden layer because of flexibility derived by the neural network. Hidden layer with enough neurons can capture complex patterns within the inputs. By adjusting the parameters like number of neurons, weights and biases we can approximate f(x) with high precision. As shown in sub-answer (a), we can divide the interval [-B, B] into multiple segments and let them be represented by multiple hidden neurons and using activation function like step activation, weights and biases we can capture complex functions. For example: we divided the condition $0 < x < \delta$ into 2 segments: $x > 0$ and $\delta - x > 0$ to capture the step function f(x) in question 1.a

c) Yes, the argument in b can be extended to a case of d-dimensional input, where input is a vector X:
- Neural Network with hidden layers can learn complex patterns in input data like text or audio as each neuron in the hidden layer can catch different simple pattern / feature, allowing neural networks to perform on complex dataset with high-dimensional data.
- Neural network can handle data with varying dimensions , which means a neural network can handle d-dimensional dataset input as input vector and model its function f(x).
Practical Issues:
- Dimensionality: Increase in input vector dimension can increase the computation required by the network exponentially which increases the resources like machinery cost and time for training and testing.
- Overfitting: More complex dataset requires more neurons to capture complex features, but this can lead to overfitting of the model on the training dataset and not generalize well.
- Interpret-ability: Small networks can be understood by humans but large networks have complex internal working and no human can understand this working of model or how it is generating the output. Hence in case of fin tuning the model it is impossible for humans to directly do it.

Q2.

$$y = softmax(z)$$

$$softmax(z_i) = \frac{e^{z_i}}{\Sigma_{j=0}^{N} e^{z_j}}$$

$$y_i = \frac{e^{z_i}}{\Sigma_{j=0}^{N} e^{z_j}}$$

$$\log y_i = \log\left(\frac{e^{z_i}}{\Sigma_{j=0}^{N} e^{z_j}}\right)$$

$$\log y_i = \log e^{z_i} - \log \Sigma_{j=0}^{N} e^{z_j}$$

$$\frac{\partial \log y_i}{\partial z_j} = \frac{\partial z_i}{\partial z_j} - \frac{\partial \log \Sigma_{j=0}^{N} e^{z_j}}{\partial z_j}$$

$$\frac{\partial \log y_i}{\partial z_j} = \frac{\partial z_i}{\partial z_j} - \frac{e^{z_j}}{\Sigma_{j=0}^{N} e^{z_j}}$$

$$\frac{\partial \log y_i}{\partial z_j} = \frac{\partial z_i}{\partial z_j} - softmax(z_j)$$

$$\frac{\partial \log y_i}{\partial z_j} = \frac{\partial z_i}{\partial z_j} - y_j$$

$\frac{\partial z_i}{\partial z_j} = \delta_{ij}$ because as per given condition $\frac{\partial z_i}{\partial z_j} = 1$ when i != j

$$\frac{1}{y_i}\frac{\partial y_i}{\partial z_j} = \delta_{ij} - y_j$$

$$\frac{\partial y_i}{\partial z_j} = y_i(\delta_{ij} - y_j)$$

Hence Proved.