Q1. Emperical Risk Minimization (ERM):-

→ Emperical Risk Minimization is used to the reduce the generalization error, this quantity is referred to as risk.

→ We replace true probability $p(x,y)$ with emperical probability $\hat{p}(x,y)$.

$$\mathbb{E}_{x,y \in \hat{p}(x,y)} [L(f(x;\theta), y)] = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)})$$

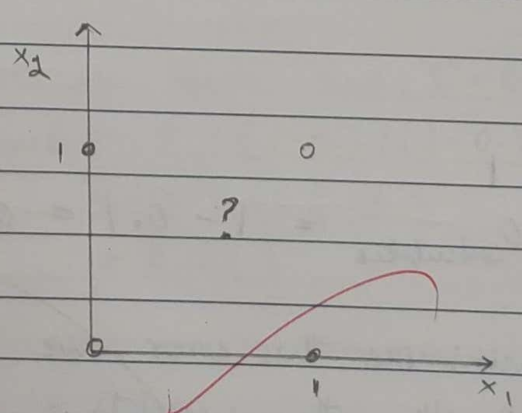
2

where $L \rightarrow$ per-example loss function.

→ It's disadvantage is that, it is prone to overfitting.

Q.2. XOR logic problem:

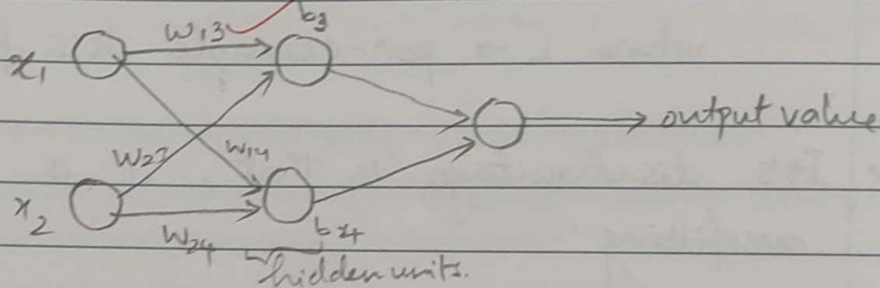
→



x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

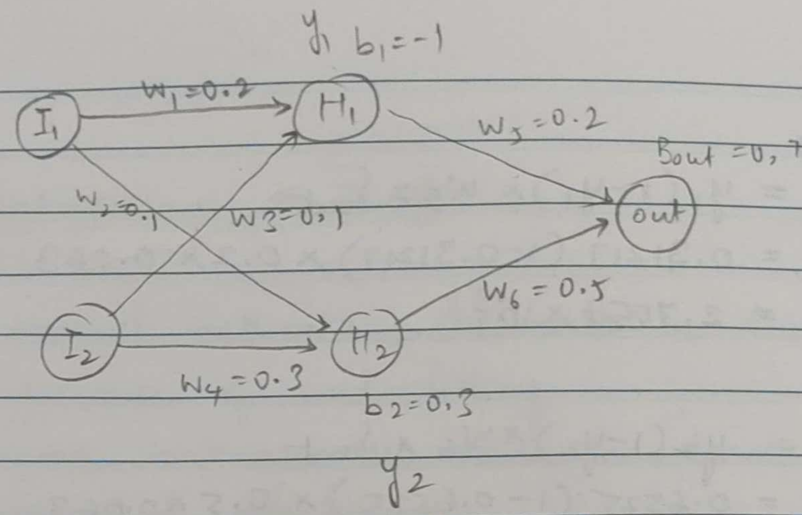
→ As we can see, we get an output value,
when $x_1 = 0$ and x_2 increases
$x_1 = 1$ and x_2 decreases.
Hence, these problems can't be solved by any linear model or the single layer perceptron.

→ The solution is to use a feedforward network with 2 hidden units to solve XOR logic problem.



Q.4. Given, $\eta = 0.5$.
$I_1 = 0.6$.
$I_2 = 0.9$.
Target output = 1
$error = O_{target} - O_{calculated} = 1 - 0.7 = 0.3$.

Now, inorder to minimize this error, we perform backpropagation and alter the weights.



let $a_1 = I_1 w_1 + I_2 w_3 = 0.218$

$a_1' = b_1 + a_1$

$= -1 + 0.218 = \cancel{0.82} = 0.79$

$$y_1 = \frac{1}{1+e^{-a_1'}} = \frac{1}{1+e^{0.79}} = \frac{1}{1+2.2035} = \cancel{0.228034}$$

$$= 0.31217$$

$a_2 = I_1 w_2 + I_2 w_4 = 0.6 \times 0.1 + 0.9 \times 0.3 = 0.33$

$a_2' = b_2 + a_2$

$= 0.3 + 0.33 = 0.63.$

$$y_2 = \frac{1}{1+e^{-a_2'}} = \frac{1}{1+e^{-0.63}} = \frac{1}{1.5325} = 0.6525$$

Now, we have error, $\delta' = 0.3$.

$\delta_{out} = 0.7(1-0.7)(1-0.7)$

$= 0.7 \times 0.3 \times 0.3$

$= 0.063.$

$$\delta_{H1} = y_1(1-y_1) \times W_5 \times \delta_{out}$$
$$= 0.31217\,(1-0.31217) \times 0.2 \times 0.063$$
$$= 2.7054 \times 10^{-3}$$

$$\delta_{H_2} = y_2(1-y_2) \times W_6 \times \delta_{out}.$$
$$= 0.6525\,(1-0.6525) \times 0.5 \times 0.063$$
$$= 7.1424 \times 10^{-3}.$$

$$\Delta W_5 = \eta \times \delta_{out} \times y_1$$
$$= 0.5 \times 0.063 \times 0.31217 = 9.83 \times 10^{-3} = 0.00983$$

$$\Delta W_6 = \eta \times \delta_{out} \times y_2$$
$$= 0.5 \times 0.063 \times 0.6525 = 0.02055$$

$$\Delta W_4 = \eta \times \delta_{H_2} \times I_2$$
$$= 0.5 \times 7.1424 \times 10^{-3} \times 0.9$$

2½

$$\Delta W_3 = \eta \times \delta_{H_1} \times I_2$$
$$= 0.5 \times 2.7054 \times 10^{-3} \times 0.9$$

$$\Delta W_2 = \eta \times \delta_{H_2} \times I_1$$
$$= 0.5 \times 7.1424 \times 10^{-3} \times 0.6$$

$$\Delta W_1 = \eta \times \delta_{H_1} \times I_1$$
$$= 0.5 \times 2.7054 \times 10^{-3} \times 0.9.$$

$$W_{1\,new} = W_{1\,old} + \Delta W_1$$

?

Q. 5. ➤ Bias-variance trade-off.

→ In neural networks, decrease in bias, leads to the problem of underfitting.

→ Whereas, increase in variance, leads to the problem of overfitting.

→ Thus, we need a good balance with decrease variance - increase bias such that the error will be minimum.

Q. 6. The activation functions:

i) tanh, $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

and

ii) Rectified-linear unit : passes only positive values.

ReLU, $ReLU(x) = max(0, x)$.

→ even the negative values pass with value 0, hence they are prone to vanishing gradients.

Q.7. Ill- conditioning:

→ The ill-conditioning in neural network training occurs in Hessian matrix H.

→ It slows down the training process even though it has a strong gradient.

1/2

→ This can be manifested by SGD (stochastic gradient descent) but stuck up because every change in parameter increases the cost.

Q.8.

→ Independent Component Analysis (ICA) separates the multivariate signals into independent, non-gaussian signals.

→ Principal Component Analysis (PCA) helps in dimensionality reduction of the data by keeping the important information intact.

→ Example: Consider a house party, and 2 people are speaking and there is a single mic.

1/2

Now, this recording has mixed voices, ICA separates them into 2 separated voices.

→ PCA handles only linear data and cannot function well on nonlinear data or skewed distribution.

→ Hence, ICA handles non-Gaussian signals with non-uniform data, which has non-uniform, mean & variance values.

Q.3. Activation vol. size = $13 \times 13 \times 64$
filter size = $3 \times 3 \times 64$.

for input matrix : $n \times n$.
padding : p
filter : $f \times f$.
output matrix : $(n+2p-f+1) \times (n+2p-f+1)$.

As, the third dimension, is same for both input matrix and output matrix. we ~~can~~ check with first 2 dimensions.

2 ⇒ for convolution with stride 2;
we can perform with stride 2, because we can have a maximum of 4 strides possible for a 3×3 filter with 13×13 input matrix.

$$\left\lfloor \frac{13}{3} \right\rfloor = 4.$$

→ even for stride 3, we can perform convolution.
→ But for stride 5, we cannot be able to perform convolution because we will be missing few pixels/input values.

→ In convolution, we flip the input matrix and multiply the values in filter matrix with input matrix.

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$