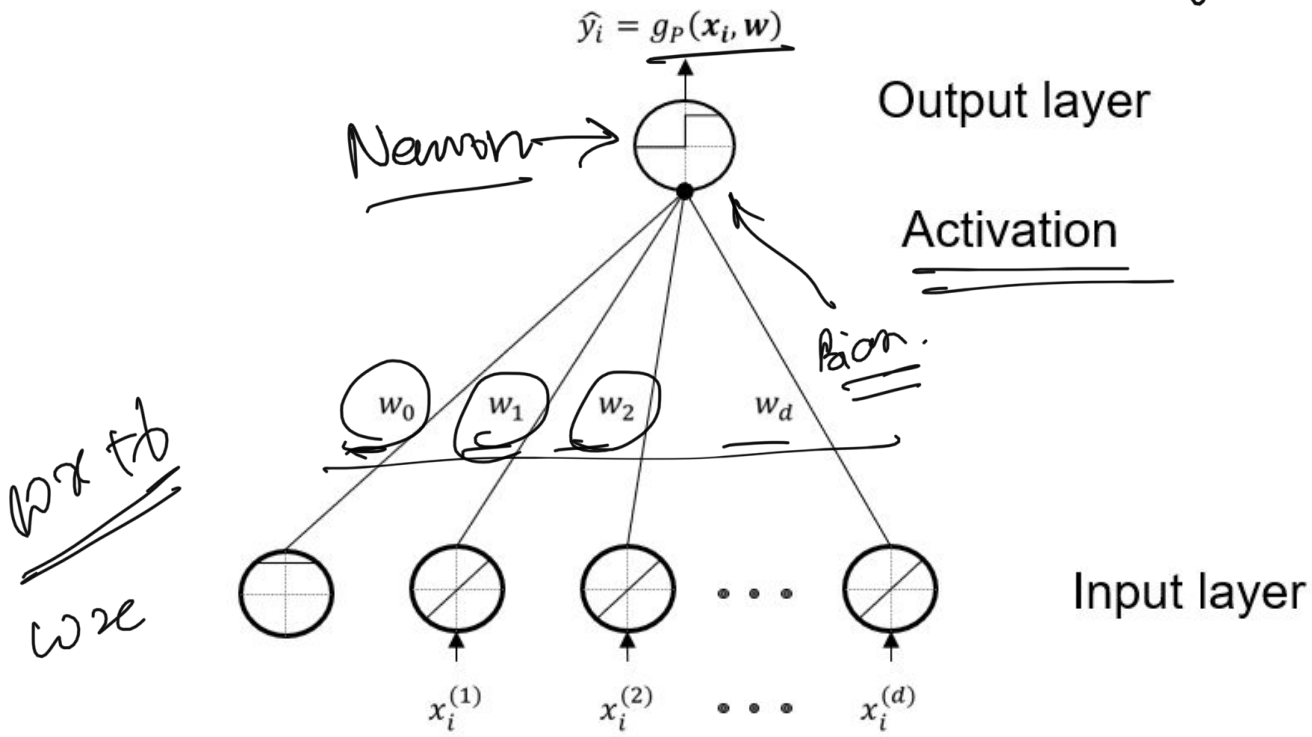
CSE 575: Statistical Machine Learning (Spring 2021)
Instructor: Nupur Thakur

Neural Networks & Deep Learning

Table of contents

What is a Perceptron?

linear binary classifier.



Learning in Perceptron

Iterate for t until a stop criterion is met

{

for each sample x_i with label y_i:

{

compute the output ($\hat{y}_i$) of the network

estimate the error of the network $e(w(t))= y_i - \hat{y}_i$

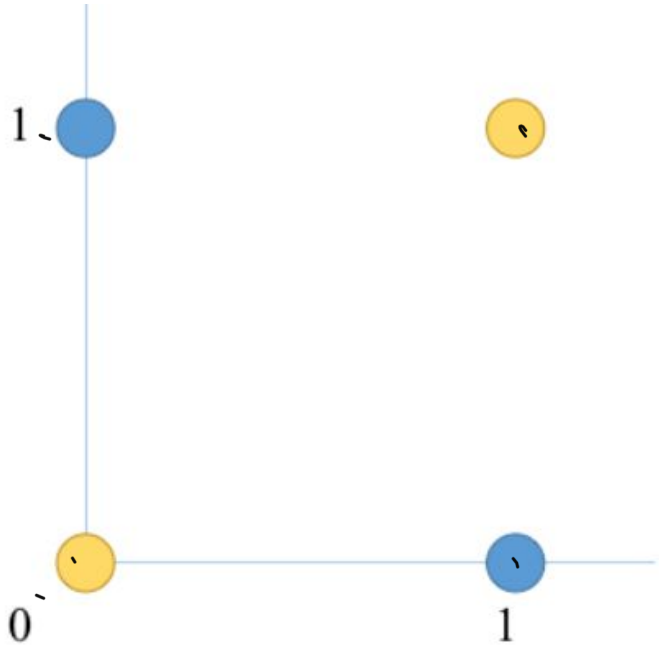
update the weight $w(t+1)=w(t)+e(w(t))x_i$

}

t++

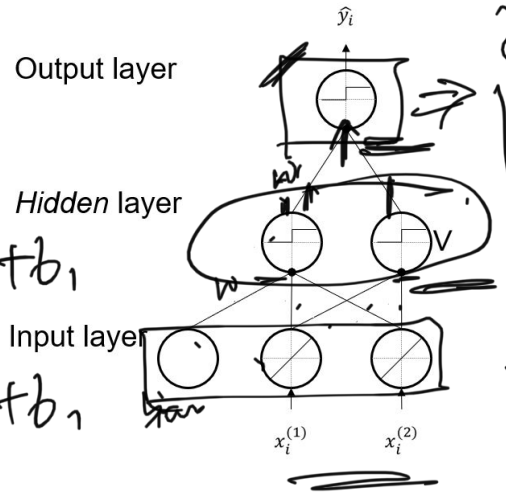
}

XOR problem



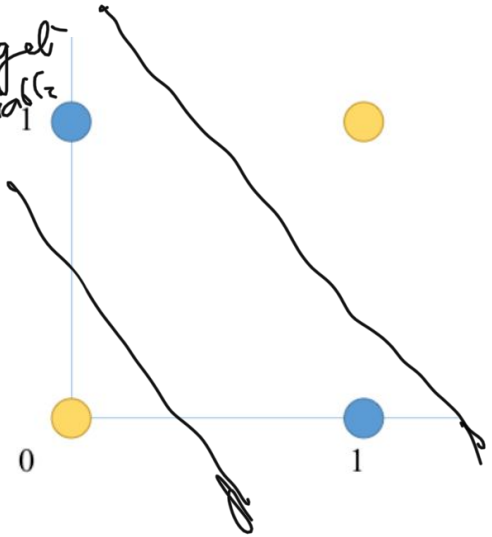
XOR problem - Using MLP (Multi-layer perceptron).

$w_{t+1} = w_t - \eta \nabla_w L$



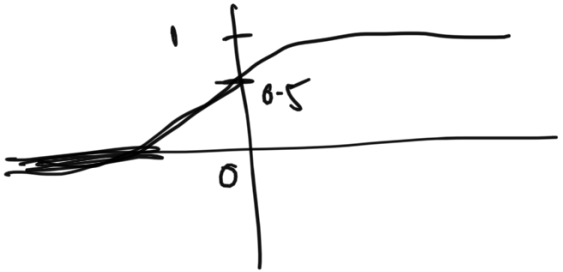
$z_1 = w_1 x + b_1$

$z_2 = w_2 x + b_1$



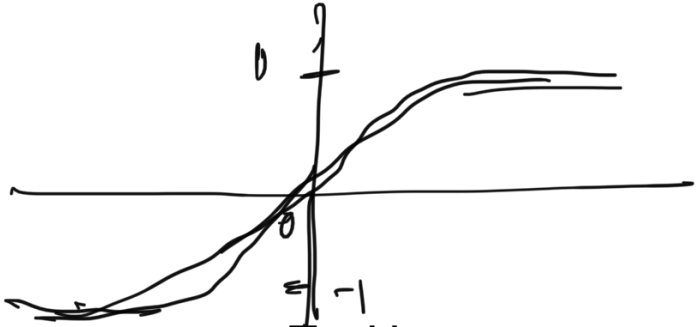
- How do we learn weights? Can perceptron learning be applied?
  - Gradient decent
  - Differentiable.
  - Backpropagation.

Activation Functions



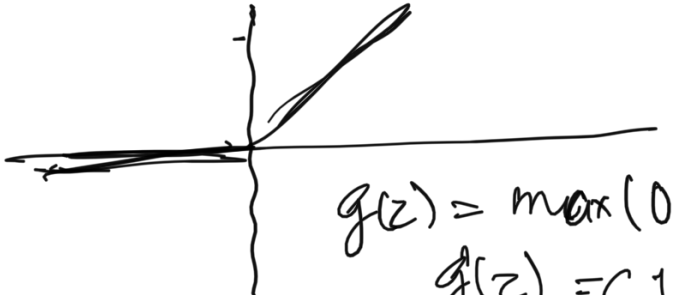
Sigmoid

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



TanH

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

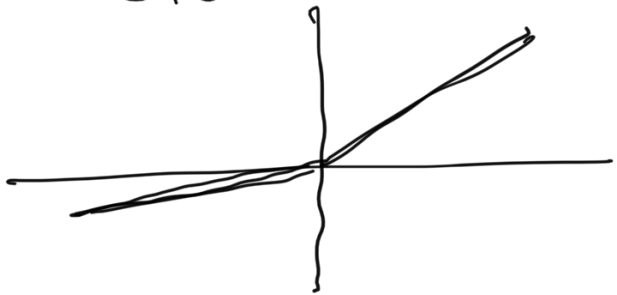


ReLU

(Rectified Linear Unit)

$g(z) = \max(0, z)$

$g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$



Leaky ReLU

$g(z) = \max(\epsilon z, z)$

Why non-linear activation functions are important?

$w_1 x + b_1 \rightarrow w_2(w_1 x + b_1) + b_2$

$w_2(\sigma(w_1 x + b_1))$

w'

- Bring non-linearity to the network.

[0.05 0.05 0.9]

Handling Multiple Classes

- No. of output neurons - No. of classes.
- Activation function - Softmax activation — $\frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$ ⇒ Normalization Constant

  C → no. of classes.

  — Bring the values betn 0 & 1.

  — argmax of softmax probabilities
- Error Function- Cross-entropy (binary cross entropy)

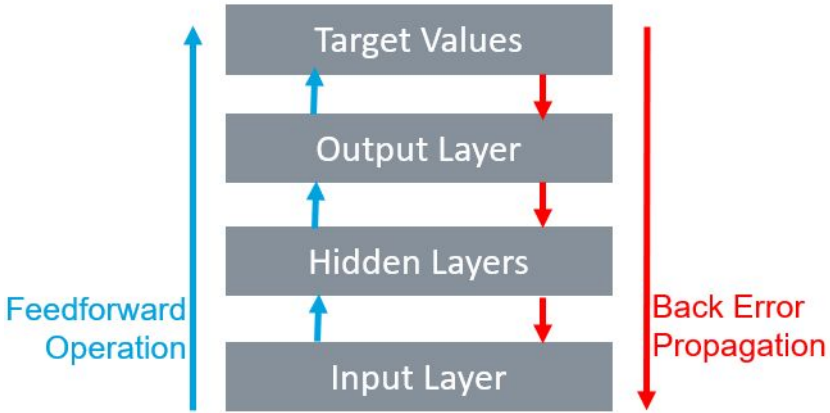
$$L = -\sum_{i=1}^{n} \sum_{j=1}^{} \Pi_j(y_i) \log P(y_i = j \mid x_i)$$

$\Pi_j(y_i) \rightarrow \begin{cases} 1 \\ 0 \end{cases}$ $y_i = j$ Otherwise

softmax probability.

MLP Learning - Back Propagation

- Key idea - Properly distribute error computed from output layer back to earlier layers to allow their weights to be updated in a way that reduce the error.



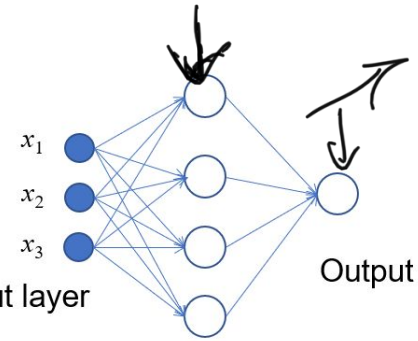
4x3 (4x1) 4x3
Broadcasting

MLP Learning - Back Propagation

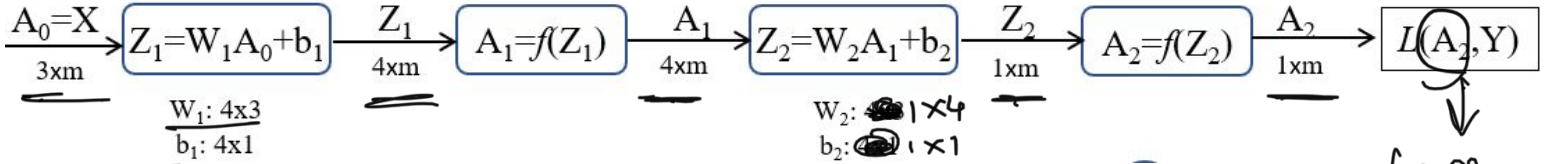
Feed-Forward Phase-

m → total no. of samples.

activation fun^n → sigmoid.



3-dimension input
4 hidden nodes
1 output nodes
(binary classification)



$Z_1 = W_1 A_0 + b_1$

(4xm) (4x3) (3xm) (4x1)

$Z_2 = W_2 A_1 + b_2$

(1xm) (1x4) (4xm) (1x1)

MLP Learning - Back Propagation

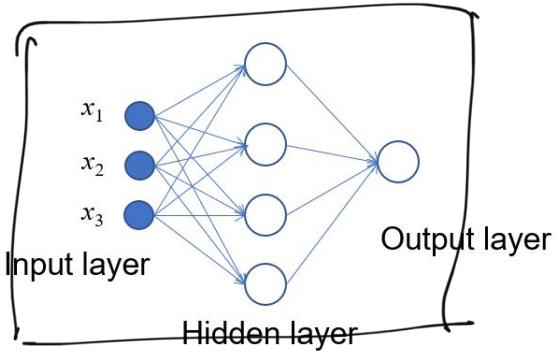
Back propagation phase-

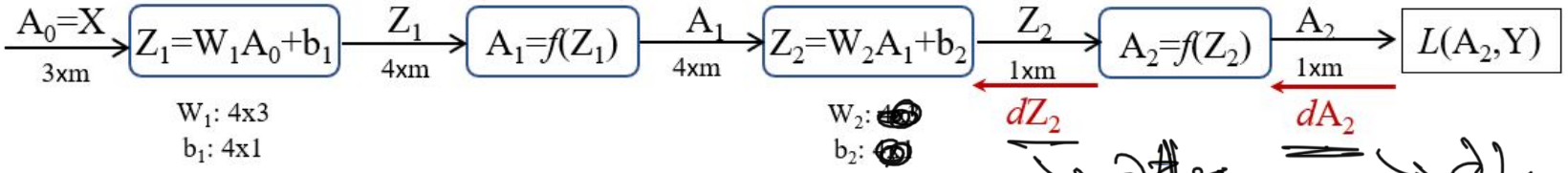
$w_{t+1} = w_t - \eta \nabla_w L$

Parameters -

W_1, b_1, W_2, b_2

L = Loss function



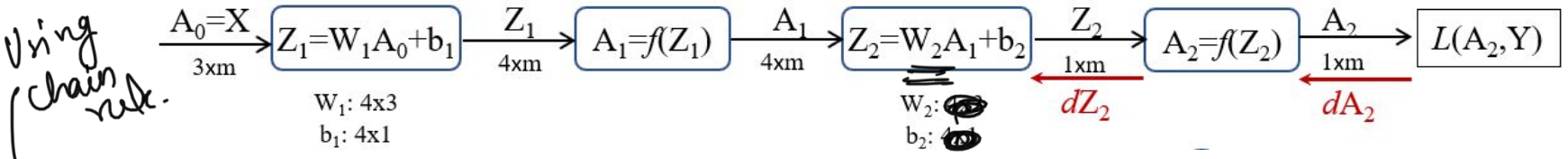


$\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}$ → chain rule.

$\frac{\partial L}{\partial Z_2}$ $\frac{\partial L}{\partial A_2}$

Activation → Sigmoid, Output → Binary loss → BCE

MLP Learning - Back Propagation



$$\frac{\partial L}{\partial W_2}=\frac{\partial L}{\partial A_2}\cdot\frac{\partial A_2}{\partial Z_2}\cdot\frac{\partial Z_2}{\partial W_2},\quad \frac{\partial L}{\partial b_2}=\frac{\partial L}{\partial A_2}\cdot\frac{\partial A_2}{\partial Z_2}\cdot\frac{\partial Z_2}{\partial b_2},\quad \frac{\partial L}{\partial W_1}=\frac{\partial L}{\partial A_2}\cdot\frac{\partial A_2}{\partial Z_2}\cdot\frac{\partial Z_2}{\partial A_1}\cdot\frac{\partial A_1}{\partial Z_1}\cdot\frac{\partial Z_1}{\partial W_1}$$

$$\frac{\partial L}{\partial b_1}=\frac{\partial L}{\partial A_2}\cdot\frac{\partial A_2}{\partial Z_2}\cdot\frac{\partial Z_2}{\partial A_1}\cdot\frac{\partial A_1}{\partial Z_1}\cdot\frac{\partial Z_1}{\partial b_1}$$

$$\frac{\partial Z_2}{\partial A_1}=W_2^T,\quad \frac{\partial A_1}{\partial Z_1}=A_1(1-A_1)$$

$$L=-\left(y\log A_2+(1-y)\log(1-A_2)\right)$$

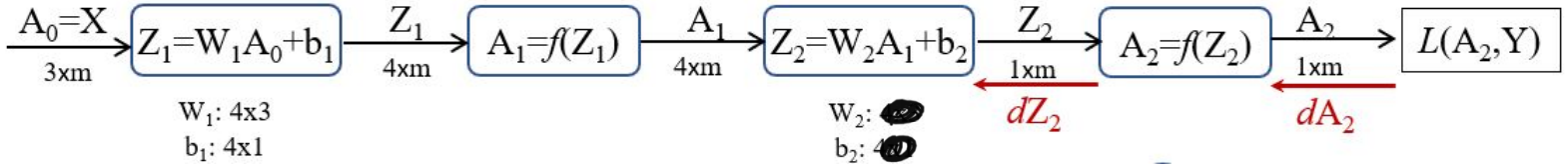
$$\frac{\partial Z_1}{\partial W_1}=A_0^T,\quad \frac{\partial Z_1}{\partial b_1}=[1,\ldots.1]^T$$

$$\frac{\partial L}{\partial A_2}=\frac{A_2-Y}{A_2(1-A_2)},\quad \frac{\partial Z_2}{\partial W_2}=A_1^T$$

$$\frac{\partial A_2}{\partial Z_2}=A_2(1-A_2),\quad \frac{\partial Z_2}{\partial b_2}=[1,1\ldots.1]^T$$

1xm 4x1 4xm mx1

MLP Learning - Back Propagation



$$\frac{\partial L}{\partial W_2} = \underbrace{(A_2-Y)}_{1\times m} \cdot \underbrace{A_2(1-A_2)}_{1\times m} \cdot A_1^T \quad (1\times 4,\ m\times 4)$$

$$\frac{\partial L}{\partial b_2} = (A_2-Y) \cdot A_2(1-A_2) \cdot [1 \dots 1]^T \quad (1\times 1,\ m\times 1)$$

$$\frac{\partial L}{\partial W_1} = (A_2-Y) \cdot A_2(1-A_2) \cdot (W_2)^T \cdot A_1(1-A_1) \cdot A_0^T$$

$$\frac{\partial L}{\partial W_1} = \underbrace{(W_2)^T}_{4\times 1}\ \underbrace{(A_2-Y)}_{1\times m}\ \underbrace{A_1(1-A_1)}_{4\times m}\ \underbrace{A_0^T}_{m\times 3} \quad (4\times 3)$$

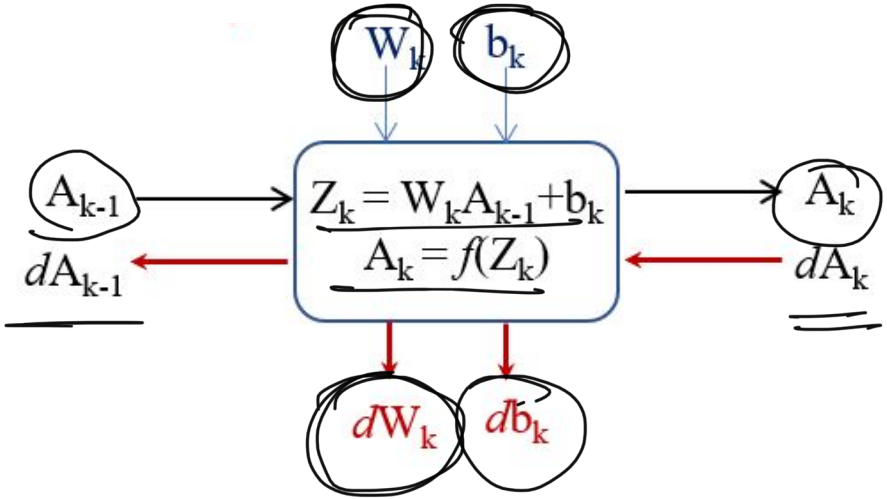
Element-wise multiplication

$$\frac{\partial L}{\partial b_1} = (A_2-Y) \cdot A_2(1-A_2) \cdot W_2^T A_1(1-A_1) \cdot [1 \dots 1]^T$$

$$\frac{\partial L}{\partial b_1} = \underbrace{W_2^T}_{4\times 1} \cdot \underbrace{(A_2-Y)}_{1\times m}\ \underbrace{A_1(1-A_1)}_{4\times m}\ \underbrace{[1\dots 1]^T}_{m\times 1} \quad (4\times 1)$$

Element-wise multiplication

MLP Learning - Back Propagation



Questions?