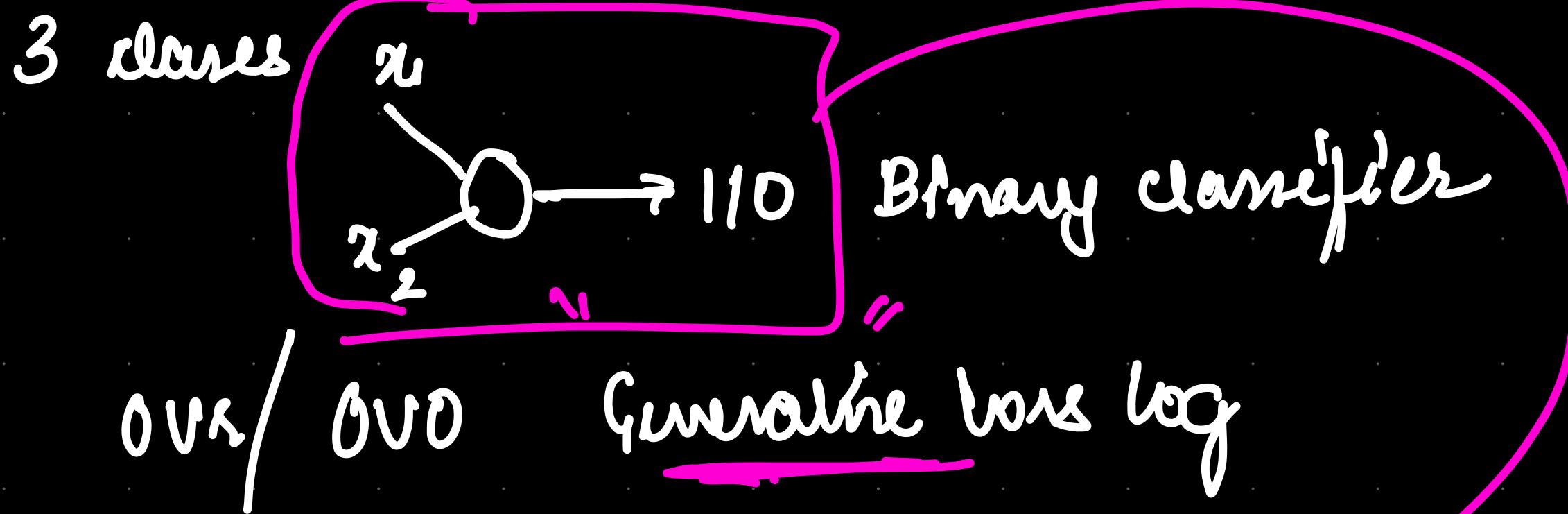


NN- lecture- 2

Forward and Backward

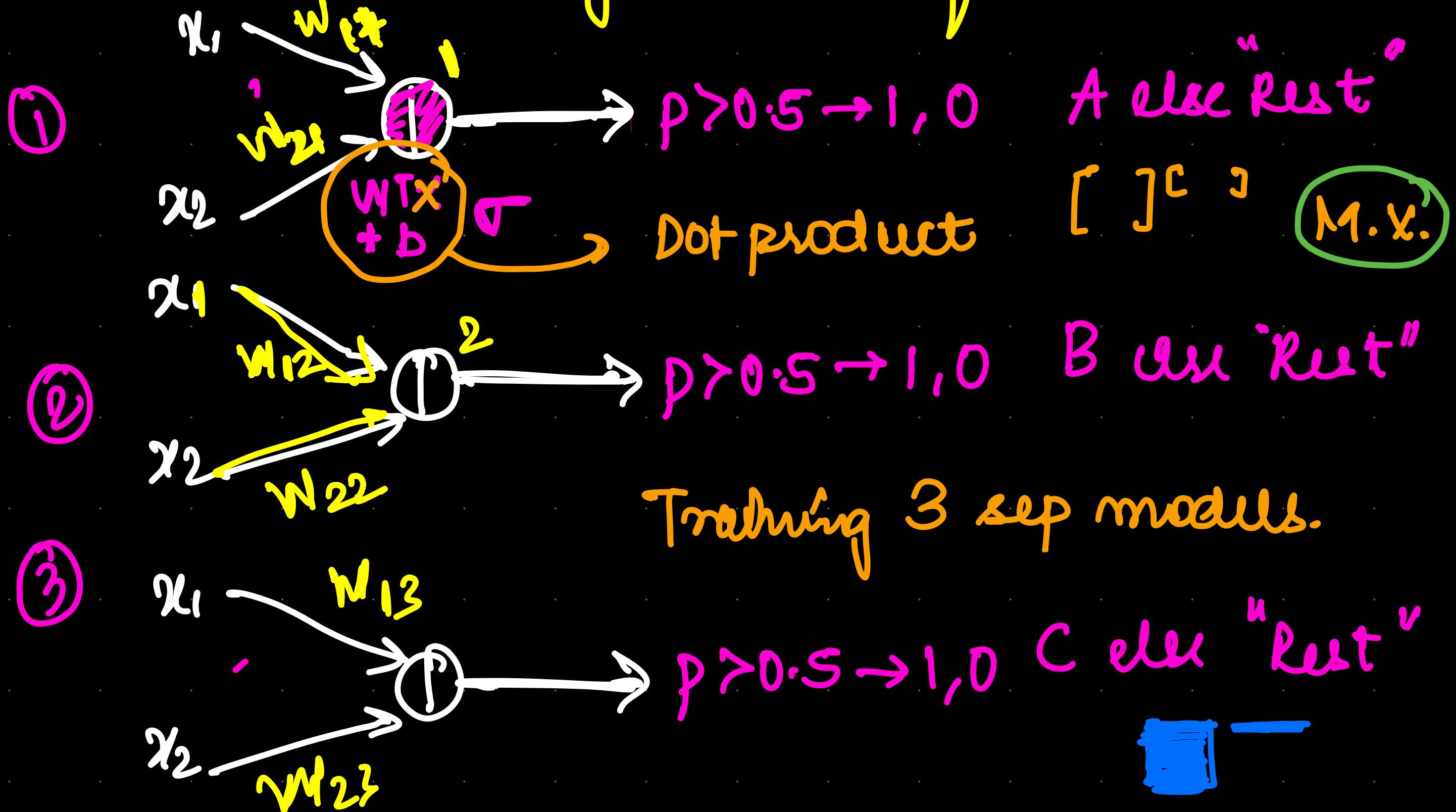
Propagation

Task: Make hogR work for Multi-class



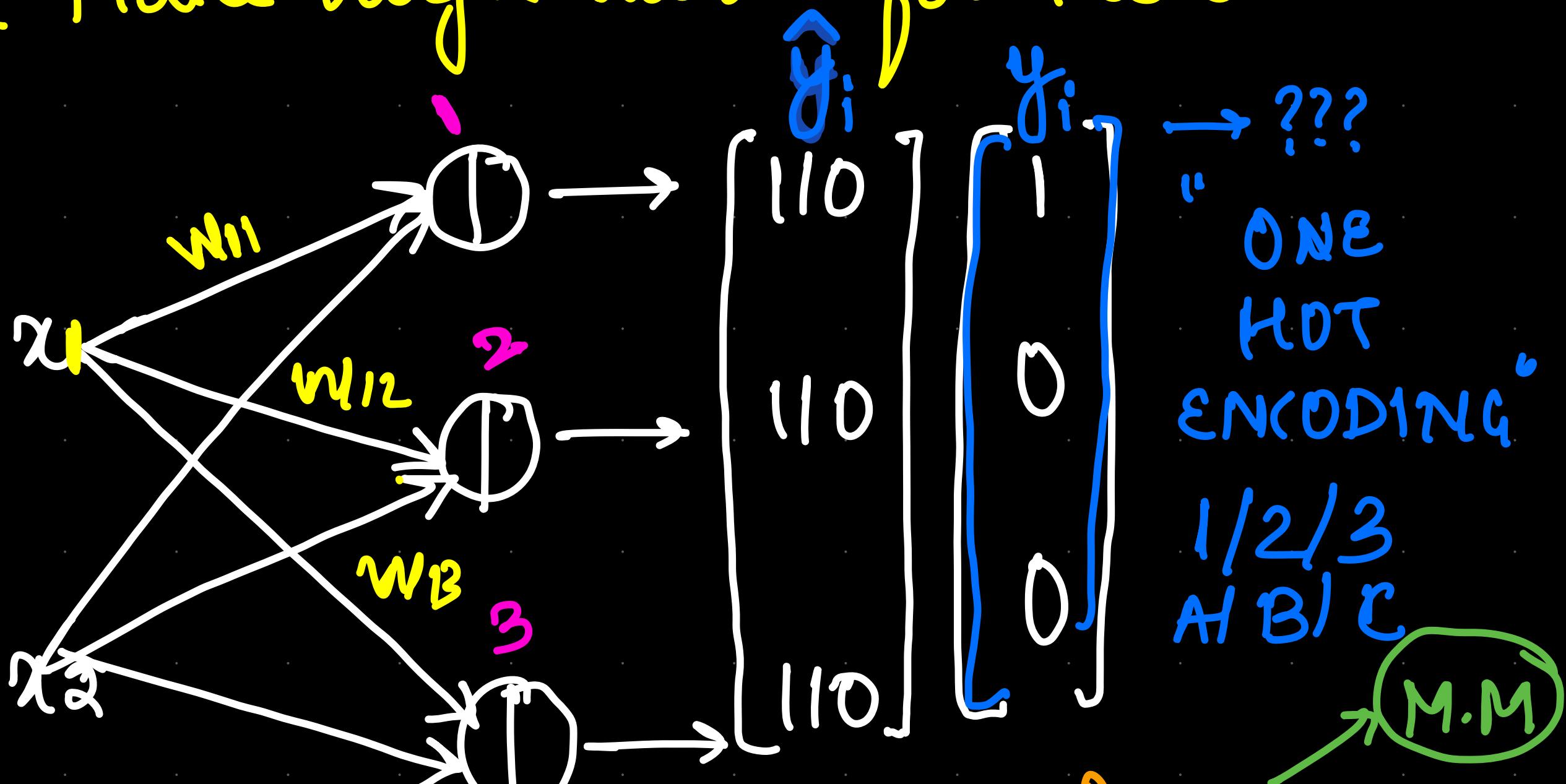
2 class.

Task: Make hogR work for Multi-class



Task: Make hogR work for Multi-class

$\hat{y}_i, y_i$   
one vector



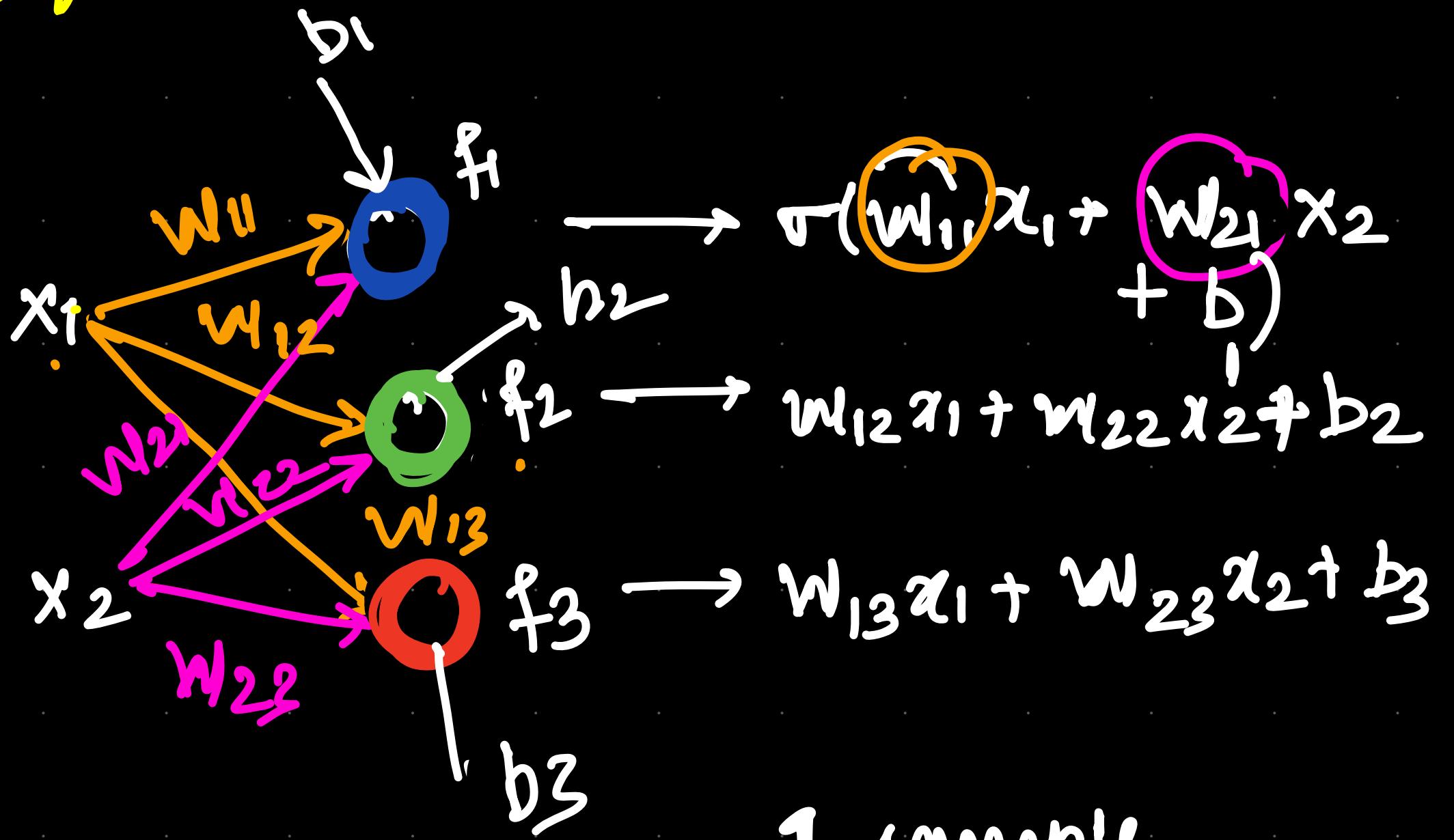
Something is happening together -

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

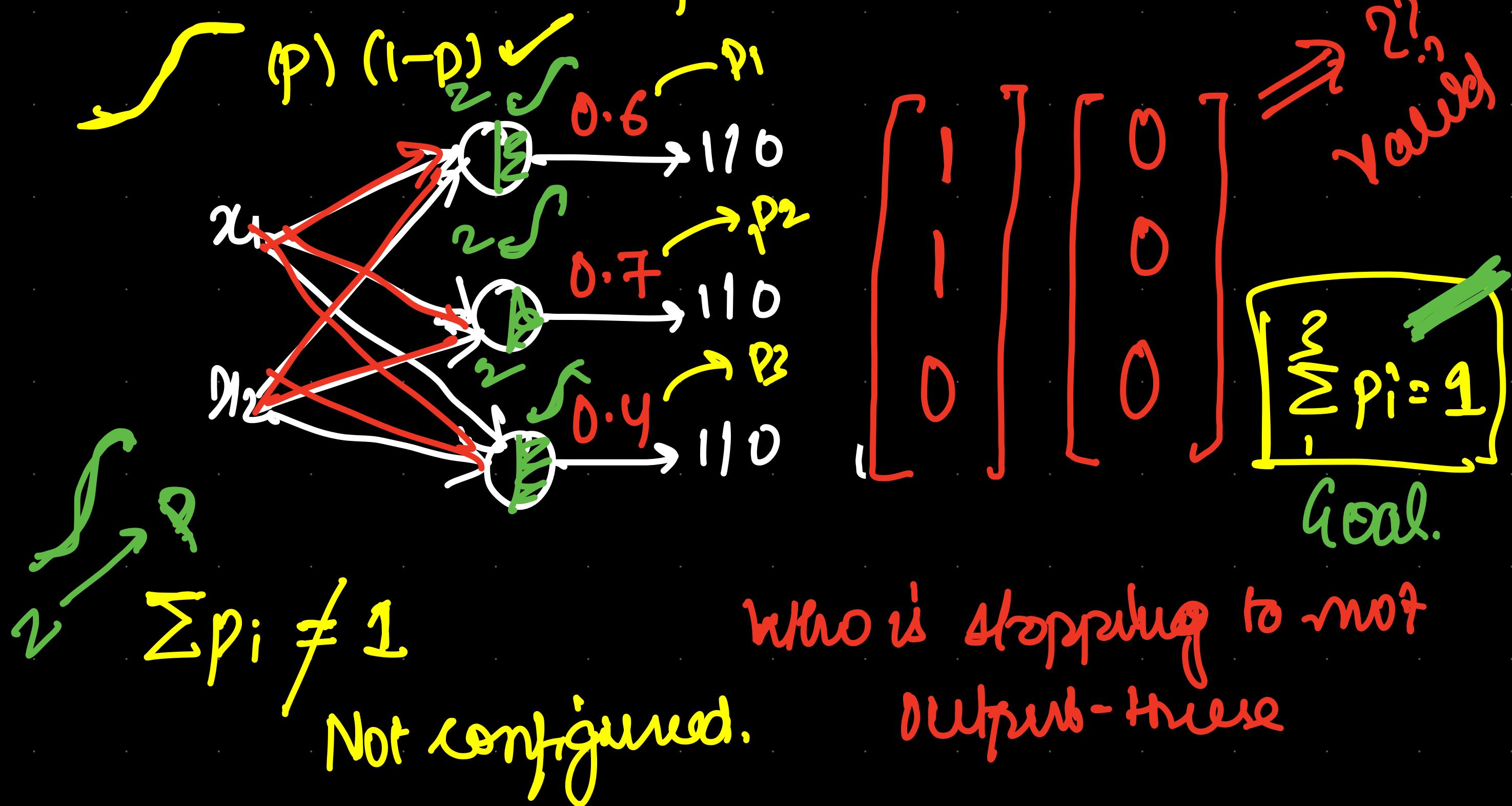
$W^T x + b$  for Multiple LRVs

weights

$$\begin{aligned} x_1 &\xrightarrow{w_{11}} f_1 \\ x_2 &\xrightarrow{w_{23}} f_3 \end{aligned}$$



What is the problem with this?



# softmax Classifier

①

$$z \rightarrow z_1 / \sum z - p_1$$

$$z \rightarrow z_2 / \sum z - p_2$$

$$z \rightarrow z_3 / \sum z - p_3$$

Normalized props

Normalizing

$$z_i \rightarrow z_i / (z_1 + z_2 + z_3)$$

②

~~$$z_1 \rightarrow e^{z_1} / \sum e^z \rightarrow p_1$$~~
~~$$z_2 \rightarrow e^{z_2} / \sum e^z \rightarrow p_2$$~~
~~$$z_3 \rightarrow e^{z_3} / \sum e^z \rightarrow p_3$$~~

$$p_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Normalising  
after  
exponentiating

Why not simple scaling?

①  $z \rightarrow [-\infty, \infty]$   $p \rightarrow [-\infty, \infty]$  Need **tve**  $0 < \leq 1$

② Differentiability -  $e^x$  gives benefits.

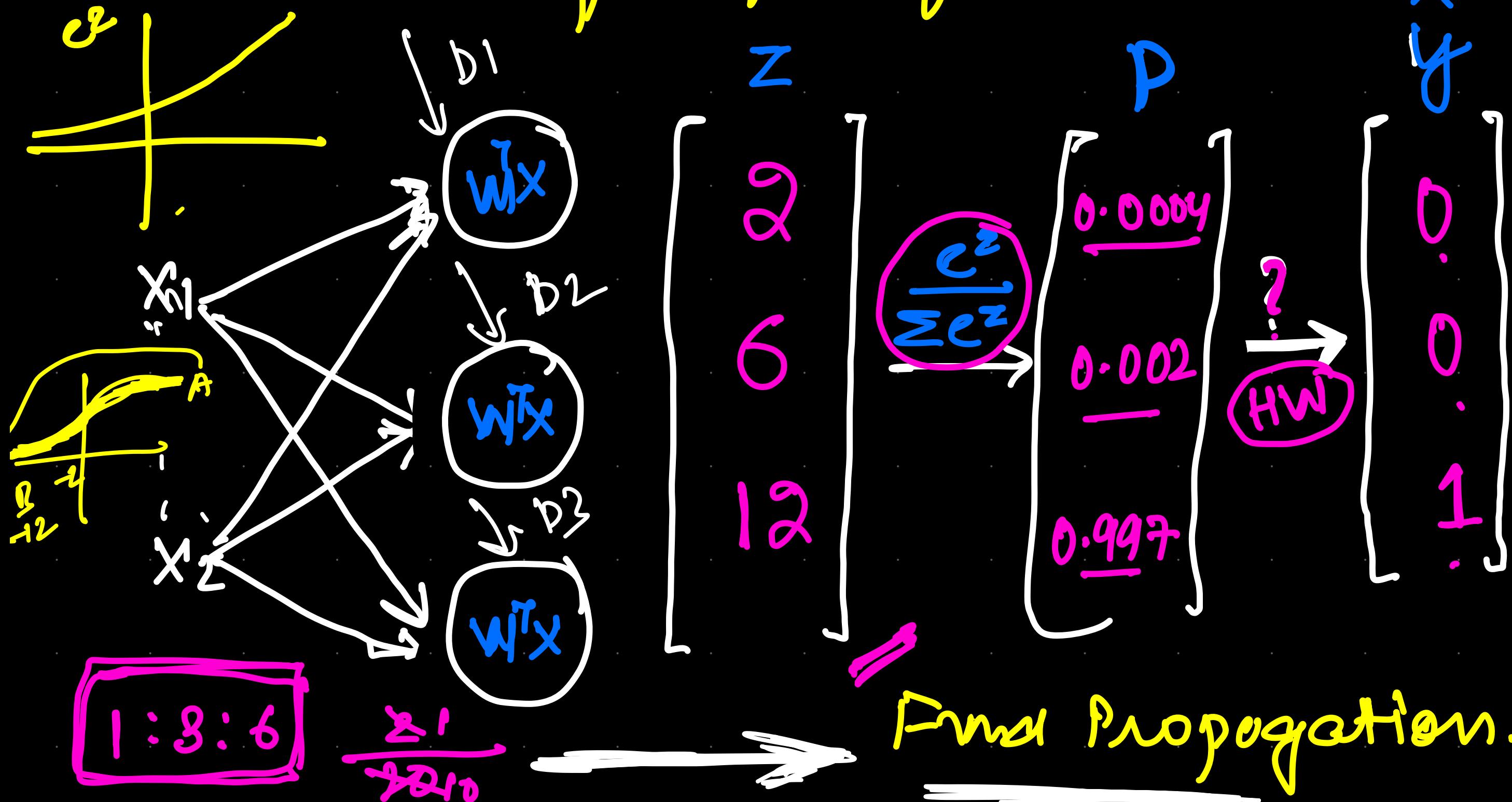
Harvest?

**ex** is expressive, why do it?  
→ **Non-linear  
decision  
boundary**.

Measuring non-linear functions.

11/10, 3/10, 6/10

# Other benefit of softmax



lets forward propagate

Data Matrix -  $m \times d$   
# samples X # features.

Output Matrix -  $m \times n$   
# samples X # classes.

Weight Matrix ??  
..

lets forward propagate - weights

D

W ???

$$\begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{m1} & x_{m2} \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_B \\ W_{21} & W_{22} & W_{23} \\ \vdots & \vdots & \vdots \\ N_1 & N_2 & N_3 \end{bmatrix} + b = \begin{bmatrix} \hat{y}_{11} & \hat{y}_{12} & \hat{y}_{13} \\ \vdots & \vdots & \vdots \\ \hat{y}_{m1} & \hat{y}_{m2} & \hat{y}_{m3} \end{bmatrix}$$

$d \times m$

$m \times q$   
 $m \times 2$

$2 \times 3$

$m \times m$   
 $m \times 3$

Bias Term is easy to add



# Net forward propagate - Bias

$$\begin{bmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{m1} & x_{m2} \end{bmatrix}_{m \times d} \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}_{m \times m}^T \begin{bmatrix} \end{bmatrix}_{d \times m}$$

$$\begin{bmatrix} b_1 & b_2 & b_3 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

Broadcasting  $m \times n$

$1 \times m$



$m \times m$

$m \times m$  = shape of output

$w \times h \times b$

# Train softmax C. using **GD**

- 1) Initialize  $W$  and  $b$
- 2) calculate  $\hat{y}$  using hypothesis  $f(WX+b)$
- 3) calculate the error  $J$  - log loss.
- 4) Repeat until  $J$  converges

→ update  $w$  and  $b$   $w \rightarrow w - lr \frac{\partial J}{\partial w}$   
→ calculate  $f(WX+b)$  again  
→ calculate  $J$  again.

lets write code

Initialisation of  $W$  and  $b$

Find Propogation. to get  $P$

$$\begin{matrix} & & \\ x & w & b \end{matrix}$$

# loss for softmax classifier

## loss for logistic Regression

log loss

$$= - \sum_{i=1}^m (y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i))$$

Class 1

Class 2

2 classes

$\circled{n}$  classes

$$\sum_{i=1}^m \sum_{j=1}^n p_{ij} \log(\hat{p}_{ij})$$

→ Multi-class CE loss.

wg loss  
Binarized  
Cross  
Entropy  
loss

$i \rightarrow$  class  
 $j \rightarrow$  sample

Loss for softmax classifer

$$M\text{-CE}_{ith} = - \sum_{j=1}^n y_{ij} \log \hat{P}_{ij}$$

1 only for one class

case:

$$M\text{-CE}_{ith} = - \log \hat{P}_{ik}$$

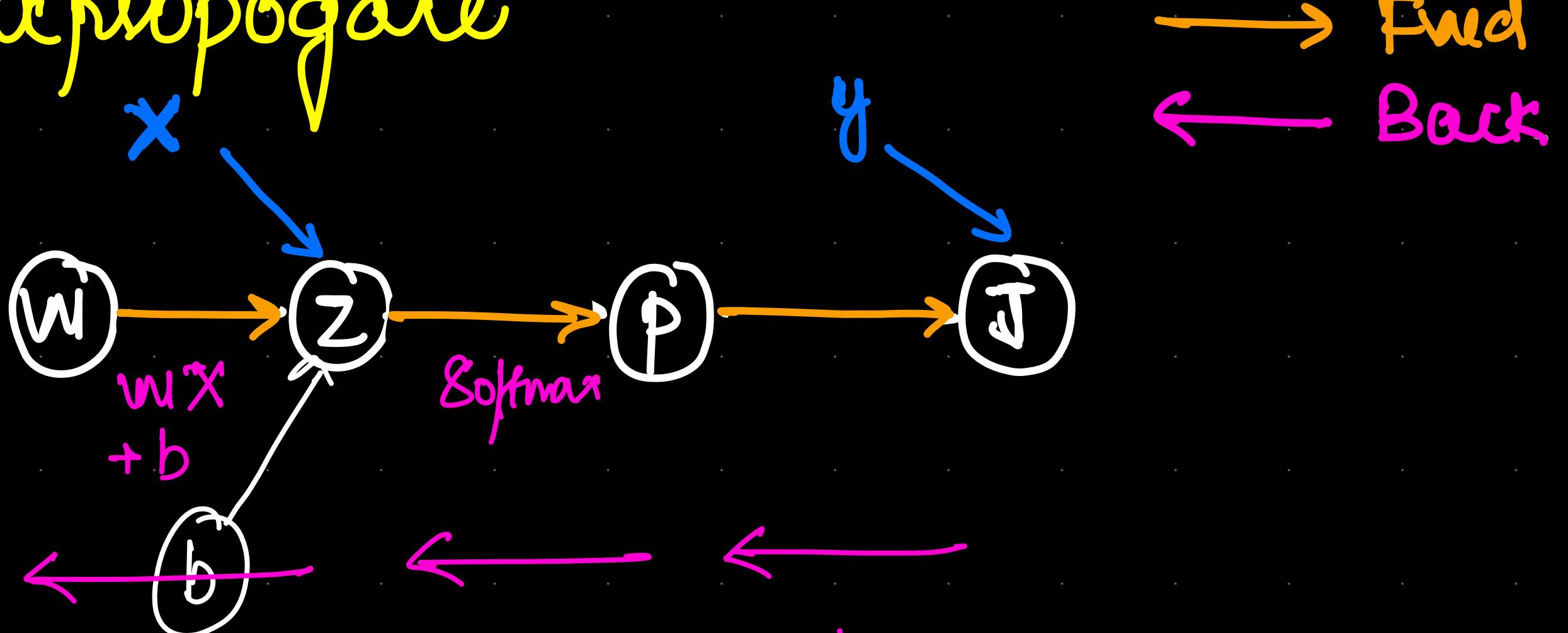
$k = \text{true class.}$

~~- log prob of the true class.~~

lets update  $w$  and  $b$  - Backpropagate

How to  
update  
the  
 $w$  and  $b$

# Backpropagate



Backpropagation

Update.

$$\boxed{\frac{\partial J}{\partial w} \quad \frac{\partial J}{\partial b}}$$

# Gradient Calculation.

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial P} \frac{\partial P}{\partial z} \frac{\partial z}{\partial w}$$

Short  
Hand

# Gradient calculation

$$\begin{bmatrix} \frac{\partial J}{\partial P} & \frac{\partial P}{\partial z} \end{bmatrix}$$

$$\frac{\partial z}{\partial w}$$

