Artificial Neural Network for Classification (2)

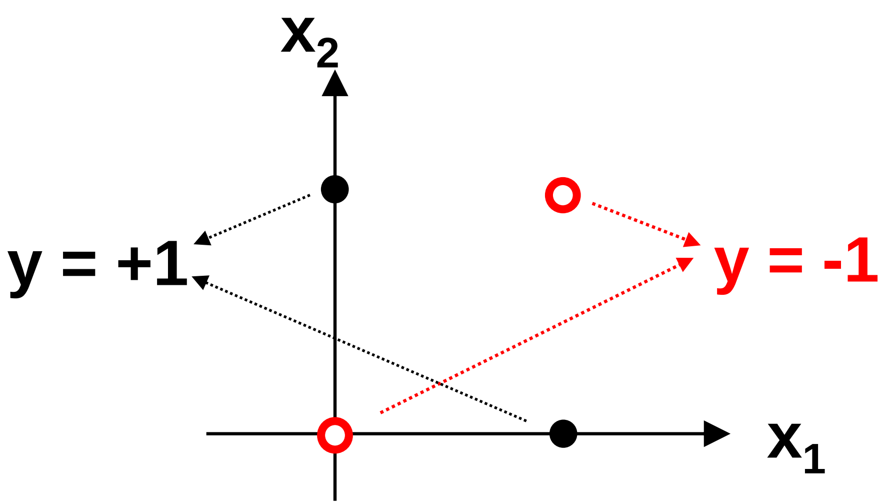
Hanwool Jeong

hwjeong@kw.ac.kr

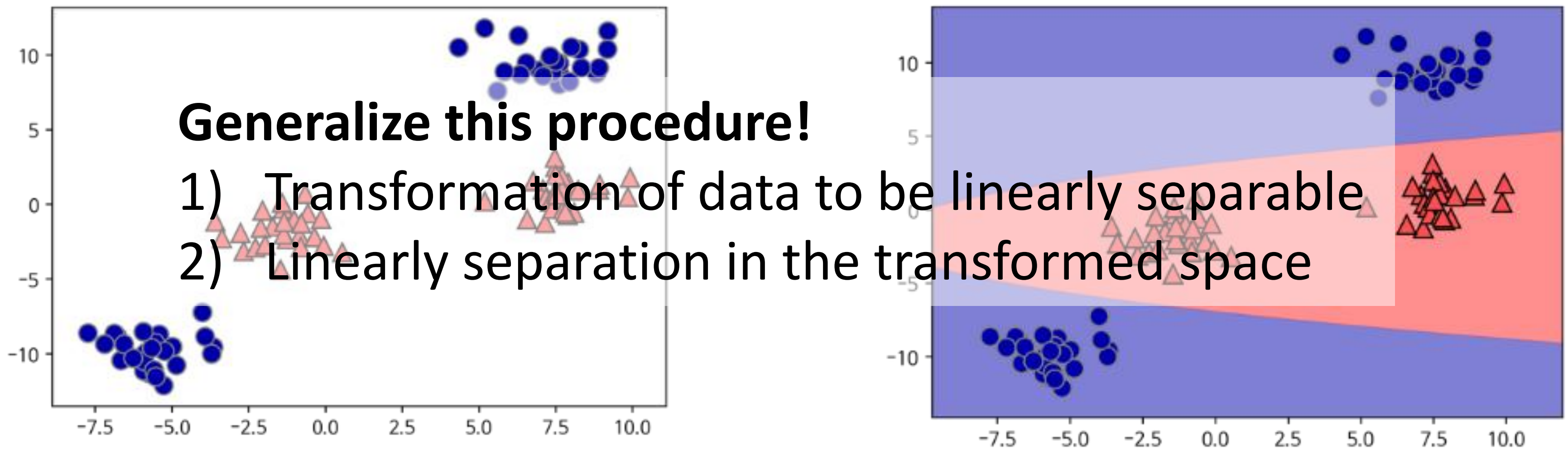
Can Perceptron Solve XOR Problem?

- Linearly non-separable dataset cannot be classified by the perceptron [Minsky, 1969].
- However, in 1974, Werbos's dissertation says linearly non-separable dataset can be classified by the perceptron through multi layers.

  ➔ **That is multi layer perceptron can solve XOR problem.**

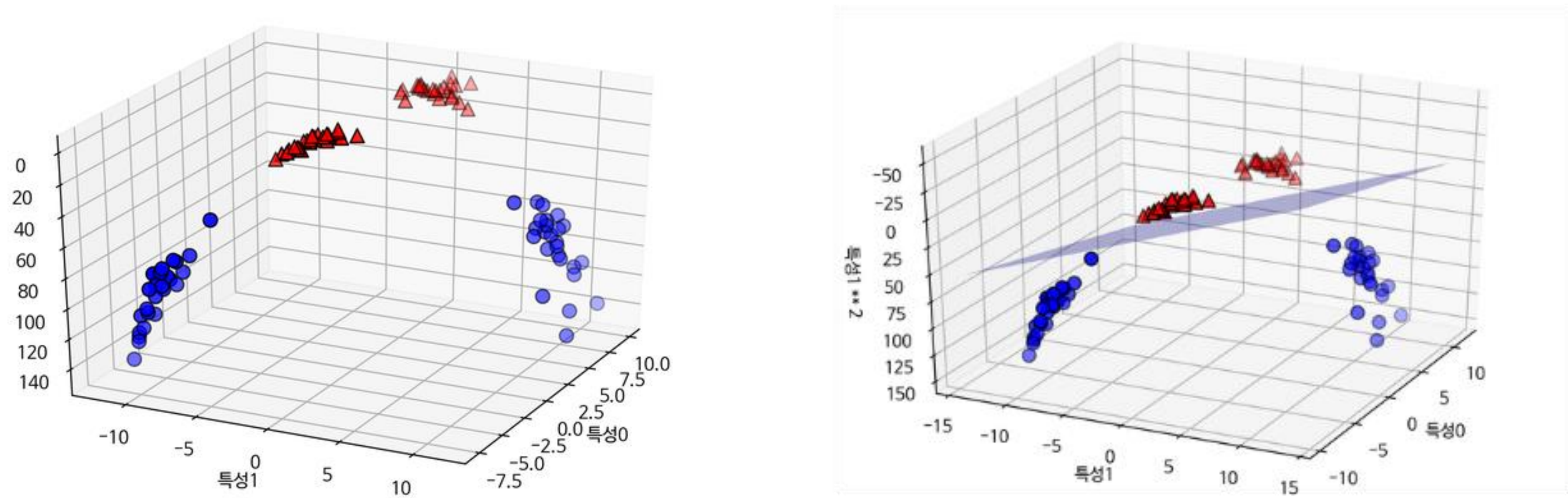


Revisit: How Do We Handle Linearly inseparable Data in SVM?



Generalize this procedure!

1) Transformation of data to be linearly separable
2) Linearly separation in the transformed space



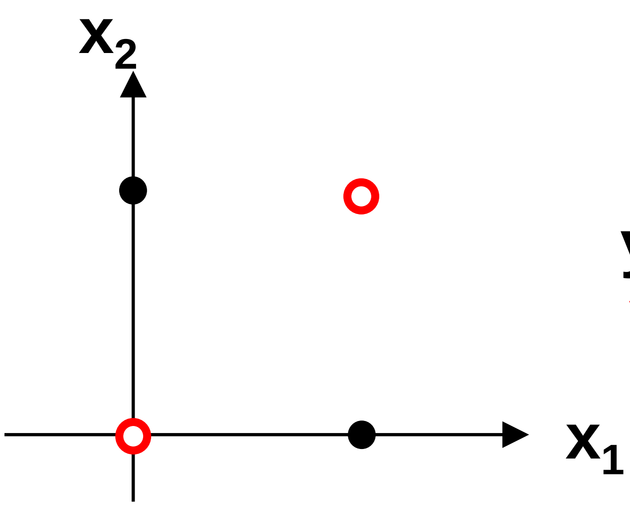
Ideation!

Can data be **transformed** using linear operation (=inner product) followed by non-linear sign function?

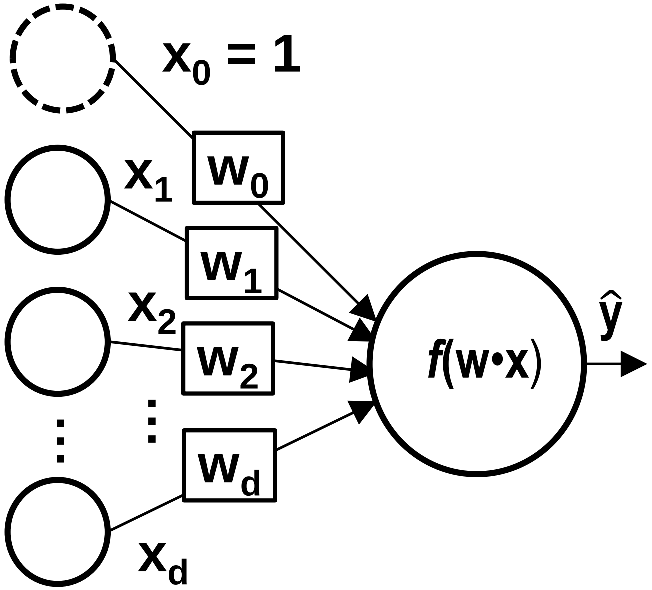
➔Let's think over how perceptron classifies data

Generalize this procedure!

1) Transformation of data to be linearly separable
2) Linearly separation in the transformed space

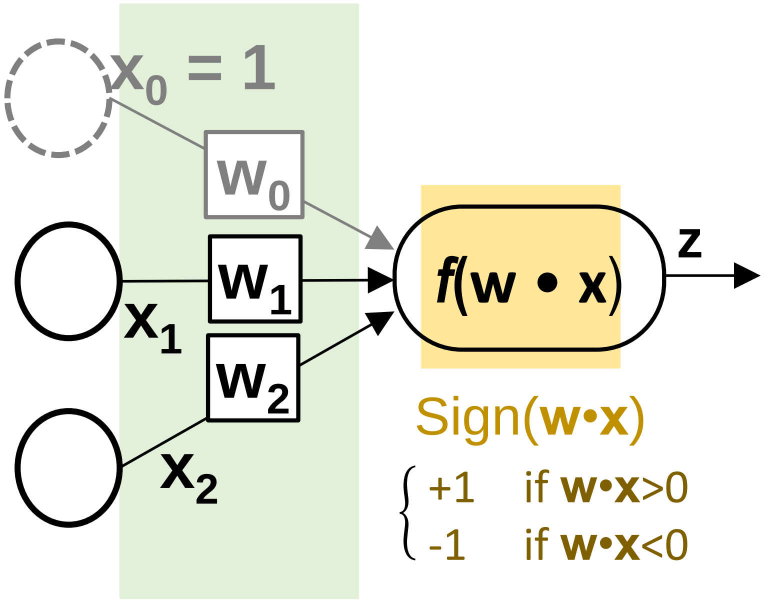


y = +1
y = -1



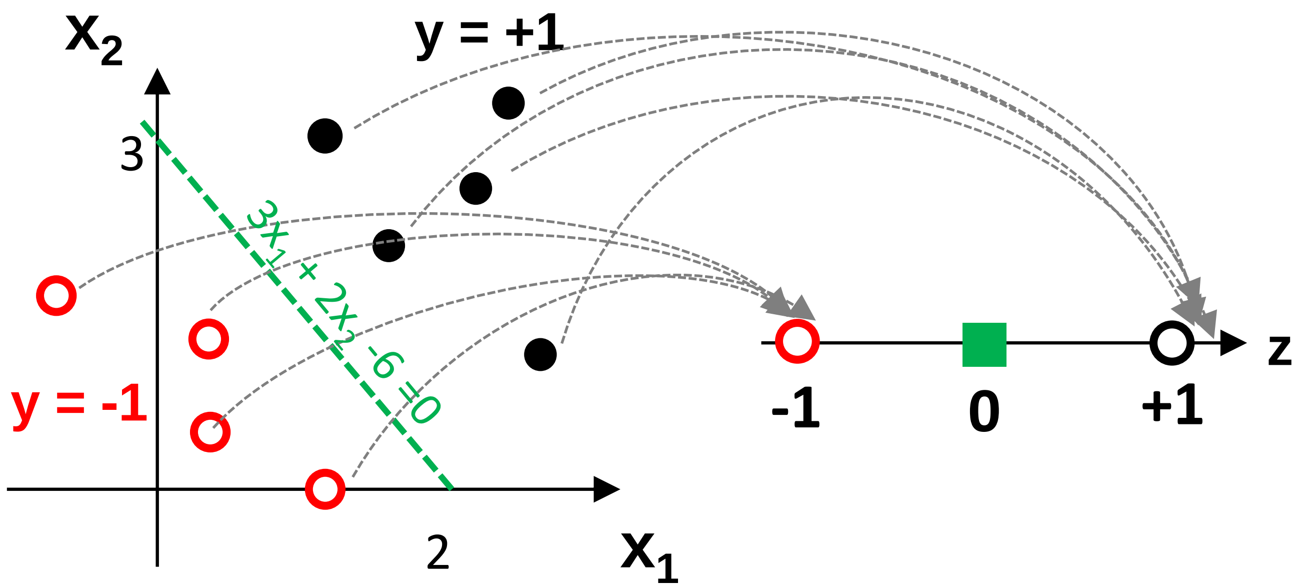
You Should Feel What Perceptron Can Do

- It is nothing but **data transformation!**



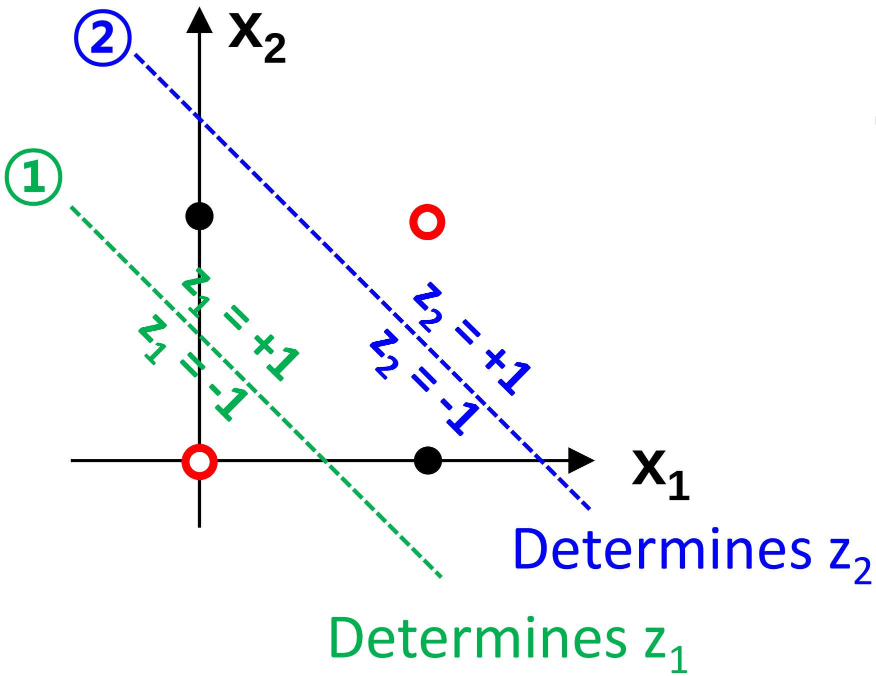
[w_0 , w_1, w_2]= [-6, 3, 2]

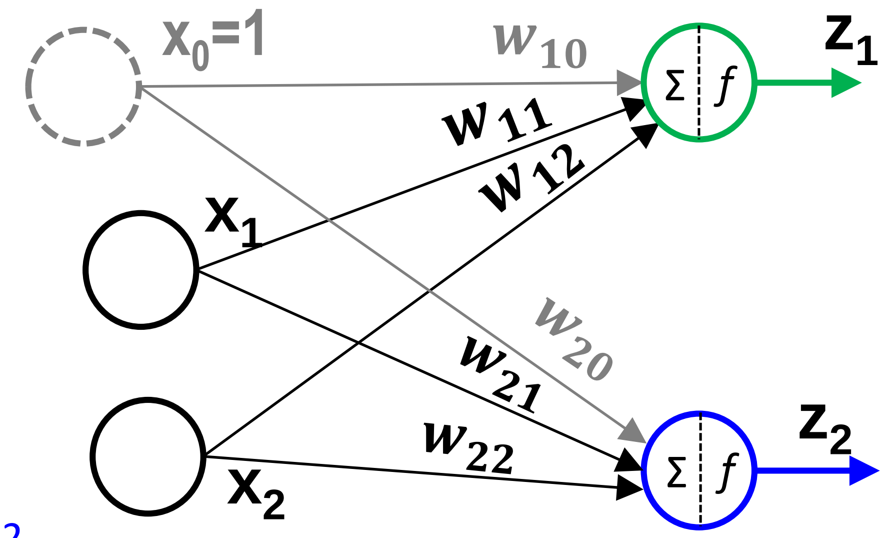
$w \bullet x = 3x_1 + 2x_2 - 6$



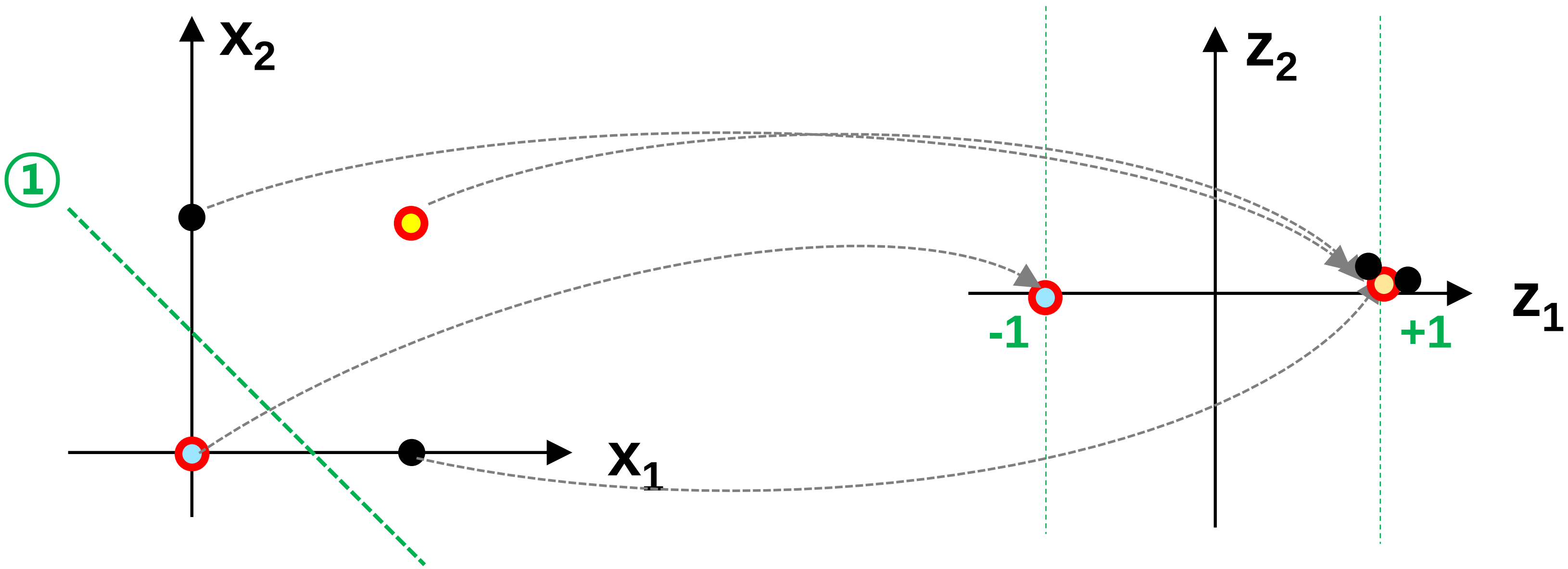
Transformation of Dataset

- Key idea is to transfer the data into new space.
- x_1-x_2 space ➔ z_1-z_2 space
- Linearly non-separable ➔ Linearly separable?

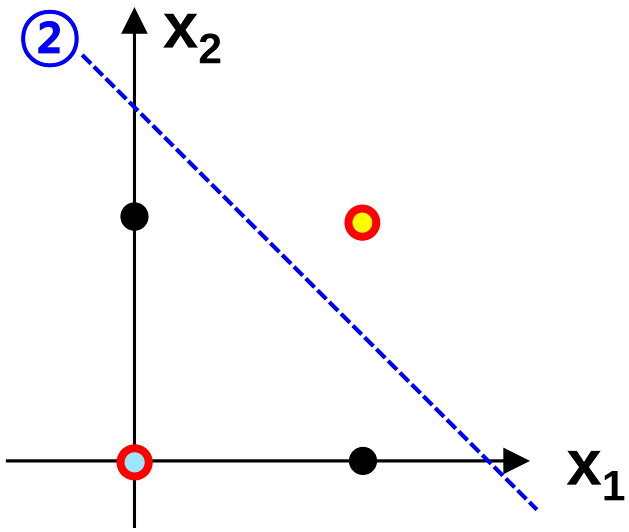


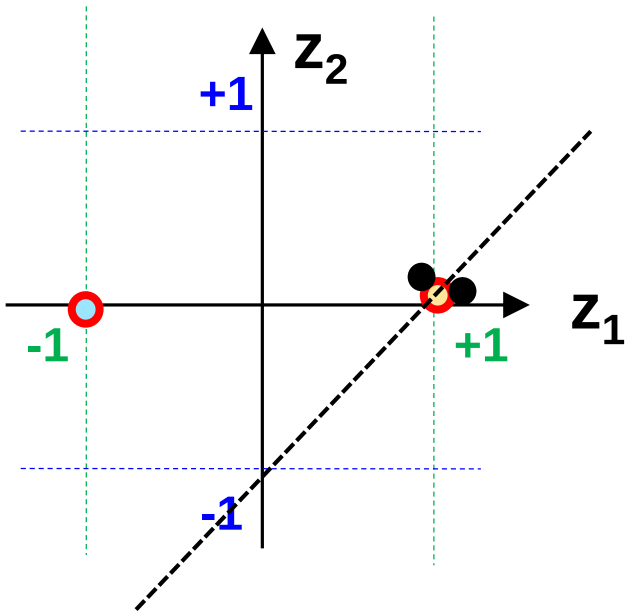


Step1 : Determines z_1



Step2 : Determines z_2

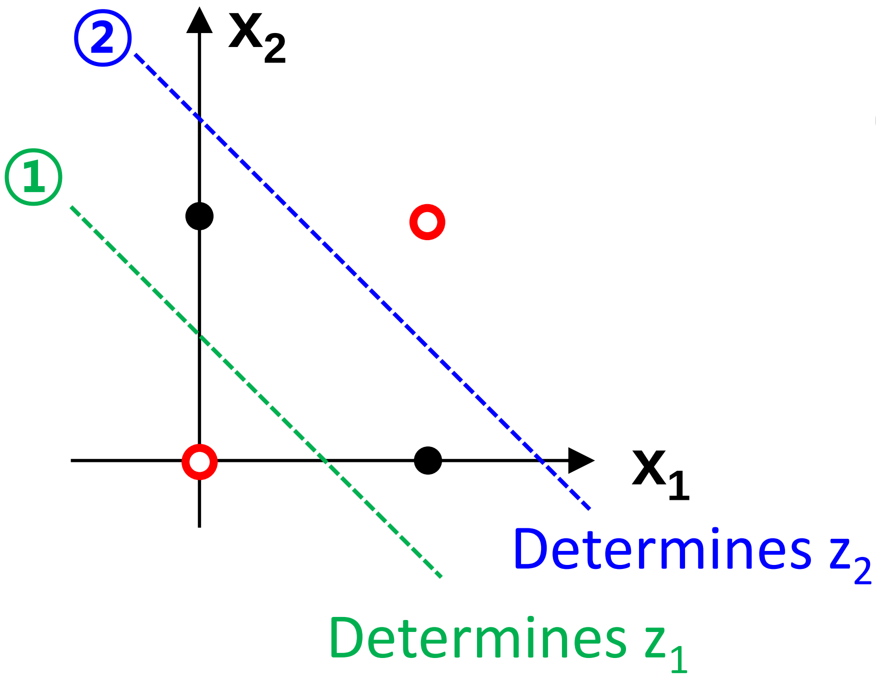


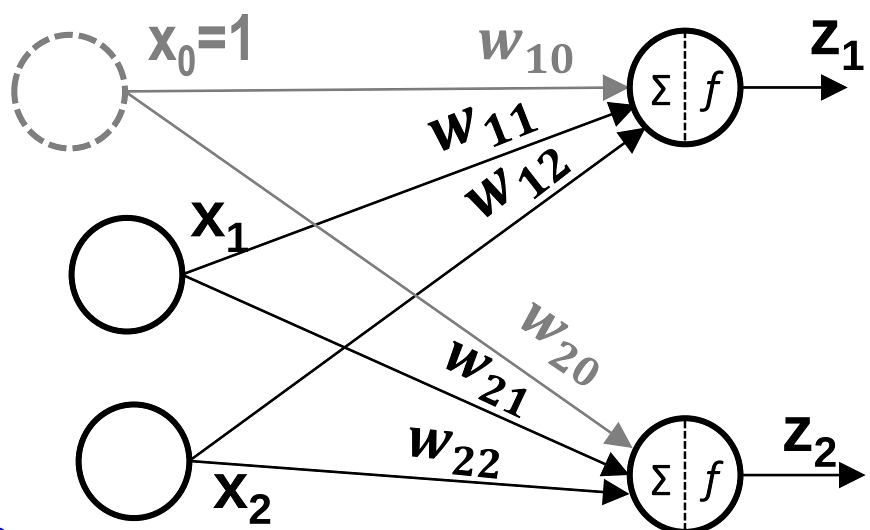


Can you Determine W Matrix?

- Saying W = $\begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$,

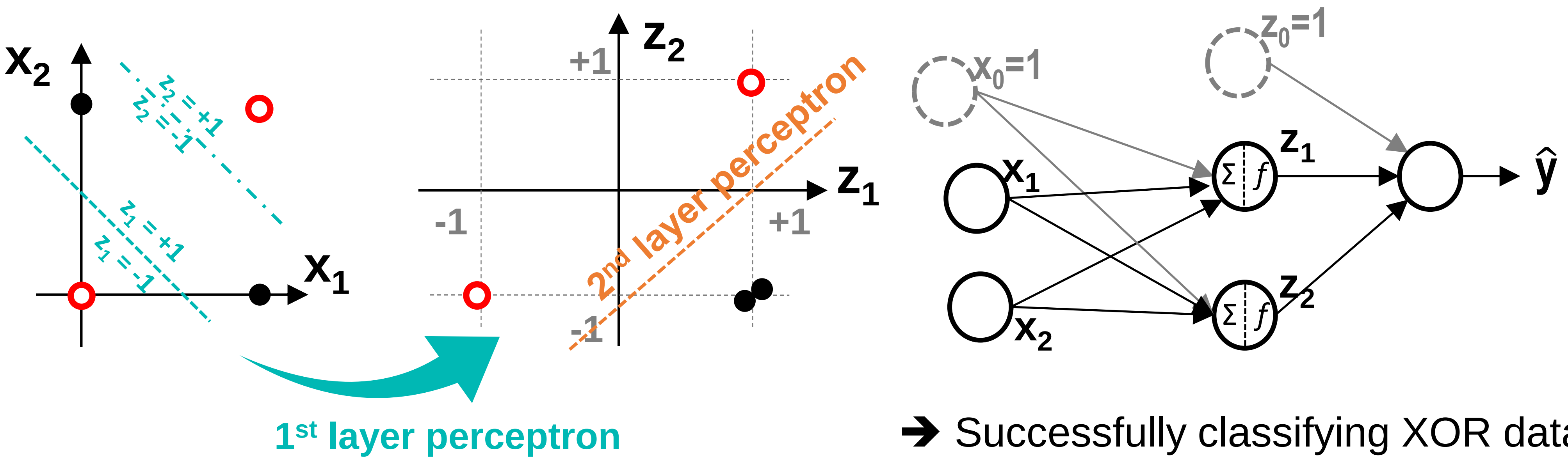
Bias





Multi Layer Perceptron (MLP)

- After became linearly separable in z_1-z_2 plane, we can do the same what we did previously, forming **MLP**

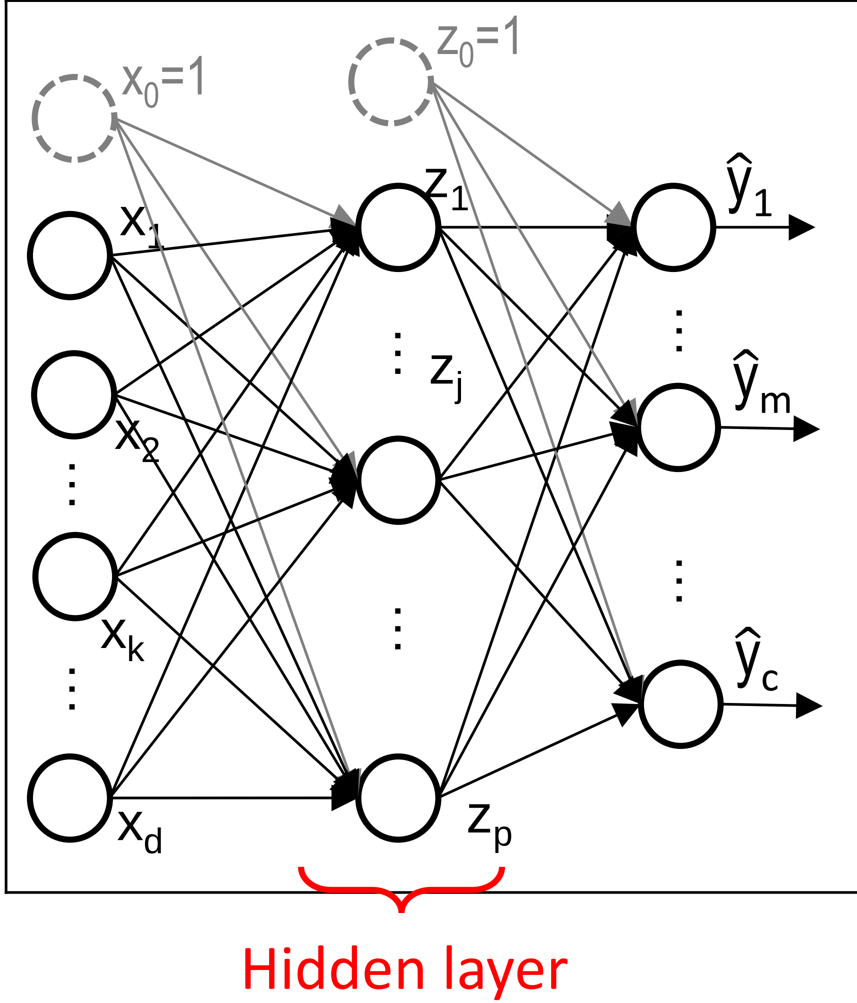


➔ Successfully classifying XOR data

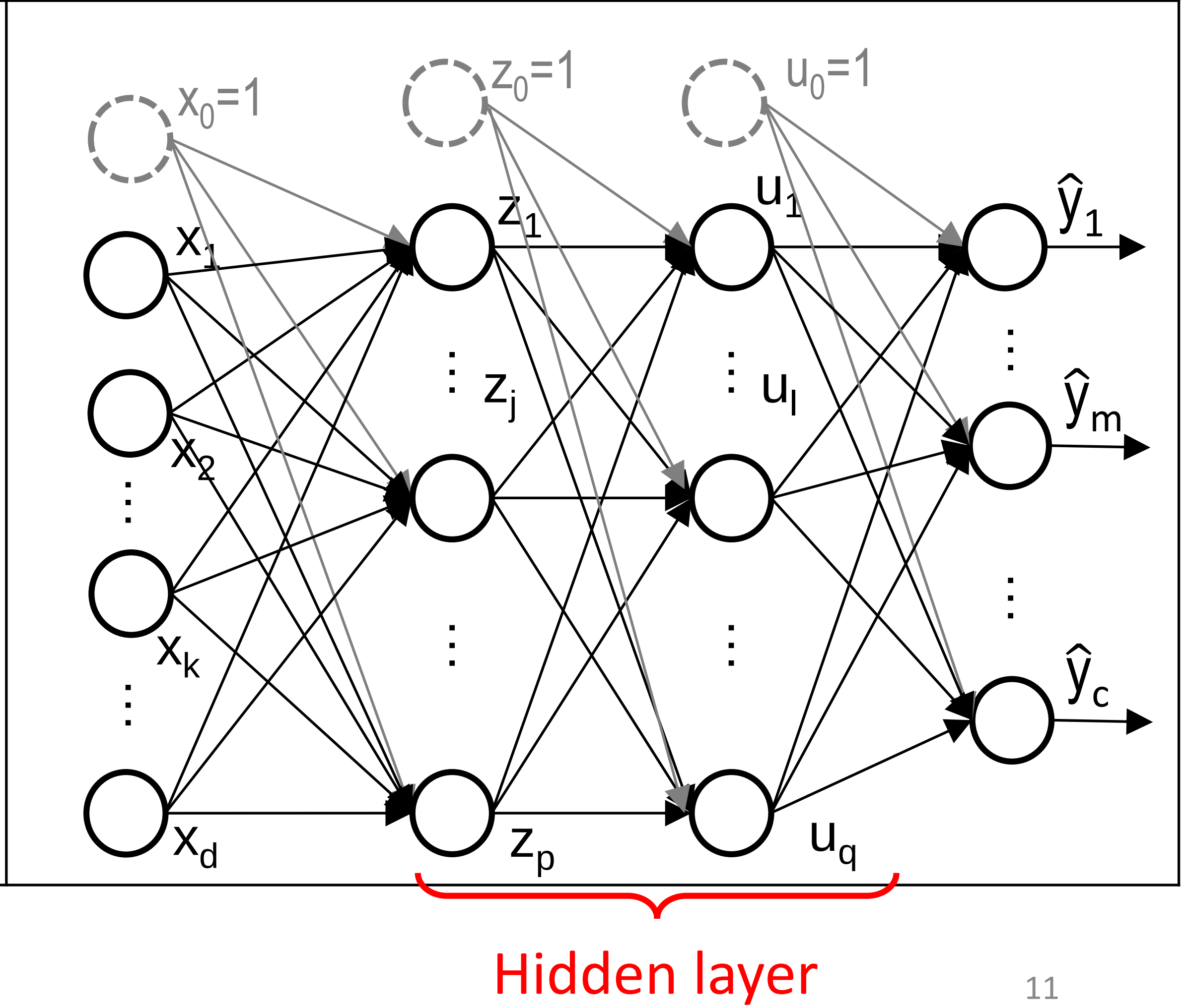
General Structure of MLP

- Please note how the final output configuration looks like

Two layer



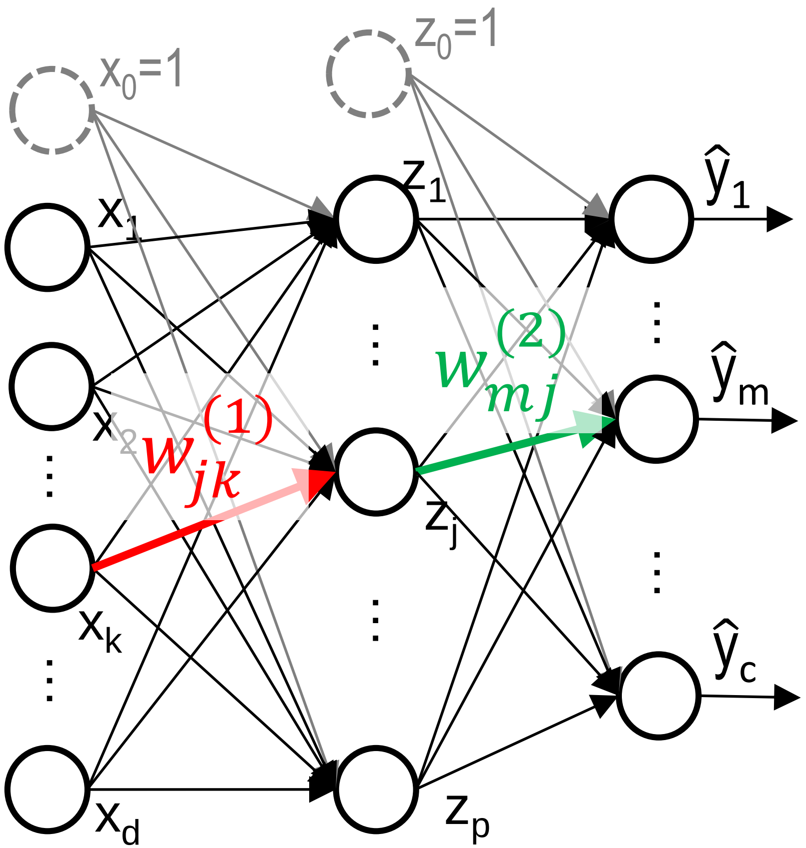
Three layer



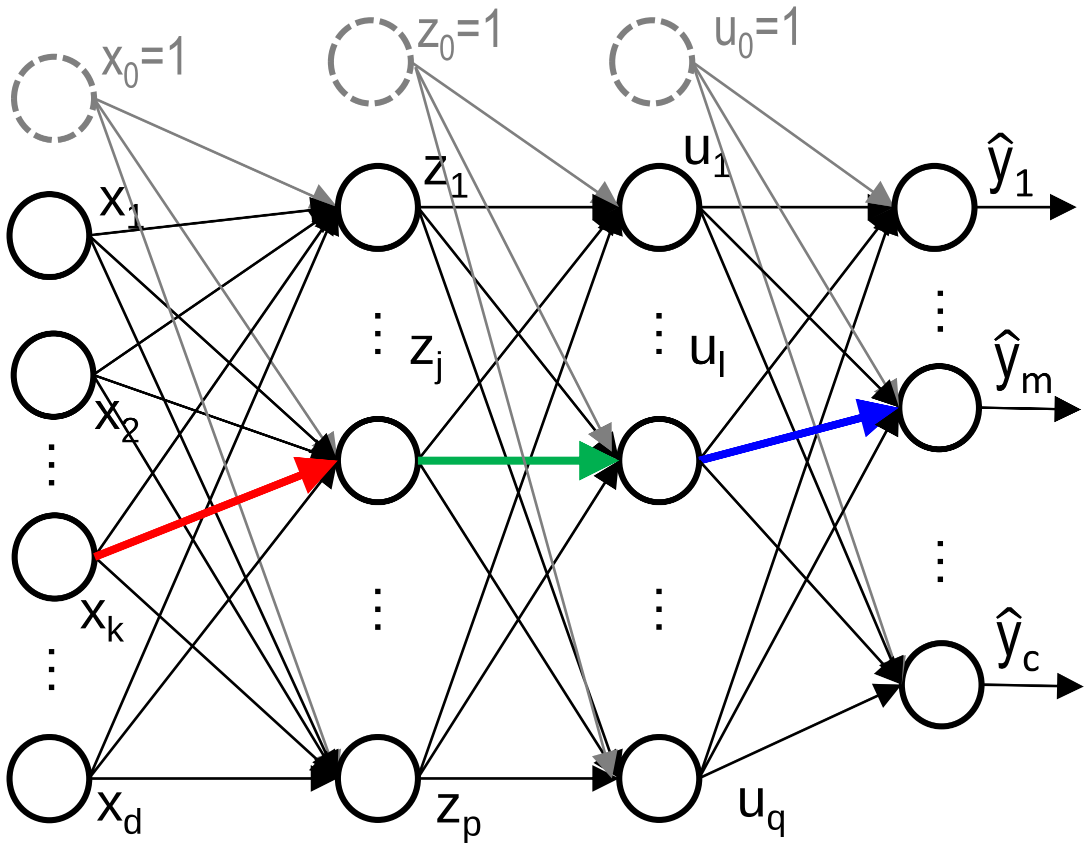
Weight Component Representation

- Please note how the final output configuration looks like

Two layer

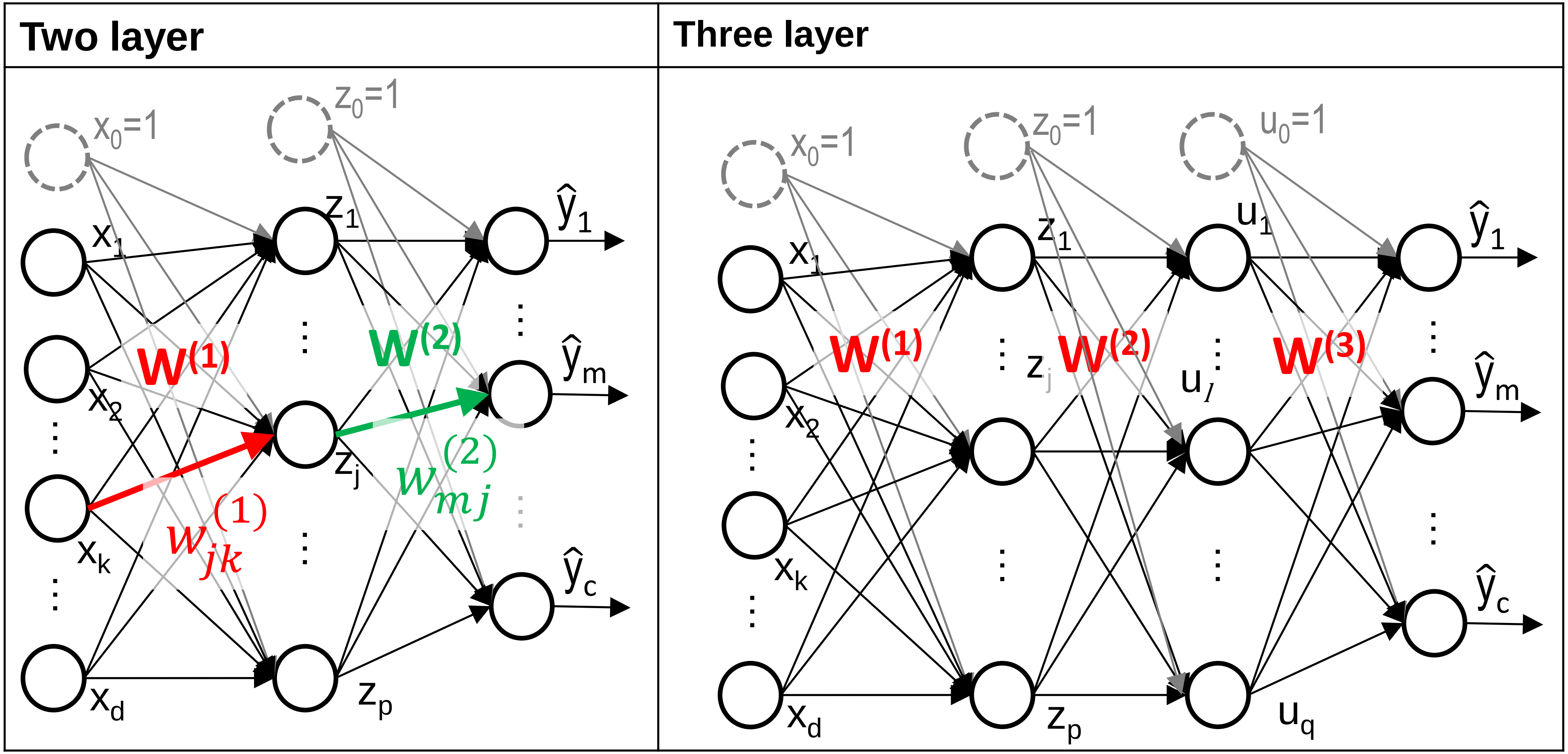


Three layer

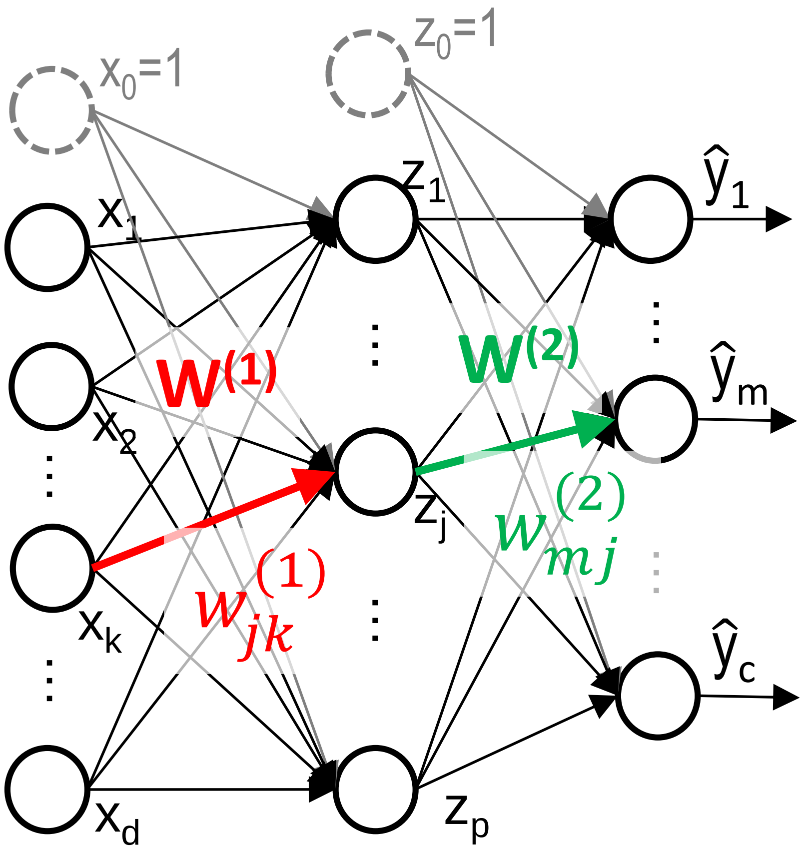


Weight Matrix Representation

- Please note how the final output configuration looks like



Form of W Matrix



$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_k \\ \cdots \\ x_d \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_j \\ \cdots \\ z_p \end{bmatrix}$$

From now on, you just simply think,
"Oh, There is W matrix
which can transform $\vec{x}$ into $\vec{z}$.

Operation Representation in MLP

- For example, $W^{(1)}$ looks like

$$W^{(1)} = \begin{bmatrix} w_{10}^{(1)} & w_{11}^{(1)} & \cdots & w_{1d}^{(1)} \\ w_{20}^{(1)} & w_{21}^{(1)} & \cdots & w_{2d}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p0}^{(1)} & w_{p1}^{(1)} & \cdots & w_{pd}^{(1)} \end{bmatrix}$$

- For two layer perceptron with activation function of f

  1st layer : $z_j = f(w_{j0}x_0 + w_{j1}x_1 + \dots + w_{jk}x_k \ \dots + w_{jd}x_d) = f(\mathbf{w_j} \bullet \mathbf{x})$

  That is, $\mathbf{z} = f(\mathbf{W}^{(1)}\mathbf{x})$

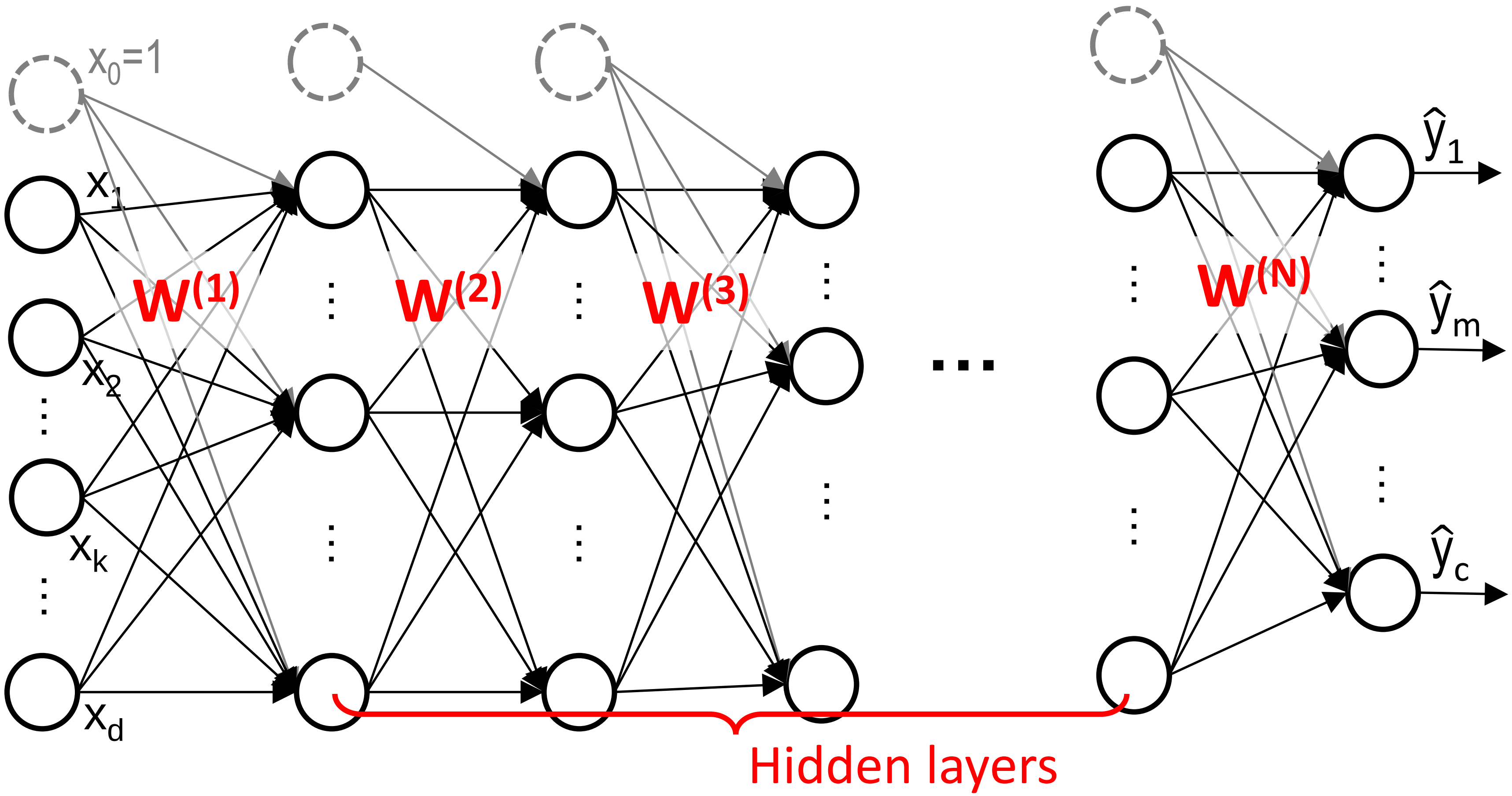
  Output (=2nd) layer : $\hat{\mathbf{y}} = f(W^{(2)}\mathbf{z}) = f(W^{(2)} f(\mathbf{W}^{(1)}\mathbf{x}))$

- How about for three layer perceptron?

Deep MLP

- More than two layers (3, 4, …)
- Not necessarily transform data being linearly separable then classify.

➔ **Arbitrary operations are performed in hidden layers.**



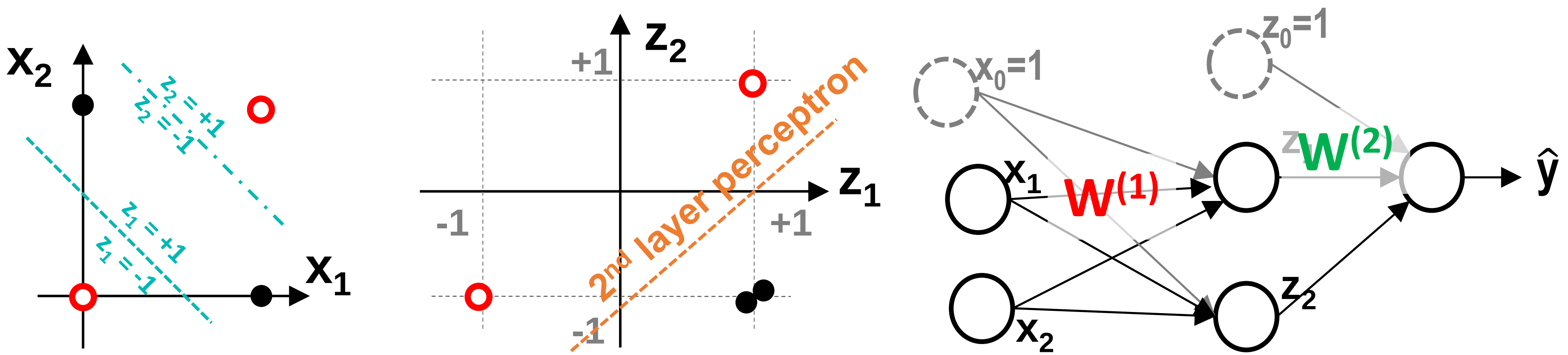
Checkpoints

✓Applying perceptron XOR data through MLP

✓Matrix representation for MLP: $\hat{\mathbf{y}} = f(\mathrm{W}^{(2)} f(\mathbf{W}^{(1)}\mathbf{x}))$

✓Deep MLP

✓Coming up next: how can we determine W? = training!



General Procedure for Training

1) Defining cost/loss function J(**θ**)
2) Then minimizing cost function (MLE vs. MAP)

- MLE cost function is derived based on **p(D/θ)**.
  - Some transform is applied for mathematical convenience.
  - For example, NLL = **-log p(D/θ)** or (½)NLL^2, ….
- Gaussian distribution is used for PDF, minimizing the square of Euclidean distance, $\|\hat{\mathbf{y}} - \mathbf{y}\|^2$, becomes the cost function.
  - $\hat{\mathbf{y}}$=output predicted value by model
  - y=output given by the training set (answer)
- After the cost function is given, differentiation or gradient descent can be used.

$$\boldsymbol{\theta}_{next} = \boldsymbol{\theta}_{present} - \boldsymbol{\eta} \nabla J(\boldsymbol{\theta})$$

Cost Function Development in Training for MLP

- Cost function? For each sample, we can consider the following as cost function.

$$||\widehat{\mathbf{y}_i} - \mathbf{y}_i||^2$$

- How about for all samples? We have the design matrix **X**.
- Saying $\hat{\mathbf{Y}}$ = *F*(**X**) where *F* is overall MLP classifier for given **Y**, then the cost becomes

$$\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|^2 = ||F(\boldsymbol{X}) - \mathbf{Y}||^2$$

- The training is to find **W*** that minimizes the above. That is,

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} ||F(\boldsymbol{X}) - \mathbf{Y}||^2$$

- It should be noted that $F(\boldsymbol{X}) = F(\boldsymbol{X}|\boldsymbol{W})$

Applying GD Method

- Defining a cost function J(**W**) as follows for mathematical convenience.

$$J(\mathbf{W}) = \frac{1}{2}\|F(\boldsymbol{X}) - \mathbf{Y}\|^2 = \frac{1}{2}\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|^2$$

- For MLP, the gradient descent to find W* is

$$W^{(1)} = W^{(1)} - \eta\frac{\partial J}{\partial W^{(1)}}$$

$$W^{(2)} = W^{(2)} - \eta\frac{\partial J}{\partial W^{(2)}}$$

$$W^{(3)} = W^{(3)} - \eta\frac{\partial J}{\partial W^{(3)}}$$

....

- We should find the expression for the above.

Revisit the Location of $W^{(\bullet)}$

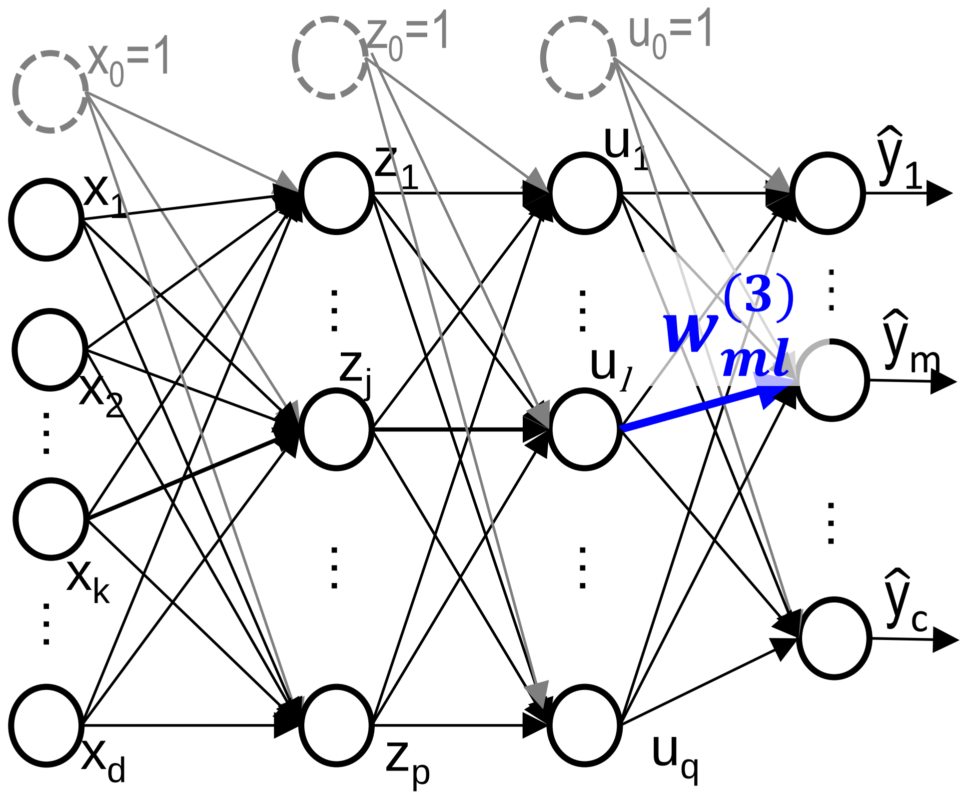
- Which one is easier to find out?

$$W^{(1)} = W^{(1)} - \eta\frac{\partial J}{\partial W^{(1)}}$$

$$W^{(2)} = W^{(2)} - \eta\frac{\partial J}{\partial W^{(2)}}$$

$$W^{(3)} = W^{(3)} - \eta\frac{\partial J}{\partial W^{(3)}}$$

$$J(\mathbf{W}) = \frac{1}{2}\|F(\boldsymbol{X}) - \mathbf{Y}\|^2 = \frac{1}{2}\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|^2$$



Starting From a Component of $\mathbf{W}^{(3)}$

$$w_{ml}^{(3)} = w_{ml}^{(3)} - \eta \frac{\partial \mathbf{J}}{\partial w_{ml}^{(3)}}$$

- To focus on $\frac{\partial \mathbf{J}}{\partial w_{ml}^{(3)}}$,

$$\frac{\partial \mathrm{J}}{\partial w_{ml}^{(3)}} = \frac{1}{2} \frac{\partial (\|\widehat{\mathbf{Y}} - \mathbf{Y}\|^2)}{\partial w_{ml}^{(3)}} = \frac{1}{2} \frac{\partial (\widehat{y_m} - y_m)^2}{\partial w_{ml}^{(3)}}$$

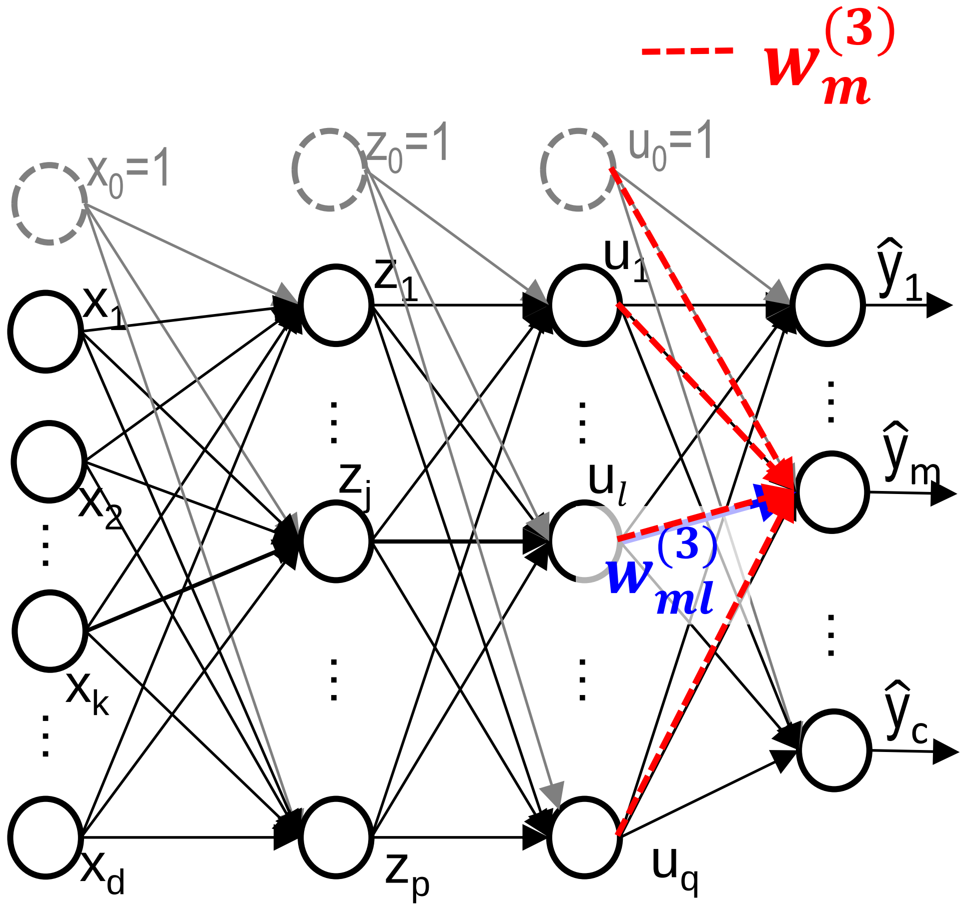
$$= (\widehat{y_m} - y_m) \frac{\partial \widehat{y_m}}{\partial w_{ml}^{(3)}}$$

$\boldsymbol{w_{ml}^{(3)}}$ **vs.** $\boldsymbol{\widehat{y}_m}$**?**
It is in $f\,(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u})$

$$= (\widehat{y_m} - y_m) \frac{\partial f\left(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}\right)}{\partial w_{ml}^{(3)}}$$

We don't know f yet

$$= (\widehat{y_m} - y_m) f'\left(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}\right) u_l$$



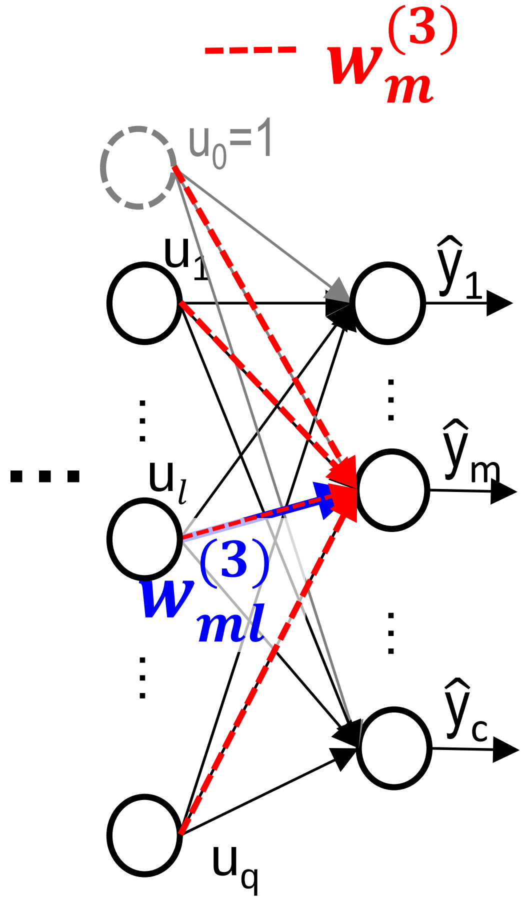
GD for $W^{(3)}$

- For each m, δ_m can be defined, thus

$$\frac{\partial \mathbf{J}}{\partial w_{ml}^{(3)}} = (\widehat{y_m} - y_m) f'\left(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}\right) u_l = \delta_m u_l$$

- Defining $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_C]^T$ and $\mathbf{u} = [u_1, \ldots, u_q]^T$

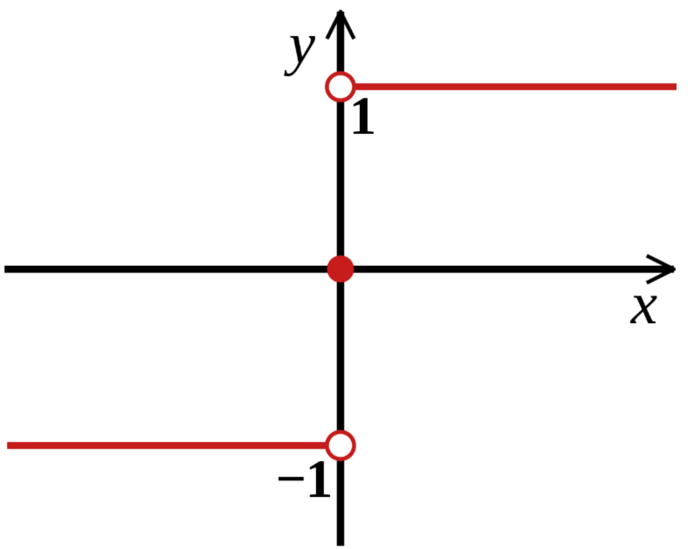
$$\mathbf{W}^{(3)} = \mathbf{W}^{(3)} - \eta \frac{\partial \mathbf{J}}{\partial \boldsymbol{W}^{(3)}} = \mathbf{W}^{(3)} - \eta \boldsymbol{\delta} \mathbf{u}^T$$

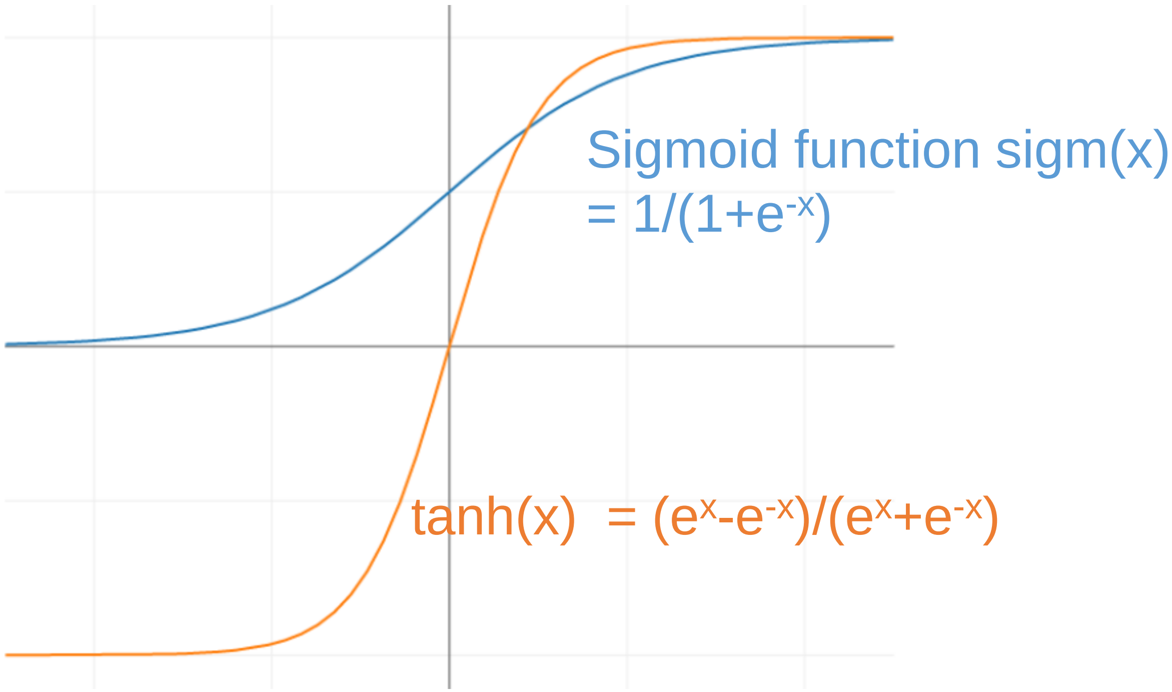


We Need f'(•); Choice of Activation Function

- We need differentiable activation function such as sigmoid or tanh, instead of sign function.

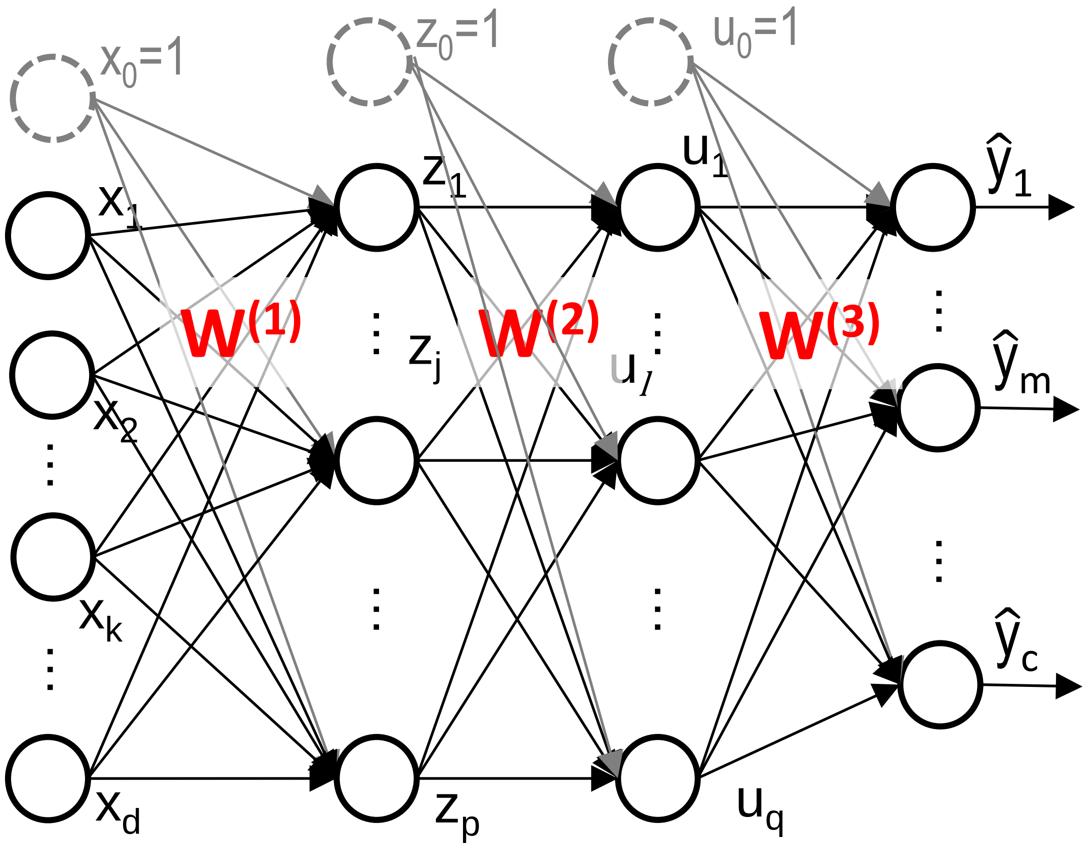
f(x) as sign function or sgn(x)





What We Did is

$$\mathbf{W}^{(3)} = \mathbf{W}^{(3)} - \eta \frac{\partial \mathbf{J}}{\partial \boldsymbol{W}^{(3)}} = \mathbf{W}^{(3)} - \eta \boldsymbol{\delta} \mathbf{u}^{\mathrm{T}}$$



For prediction,

$\hat{\mathbf{y}} = f(\mathbf{W}^{(3)} f(\mathbf{W}^{(2)} f(\mathbf{W}^{(1)} \mathbf{x})))$

How About $W^{(2)}$?

It may affect all outputs

$$\frac{\partial J}{\partial w_{lj}^{(2)}} = \frac{1}{2}\frac{\partial(\|\hat{\mathbf{Y}}-\mathbf{Y}\|^2)}{\partial w_{lj}^{(2)}} = \frac{1}{2}\frac{\partial \sum_{m=1}^{c}(\widehat{y_m}-y_m)^2}{\partial w_{lj}^{(2)}}$$

$$=\sum_{m=1}^{c}(\widehat{y_m}-y_m)\frac{\partial \widehat{y_m}}{\partial w_{lj}^{(2)}}$$

$w_{lj}^{(2)}$ vs. $\hat{y}_m$?

It is in $f\,(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u})$

$$=\sum_{m=1}^{c}(\widehat{y_m}-y_m)\frac{\partial f\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)}{\partial w_{lj}^{(2)}}$$

$$=\sum_{m=1}^{c}(\widehat{y_m}-y_m)\,f'\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)\frac{\partial\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)}{\partial w_{lj}^{(2)}}$$

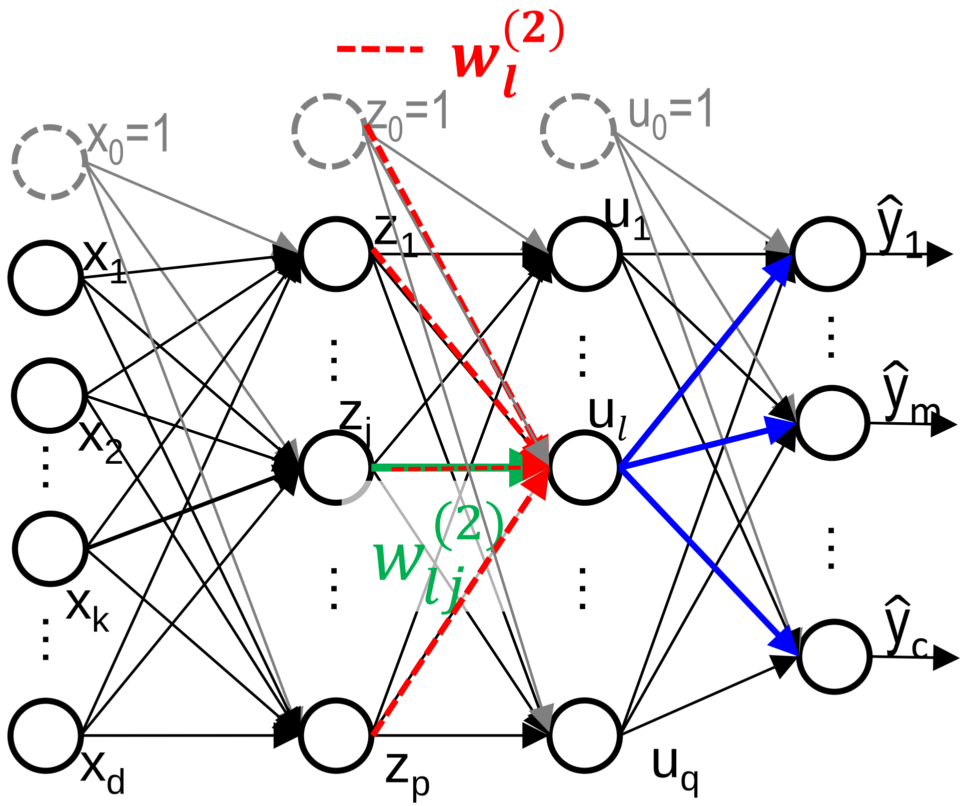
$w_{lj}^{(2)}$ vs. $\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}$?

$w_{lj}^{(2)} \rightarrow f\,(\boldsymbol{w_l^{(2)}} \bullet \boldsymbol{z}) = \boldsymbol{u_l} \rightarrow \boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}$

$$=\sum_{m=1}^{c}\left\{(\widehat{y_m}-y_m)f'\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)\ w_{ml}^{(3)}\ \right\}\frac{\partial u_l}{\partial w_{lj}^{(2)}}$$

$$=\frac{\partial f(\boldsymbol{w_l^{(2)}}\bullet\boldsymbol{z})}{\partial w_{lj}^{(2)}}\sum_{m=1}^{c}\left\{(\widehat{y_m}-y_m)f'\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)w_{ml}^{(3)}\right\}$$

$$= z_j\, f'(\boldsymbol{w_l^{(2)}}\bullet\boldsymbol{z})\sum_{m=1}^{c}\left\{(\widehat{y_m}-y_m)f'\left(\boldsymbol{w_m^{(3)}}\bullet\boldsymbol{u}\right)w_{ml}^{(3)}\right\}$$



GD for $W^{(2)}$

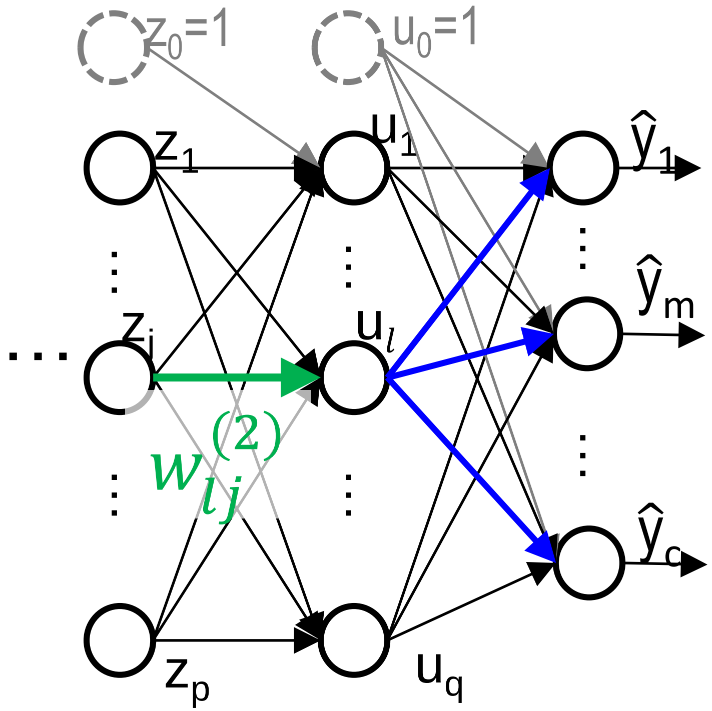
$$\frac{\partial J}{\partial w_{lj}^{(2)}} = z_j\, f'(\boldsymbol{w_l^{(2)}} \bullet \boldsymbol{z}) \sum_{m=1}^{c} \left\{ (\widehat{y_m} - y_m) f'\left(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}\right) w_{ml}^{(3)} \right\}$$

- For each l, γ_l can be defined as

$$\gamma_l = f'(\boldsymbol{w_l^{(2)}} \bullet \boldsymbol{z}) \sum_{m=1}^{c} \left\{ (\widehat{y_m} - y_m) f'\left(\boldsymbol{w_m^{(3)}} \bullet \boldsymbol{u}\right) w_{ml}^{(3)} \right\}$$
$$= f'(\boldsymbol{w_l^{(2)}} \bullet \boldsymbol{z}) \sum_{m=1}^{c} \left\{ \delta_m w_{ml}^{(3)} \right\}$$

- Then,

$$\frac{\partial \mathbf{J}}{\partial w_{lj}^{(2)}} = \gamma_l z_j$$



- Defining $\boldsymbol{\gamma} = [\gamma_1, \dots, \gamma_q]^T$ & $\mathbf{z} = [z_1, \dots, z_p]^T$

$$\mathbf{W}^{(2)} = \mathbf{W}^{(2)} - \eta \frac{\partial \mathbf{J}}{\partial \boldsymbol{W}^{(2)}} = \mathbf{W}^{(2)} - \eta \boldsymbol{\gamma} \mathbf{z}^T$$

You should get a feel for the terms

We Can Extend it to $W^{(1)}$

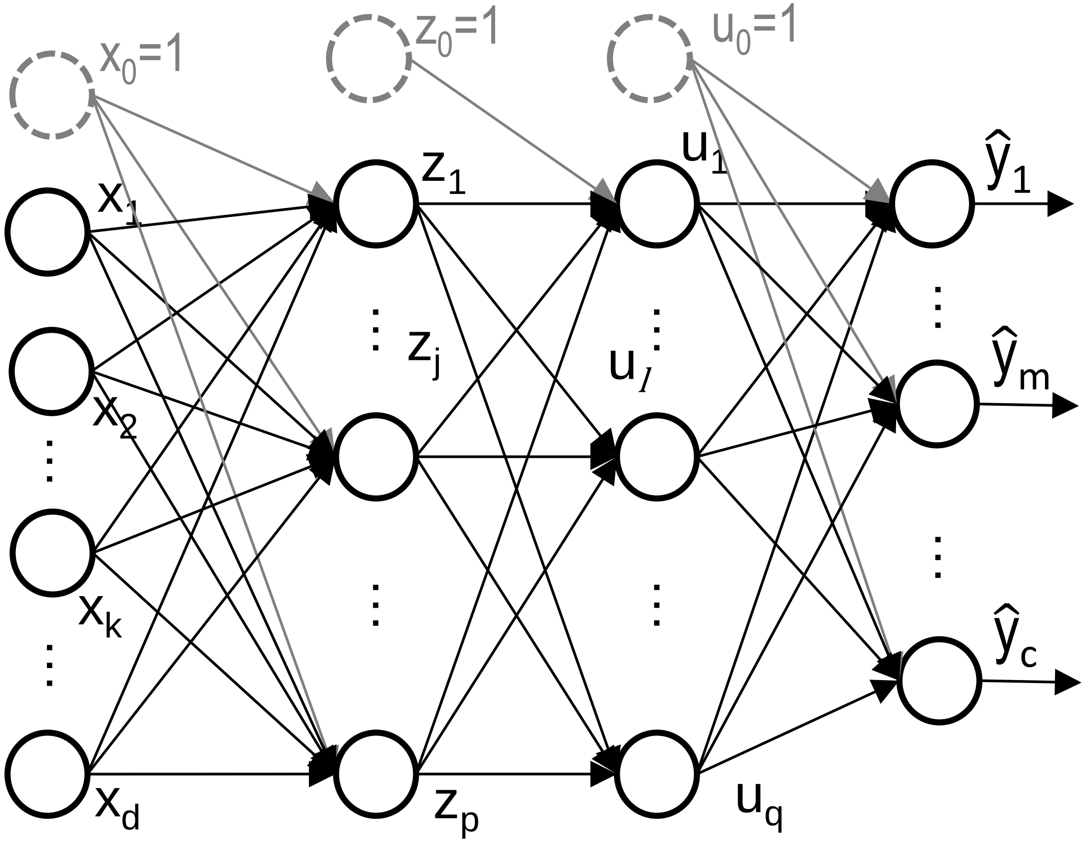
- Calculating β for each j,

$$\beta_j = f'(\boldsymbol{w}_{\boldsymbol{j}}^{(\mathbf{1})} \bullet \boldsymbol{x}) \sum_{l=1}^{q} \left\{ \gamma_l w_{lj}^{(2)} \right\}$$

- Then, with a vector definition $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots \beta_p)$

$$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - \eta \frac{\partial \mathbf{J}}{\partial \boldsymbol{W}^{(1)}} = \mathbf{W}^{(1)} - \eta \boldsymbol{\beta} \mathbf{x}^{\mathbf{T}}$$

Direction?



Training Algorithm (Mini-batch version)

Initialize $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$ properly

Until (No change):

Extracting n' samples from **X** to form **X'**

$\Delta\mathbf{W}^{(1)} = 0$, $\Delta\mathbf{W}^{(2)} = 0$, $\Delta\mathbf{W}^{(3)} = 0$

for each comp of **X'**:

$\mathbf{z}=f(\mathbf{W}^{(1)}\mathbf{x})$

$\mathbf{u}=f(\mathbf{W}^{(2)}\mathbf{z})$

$\mathbf{yh}=f(\mathbf{W}^{(3)}\mathbf{u})$

$\boldsymbol{\delta} = (\mathbf{yh}-\mathbf{y}) \times f'(\mathbf{W}^{(3)}\mathbf{u})$ #c×1

$\boldsymbol{\gamma} = \boldsymbol{\delta} \bullet \widetilde{\boldsymbol{W}}^{(3)} \times f'(\mathbf{W}^{(2)}\mathbf{z})$ #q×1

$\boldsymbol{\beta} = \boldsymbol{\gamma} \bullet \widetilde{\boldsymbol{W}}^{(2)} \times f'(\mathbf{W}^{(1)}\mathbf{z})$ #p×1

$\mathbf{W}^{(3)} = \mathbf{W}^{(3)} - (1/n')\eta\boldsymbol{\delta}\mathbf{u}^T$

$\mathbf{W}^{(2)} = \mathbf{W}^{(2)} - (1/n')\eta\boldsymbol{\gamma}\mathbf{z}^T$

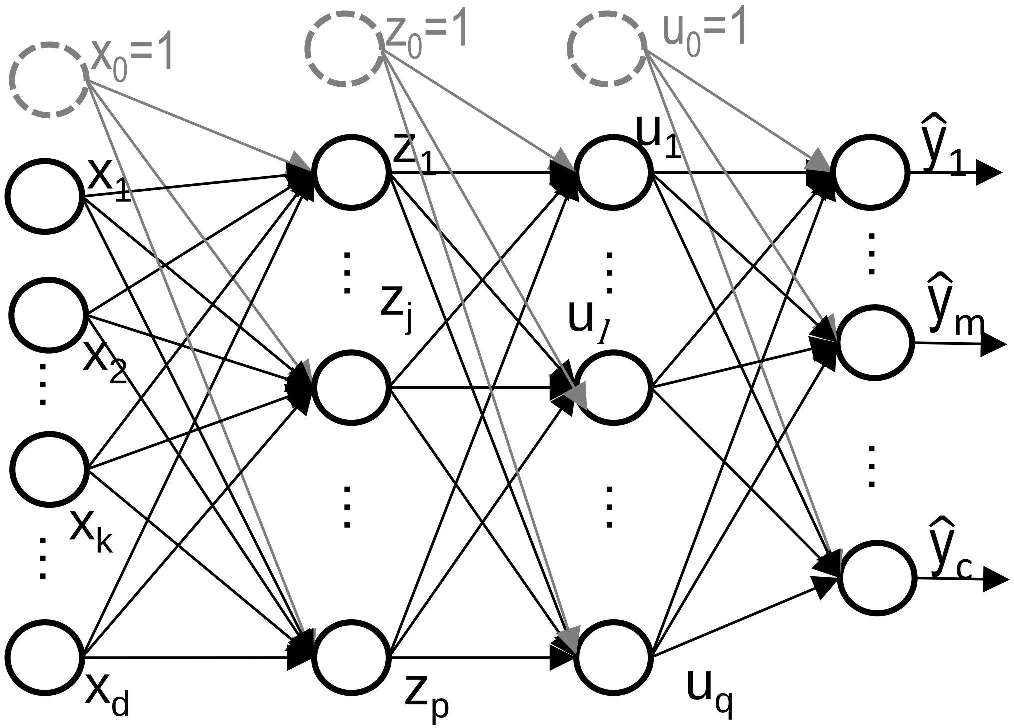
$\mathbf{W}^{(1)} = \mathbf{W}^{(1)} - (1/n')\eta\boldsymbol{\beta}\mathbf{x}^T$

You can see the W is determined in backward

➔ **Called error back(ward) propagation**

Prediction

- At the output stage, the softmax function is utilized for deriving probability (soft decision)
- For hard decision, **m*** = $\underset{m}{\text{argmax}}\ \hat{y}_m$



Softmax function

$$\sigma(\hat{y}_m) = \frac{\exp(\hat{y}_m)}{\sum_{k=1}^{c} \exp(\hat{y}_k)}$$

[2, 0.4, 0.2]

➔ [0.73, 0.15, 0.12]

➔ Called error forward propagation

Limitation of Deep Layer

- Large computation
- Vanishing/Exploding gradient
- Overfitting risks

Enable of Deep Layer: Deep Learning

- Large computation

  ➔Development of high performance hardware

  ➔Use of more efficient algorithm such as CNN

- Vanishing/Exploding gradient

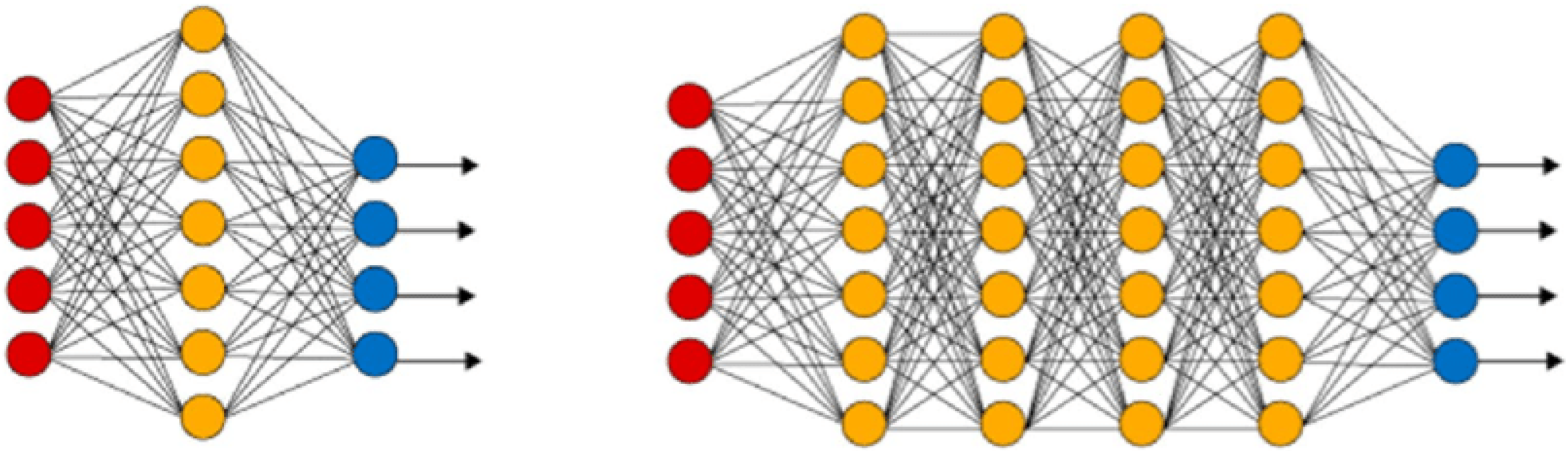
  ➔Use of improved activation function such as RELU or its variants

  ➔Use of cross-entropy or LLE for cost function

- Overfitting risks

  ➔ Development of various regulation techniques

Deep Learning from
Deep MLP (Deep MLP)



Package for Deep Learning in Python; Tensorflow

- Using tensorflow developed by google
  - Compatible with scklearn (tf.learn)
  - Various high-level API such as Keras or Pretty Tensor
  - Visualization is easy with TensorBoard
- https://www.tensorflow.org/
- Install tensorflow package by typing in Anaconda prompt:

  conda install tensorflow

Checkpoints

- ✓Backward propagation for training MLP
- ✓Deep learning and its limitation
- ✓Coming up next: Tensorflow for python coding MLP