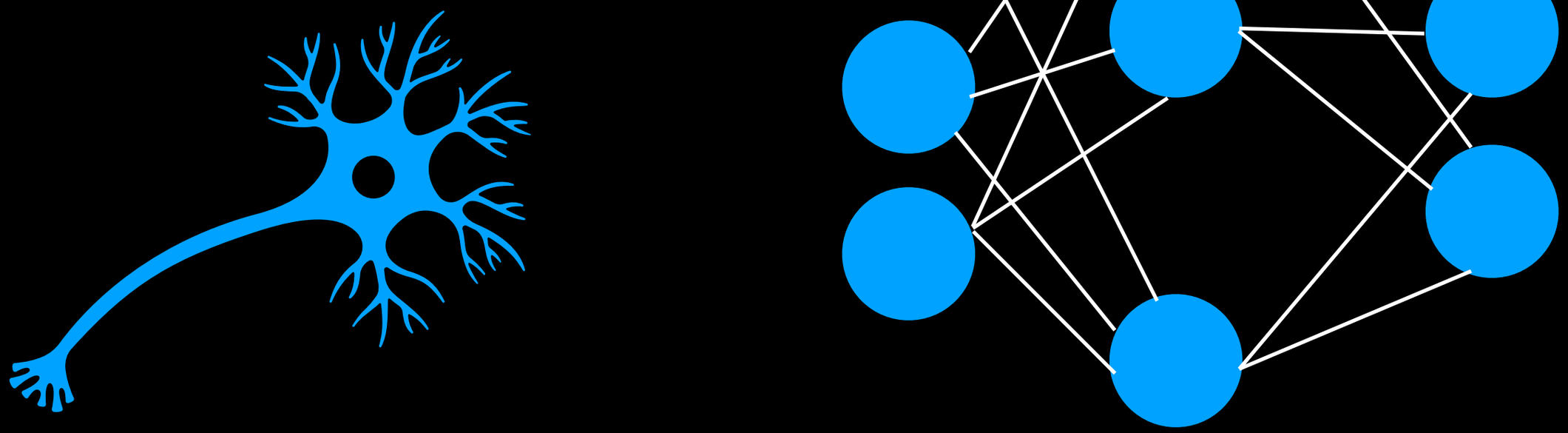
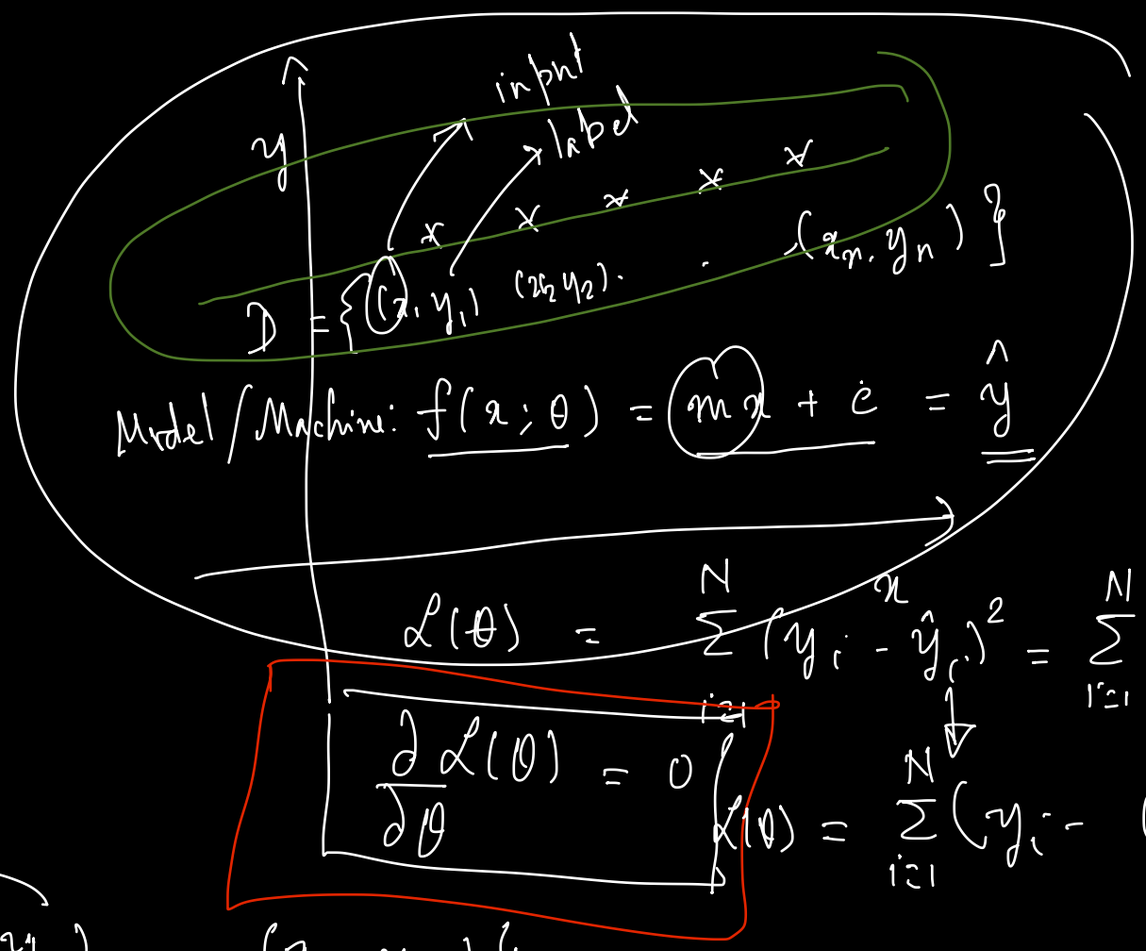


Neural Networks

Sumohana S. Channappayya

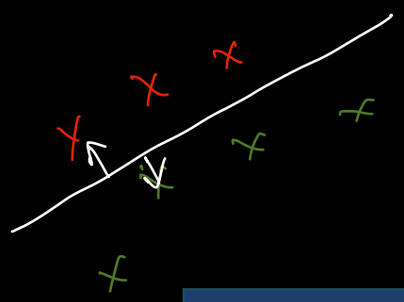
Review

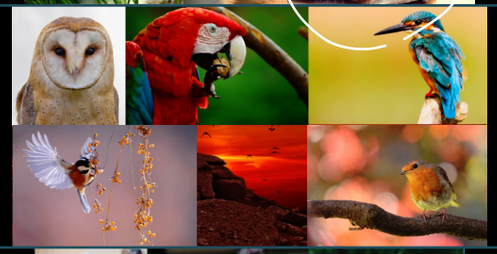
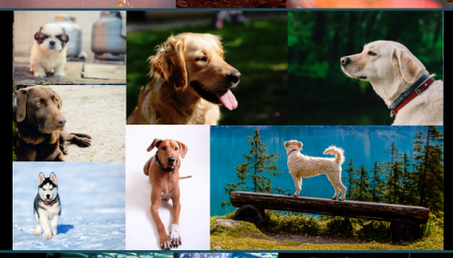
- Machine Learning
- **Supervised**
- Unsupervised
- Reinforcement



- SL: $D = \{(x_1, y_1), (x_2, y_2) \ldots (x_{N_T}, y_{N_T})$ (training) $\ldots (x_N, y_N)\}$ (validation / testing)
  - $\hat{y} = f(x;\theta)$ ← machine model.
  - $\mathcal{L}(\theta) = \sum_{i=1}^{I} d(y_i, \hat{y}_i)$; $\theta^* = \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}(\theta)$ ◦ Evaluate model on validation set

Review: Supervised Learning



Input (**x**)	Label (y)	
	Cat	0
	Bird	1
	Dog	2
	Fish	3
	?	

$x = 32 \times 32 \times 3$

$\hat{y} = f(x; \theta)$?

$\hat{y} = \sum_{i=1}^{N} \sum_{j=1}^{M} W(i,j) \cdot 4$ parameters

$X(i,j)$

$\hat{y} = f(x; \theta)$

= NLO Linear

$= \sigma\left(\sum_{i}^{N} W_i X_i + W_0\right)$

Review: Supervised Learning

- Given a set of **input points** and corresponding **ground truth labels:** $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$
  - Discrete labels: **classification**
  - Continuous labels: **regression**
- Learn a **parameterised (θ) functional relationship** to **estimate label** from input: $\hat{y} = f(\mathbf{x}; \theta)$
- Makes use of **training data** to learn parameters $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

Review: Supervised Learning

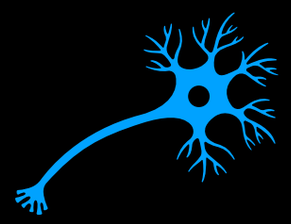
- How to learn parameters $\boldsymbol{\theta}$?
  - Quantify error: $d(y, \hat{y})$
  - Find **error** between estimate and ground truth over **training data** points: $R(\boldsymbol{\theta}) = \sum_{i=1}^{n} d(y_i, \hat{y}_i)$
  - Find parameters that **minimise error**
- How **good** is our **model**?
  - Evaluate model's performance on test data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$

Review: Supervised Learning

- Models covered so far:
  - Linear regression
  - k-NN
  - Naive Bayes
  - Logistic regression
  - Support Vector Machines (SVM) ✓
  - Neural Network

Motivation for Neural Networks

- Modelling the **neuron**



- Modelling **nonlinear** relation between $\{\mathbf{x}_i\}$ and $\{y_i\}$

XOR

$f(x_1, x_2)$

$= \boxed{XOR(x_1, x_2)}$

$\hat{y} = w_1 x_1 + w_2 x_2$

$d(y, \hat{y})$ ↓

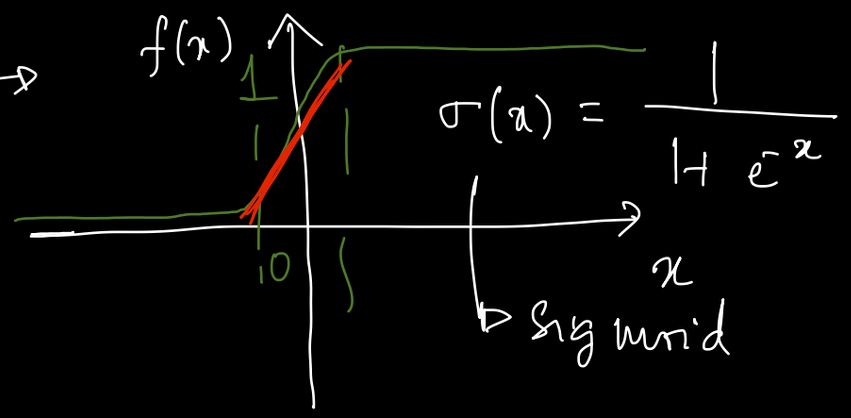
Input ($\mathbf{x}$)		Label (y)
0	0	0
0	1	1
1	0	1
1	1	0

Can we find a model for the XOR gate?

Yes. Neural Networks

A Neuron

$y \in \{0, 1\}$



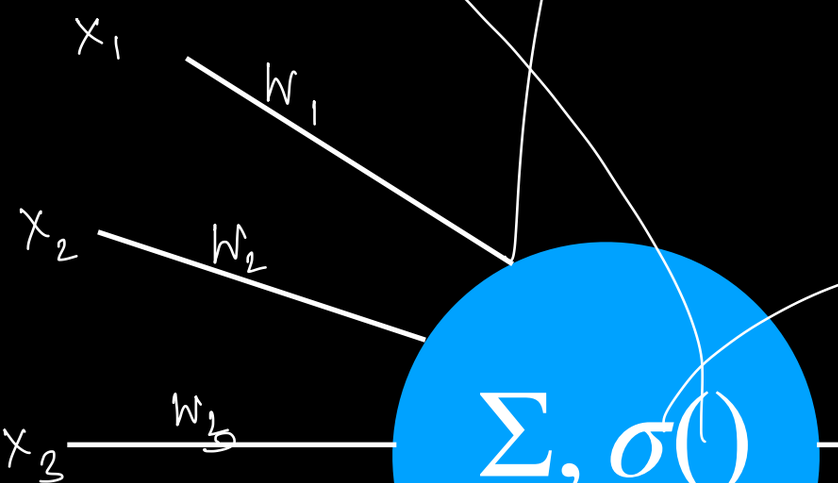
$\sigma(x) = \frac{1}{1 + e^{-x}}$

$f = g \circ h$

$d(y, \hat{y})$

GT prediction

$\underline{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_5 \end{bmatrix}$



$L(\theta) = \sum_{i=1}^{N} d(y_i, \hat{y}_i)$

$\frac{\partial L(\theta)}{\partial \theta}$

Output

$\hat{y}_L = \sum_{i=1}^{5} w_i x_i + w_0$

$\hat{y} = \sigma(\hat{y}_L)$

$\underline{\hat{y}} = \sigma\left(\sum_{i=1}^{5} w_i x_i + w_0\right)$

bias

$\underline{X}' = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_5 \end{bmatrix}$

$x_0 = 1$

$\hat{y} = \sigma(\langle w', x' \rangle)$

$\underline{w}' = \begin{bmatrix} w_0 \\ \vdots \\ w_5 \end{bmatrix}$

$\hat{y} = \sigma\left(\sum_{i=0}^{5} w_i x_i'\right)$

Key function: $\sigma()$

- Recall motivation: **nonlinear** input output relationship

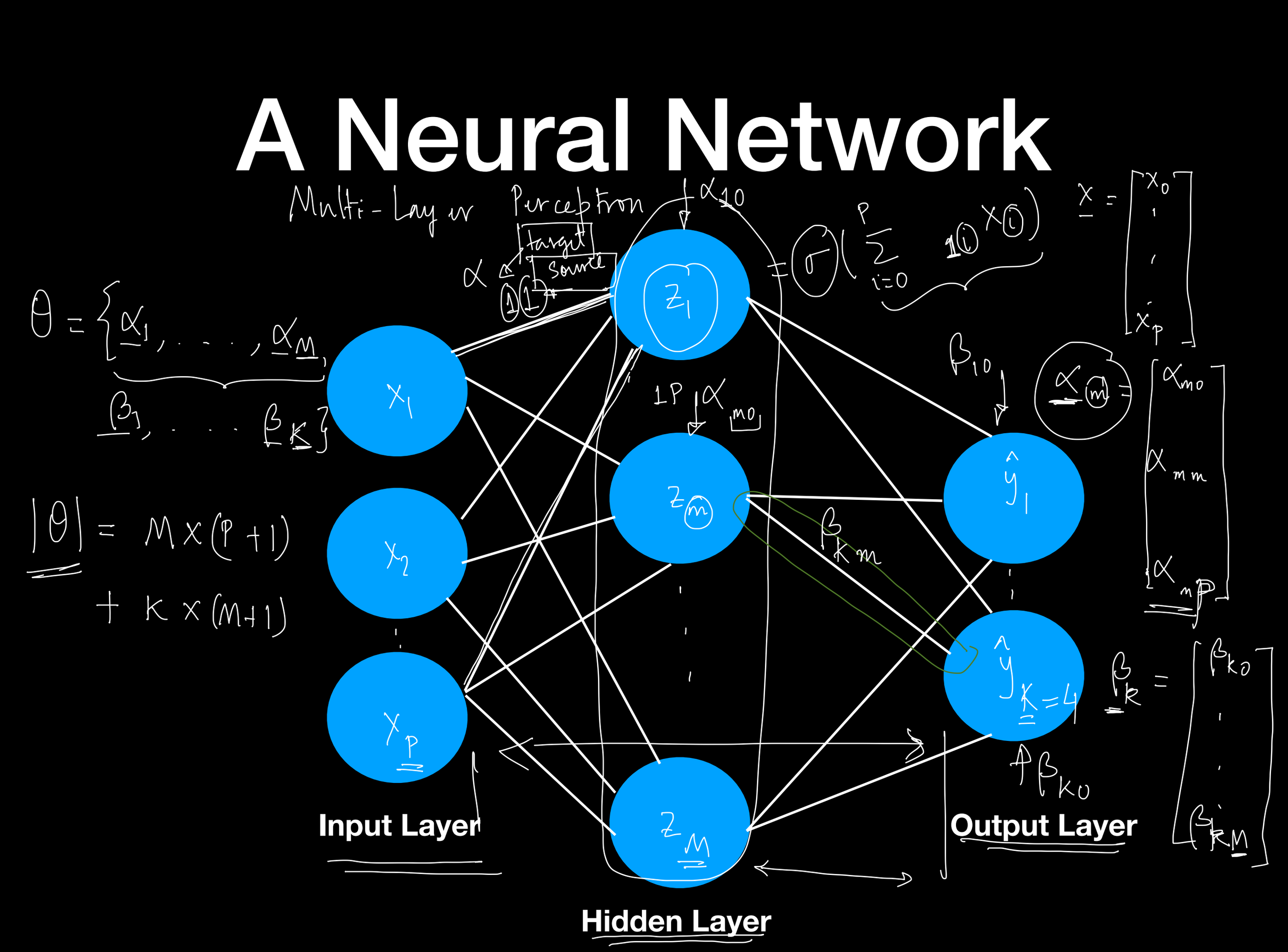
$$\hat{y} = \sigma\left(\sum_{i=0}^{p} w_i x_i\right)$$

$$\underline{x} = [1, x_1, \ldots, x_p]^T$$

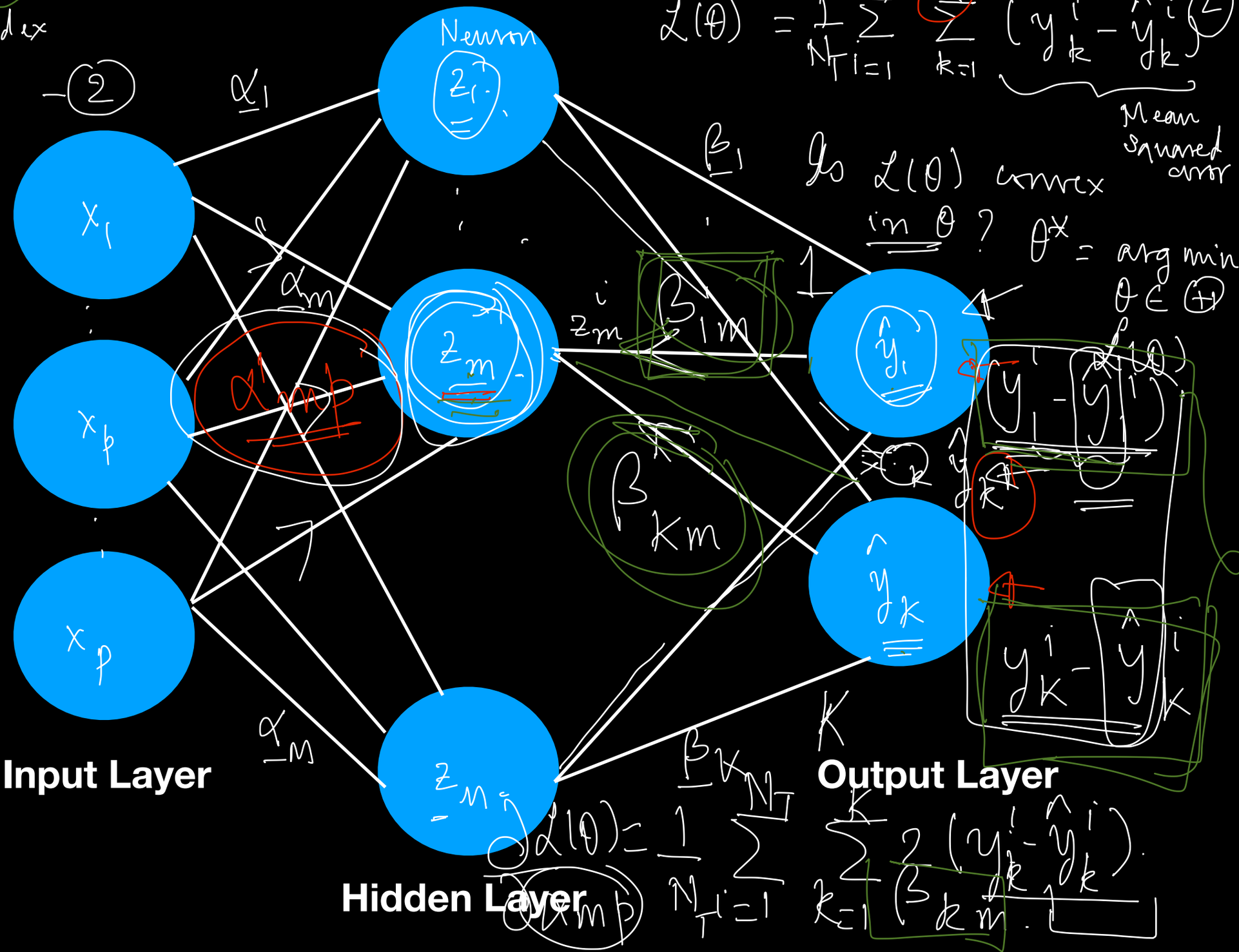
$$\underline{w} = [w_0, w_1, \ldots, w_p]^T$$

- Properties of $\sigma()$ — nonlinearity
  - Nonlinear ✓
  - Continuous ✓
  - Differentiable? ✓
  - Examples: tanh, sigmoid, ReLU etc.

A Neural Network



Feedforward Neural Network

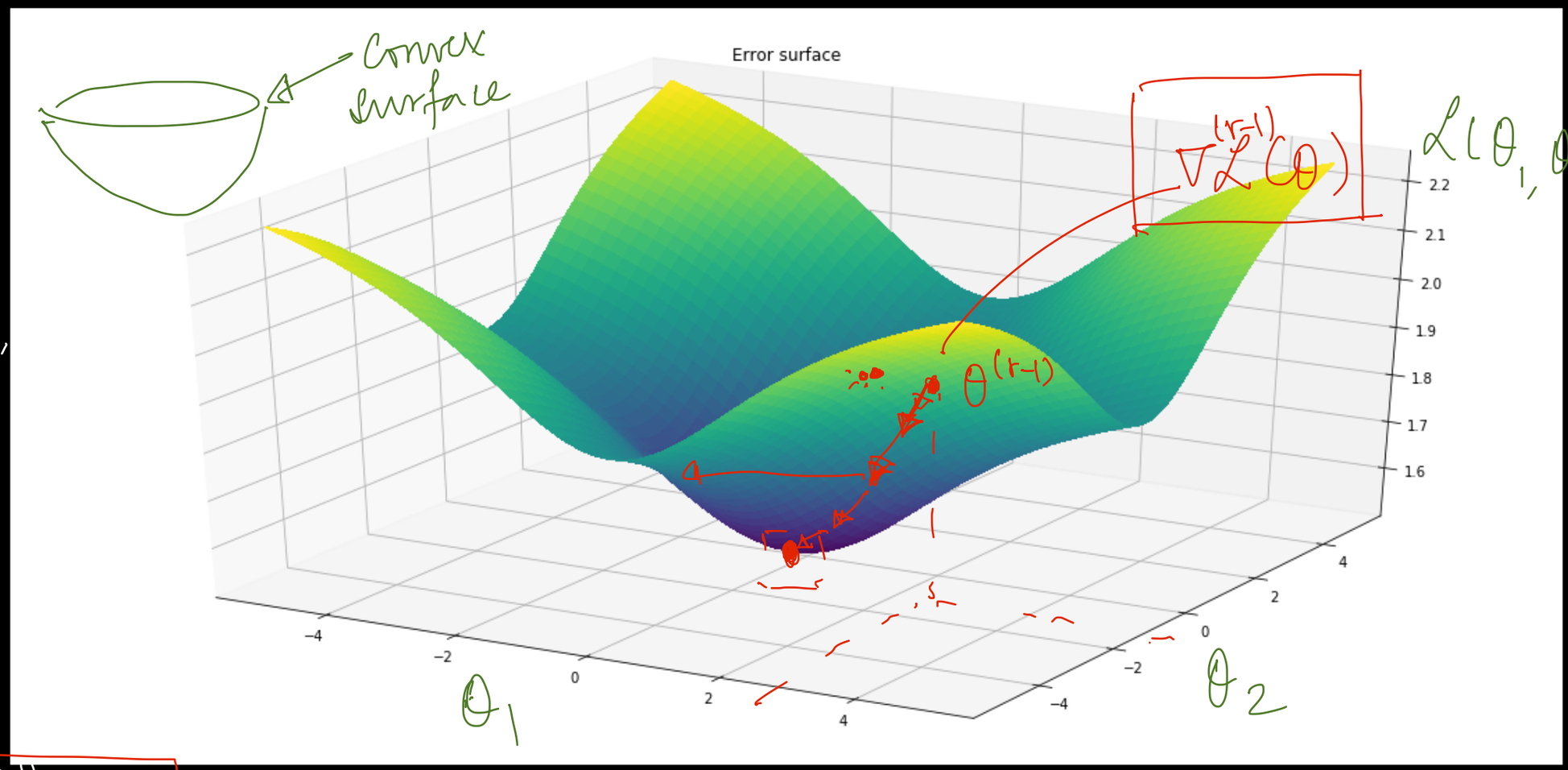


Error Surface

- $\mathcal{L}(\theta)$ is not convex in $\theta \Rightarrow$ no closed form solution ✓
  This is due to the non-linearity present in our model.

- θ^* is not going to global

- Iterative, gradient based solution



$$\theta^{(r)} = \theta^{(r-1)} - \eta \nabla\mathcal{L}(\theta)^{(r-1)}$$

$$\theta^* = \underset{=}{A}^{-1} \cdot \underset{=}{X}$$

$\nabla\mathcal{L}(\theta)$?

Minimising Error: $\frac{\partial R(\boldsymbol{\theta})}{\partial \theta_i}$

- Find local minimum ✓
- Iterative procedure ✓
- Gradient descent ✓
- Let’s derive the gradient update expression

Minimising Error: $\frac{\partial R(\theta)}{\partial \theta_i}$

$$\nabla_\theta \mathcal{L}(\theta) = \begin{bmatrix} \frac{\partial \mathcal{L}(\theta)}{\partial \beta_{10}} \\ \frac{\partial \mathcal{L}(\theta)}{\partial \beta_{1p}} \\ \frac{\partial \mathcal{L}(\theta)}{\partial \beta_{20}} \\ \frac{\partial \mathcal{L}(\theta)}{\partial \beta_{2m}} \\ \vdots \\ \frac{\partial \mathcal{L}(\theta)}{\partial \beta_{km}} \end{bmatrix}$$

$$\mathcal{L}(\theta) = \frac{1}{N_T} \sum_{i=1}^{N_T} \sum_{k=1}^{K} (y_k^i - \hat{y}_k^i)^2 = \frac{1}{N_T} \sum_{i=1}^{N_T} \sum_{k=1}^{K} (y_k^i - g_k(\underline{T}^i))^2$$

- Derivation:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \beta_{km}} = \frac{\partial}{\partial \beta_{km}} \frac{1}{N_T} \sum_{i=1}^{N_T} \frac{\mathcal{L}(\theta)}{(y_k^i - \hat{y}_k^i)^2} = \frac{1}{N_T} \sum_{i=1}^{N_T} \left(\sum_{k=1}^{K} \right) (y_k^i - g_k(\begin{bmatrix} \langle \beta_1, z^i \rangle \\ \langle \beta_k, z^i \rangle \\ \vdots \\ \langle \beta_K, z^i \rangle \end{bmatrix}))^2$$

$(\because \beta_{km}$ affects only $\hat{y}_k)$

$$= \frac{1}{N_T} \sum_{i=1}^{N_T} -2(y_k^i - \hat{y}_k^i) \cdot \frac{\partial \hat{y}_k^i}{\partial \beta_{km}} \quad - (1)$$

Now, let's find $\frac{\partial \hat{y}_k^i}{\partial \beta_{km}} = \frac{\partial}{\partial \beta_{km}} g_k(\underline{T}^i)$

$$= \underline{g_k'(\underline{T}^i)} \cdot \frac{\partial}{\partial \beta_{km}} \langle \underline{\beta}_k, \underline{z}^i \rangle = g_k'(\underline{T}^i) \cdot \frac{\partial}{\partial \beta_{km}} (\beta_{k0} \cdot z_0^i + \ldots + \beta_{km} z_m^i + \ldots + \beta_{kM} z_M^i) \quad - (2)$$

($\beta_{k0} \cdot z_0^i$: 0)

$$\boxed{\frac{\partial \hat{y}_k^i}{\partial \beta_{km}} = g_k'(\underline{T}^i) \cdot z_m^i}$$

Plug (2) into (1) gives us

$$\frac{\partial \mathcal{L}(\theta)}{\partial \beta_{km}} //$$

Minimising Error: $\frac{\partial R(\theta)}{\partial \theta_i}$

- Derivation:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \alpha_{mp}} = -\frac{1}{N_T} \sum_{i=1}^{N_T} \sum_{k=1}^{K} 2 \left(y_k^i - \hat{y}_k^i\right) g_k'(T^i) \beta_{km} g_m'(\alpha_m^T x^i) x_p^i$$

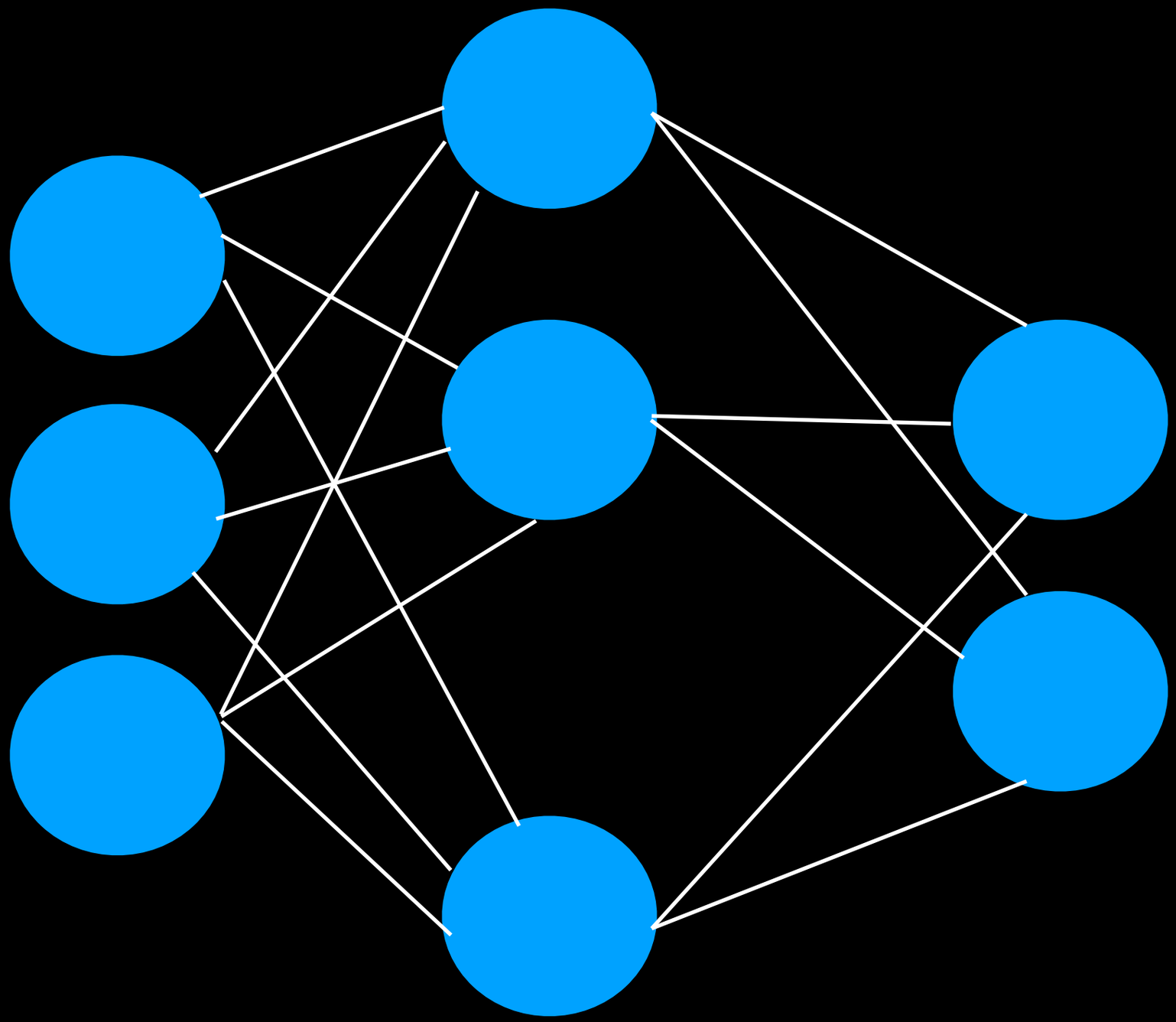
$$\theta^{(r)} = \theta^{(r-1)} - \eta \nabla \mathcal{L}(\theta)^{(r-1)}$$

found using back propagation.

Back propagation

- Neuron $\quad \hat{y} = \sigma(\langle \underline{w}, \underline{x} \rangle)$
- A multilayer perceptron
- $\hat{y} = f(\underline{x}; \theta)$
- $\mathcal{L}(\theta)$
- $\nabla \mathcal{L}(\theta)$: back propagation
- $\underline{\theta}^{(r)} = \underline{\theta}^{(r-1)} - \eta \nabla^{(r-1)} \mathcal{L}(\theta)$

Back propagation



Challenges

- Initialization : ensure weights are not 0. Random init.
- Overfitting : $|\theta|$ grows with # neurons
- Input range : Large dynamic range.
- Network architecture : Choice of neurons not straightforward
- Multiple minima # data points, I/O relationship