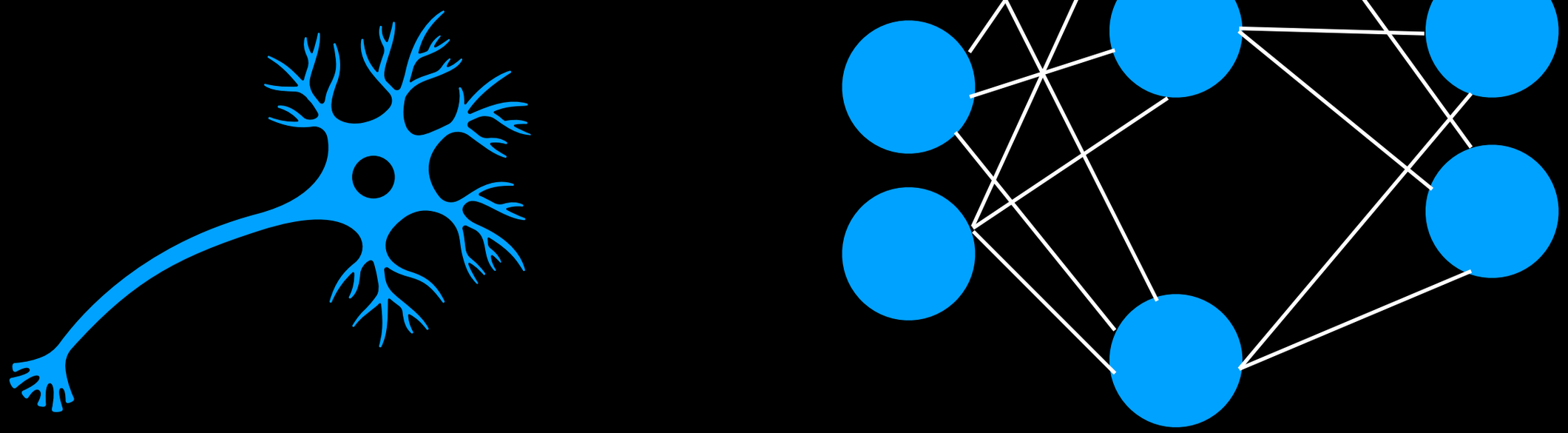


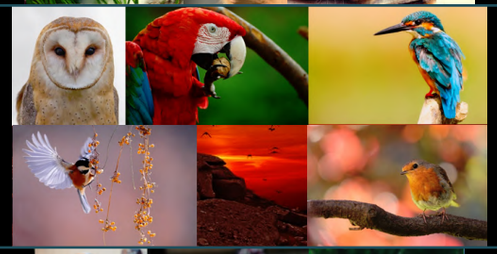
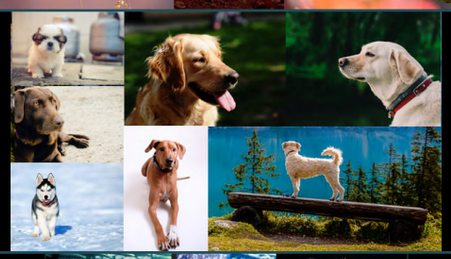
Neural Networks

Sumohana S. Channappayya

Review

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

Review: Supervised Learning

Input (**x**)	Label (y)
	Cat
	Bird
	Dog
	Fish
	?

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

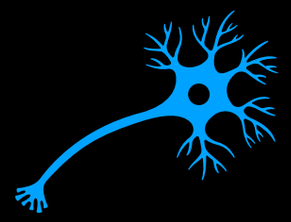
- How to learn parameters θ?
  - Quantify error: $d(y, \hat{y})$
    - Find **error** between estimate and ground truth over **training data** points: $R(\theta) = \sum_{i=1}^{n} d(y_i, \hat{y}_i)$
    - Find parameters that **minimise error**
- How **good** is our **model**?
  - Evaluate model's performance 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)

Motivation for Neural Networks

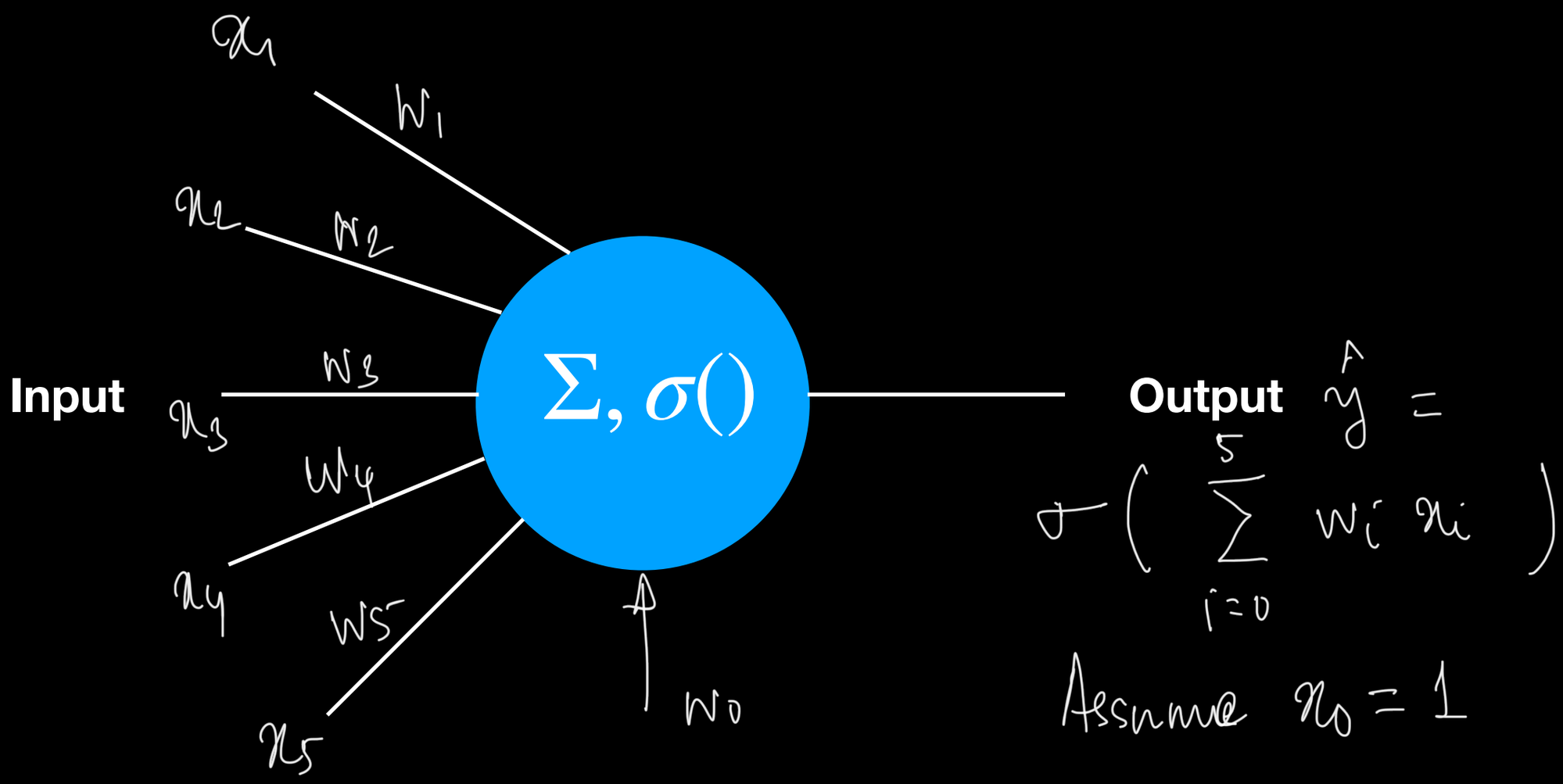
- Modelling the **neuron**



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

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

A Neuron


Input
$\Sigma, \sigma()$
Output $\hat{y} = \sigma\left(\sum_{i=0}^{5} w_i x_i\right)$
Assume $x_0 = 1$

Key function: $\sigma()$

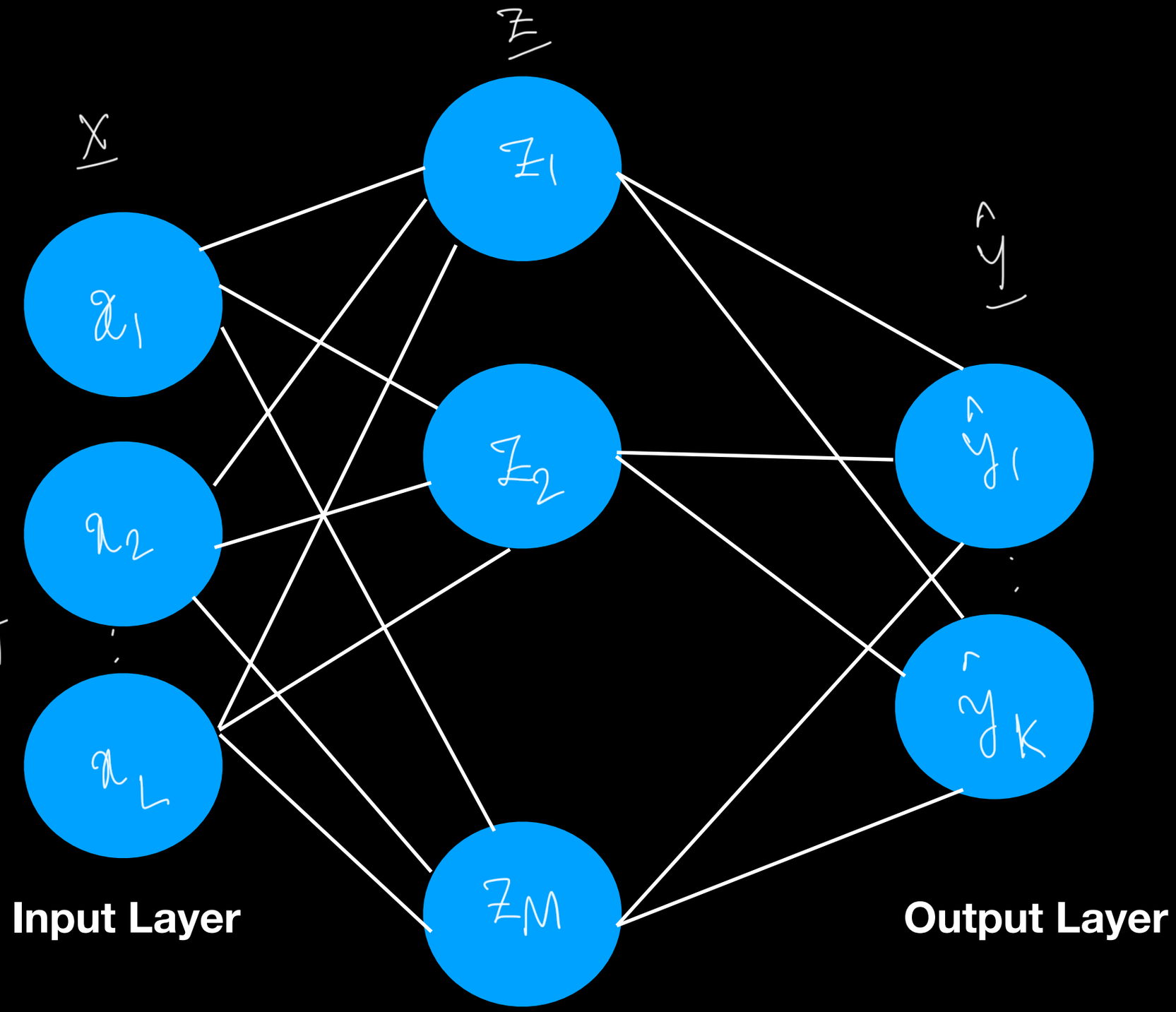
- Recall motivation: **nonlinear** input output relationship
- Properties of $\sigma()$
  - Nonlinear
  - Continuous
  - Differentiable?
  - Examples: tanh, sigmoid, ReLU etc.

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

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

$$r(x) = \max\{0, x\}$$

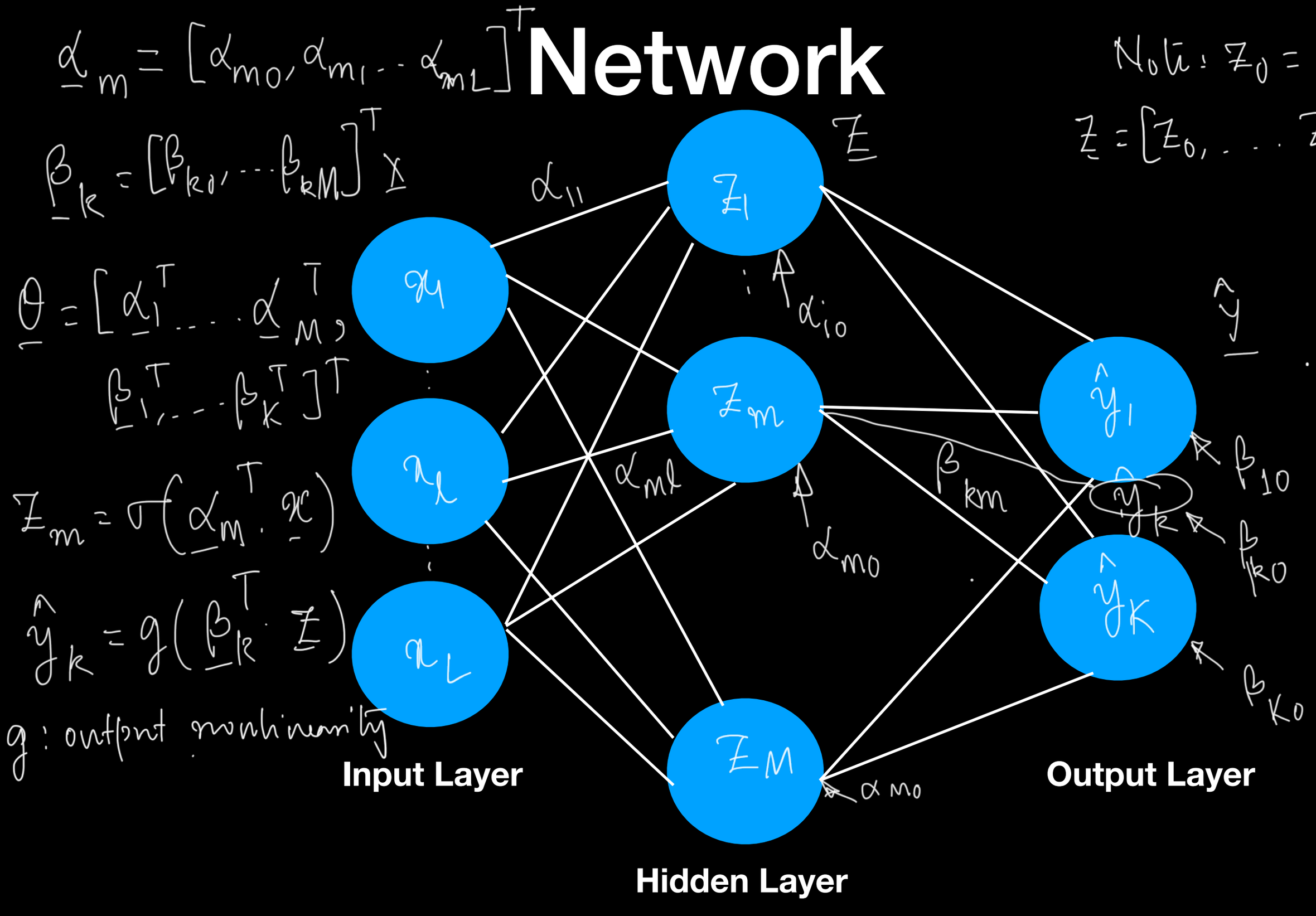
A Neural Network



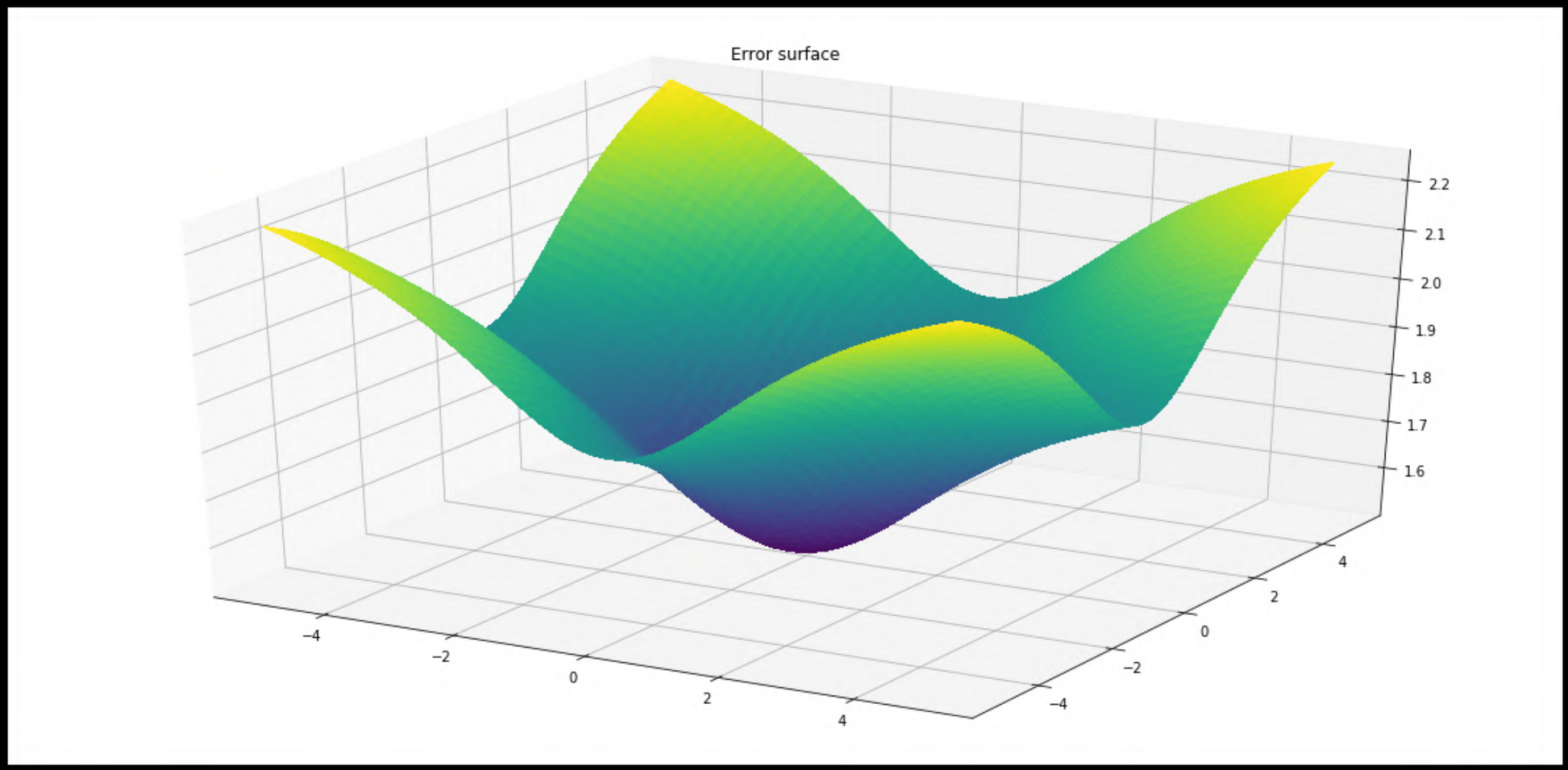
Note: $a_0 = 1$

$\underline{x} = [x_0, \cdots x_L]^T$

Feedforward Neural Network



Error Surface



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

S.S.E: $d(y, \hat{y}) = \sum_{i=1}^{K} (y_i - \hat{y}_i)^2$; $d_2(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i$ — C.E

- Derivation:

$$R(\underline{\theta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_{ki} - \hat{y}_{ki})^2$$

Let's find $\frac{\partial R(\underline{\theta})}{\partial \beta_{km}}$. Observation: β_{km} affects only $\hat{y}_k$. $\Rightarrow$ we can ignore the contribution of other outputs to $R(\underline{\theta})$.

$$\therefore \frac{\partial}{\partial \beta_{km}} \sum_{i=1}^{n} \sum_{j=1}^{K} (y_{ji} - \hat{y}_{ji})^2 = \frac{\partial}{\partial \beta_{km}} \sum_{i=1}^{n} (y_{ki} - \hat{y}_{ki})^2$$

$$= \frac{\partial}{\partial \beta_{km}} \sum_{i=1}^{n} (y_{ki} - g(\underline{\beta}_k^T \cdot \underline{z}_i))^2$$

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

- Derivation: $\Rightarrow \frac{\partial}{\partial \beta_{km}} R(\underline{\theta}) = -\sum_{i=1}^{n} 2(y_{ki} - \hat{y}_{ki}) \cdot g'(\underline{\beta}_k^T \cdot \underline{z}_i)\, z_{mi}$ — (1) ✓

$$\frac{\partial}{\partial \beta_{km}} [\underline{\beta}_k^T \cdot \underline{z}_i] = \frac{\partial}{\partial \beta_{km}} [\beta_{k0} \cdot z_{0i} + \beta_{k1} z_{1i} + \cdots \underbrace{\beta_{km} \cdot z_{mi}}_{= z_{mi}} + \cdots \beta_{kM} \cdot z_{Mi}]$$

$$\therefore \boxed{\frac{\partial R(\underline{\theta})}{\partial \beta_{km}} = \sum_{i=1}^{n} \delta_{ki} z_{mi}} \quad \text{where } \delta_{ki} = -2(y_{ki} - \hat{y}_{ki}) \cdot g'(\underline{\beta}_k^T \cdot \underline{z}_i)$$

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

- Derivation:

$$\frac{\partial R(\underline{\theta})}{\partial \alpha_{ml}} = \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{\partial}{\partial \alpha_{ml}} \left(y_{ki} - \hat{y}_{ki}\right)^2$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{\partial}{\partial \alpha_{ml}} \left(y_{ki} - g(\beta_k^T \underline{z}_i)\right)^2 = \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{\partial}{\partial \alpha_{ml}} \left(y_{ki} - g\left(\beta_k^T \cdot \begin{bmatrix} z_{0i} \\ z_{1i} \\ \vdots \\ z_{mi} \end{bmatrix}\right)\right)^2$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \frac{\partial}{\partial \alpha_{ml}} \left(y_{ki} - g(\ldots + \beta_{km} \cdot \sigma(\underline{\alpha}_m^T \cdot \underline{x}_i))\right)^2$$

$$\frac{\partial R(\underline{\theta})}{\partial \alpha_{ml}} = \sum_{k=1}^{K} \sum_{i=1}^{n} -2 \cdot \underbrace{(y_{ki} - g(\quad))}_{} \cdot g'(\quad) \beta_{km} \cdot \sigma'(\underline{\alpha}_m^T \cdot \underline{x}_i) \cdot \underline{x}_{li} \; g_{ki} \quad -(2)$$

$$\underline{\alpha}_m^T \cdot \underline{x}_i = \alpha_{m0} \cdot x_{0i} + \alpha_{m1} x_{1i} + \cdots + \alpha_{ml} \cdot x_{li} + \cdots + \alpha_{mL} \cdot x_{Li}$$

Back propagation

$$\frac{\partial R(\underline{\theta})}{\partial \alpha_{ml}} = \sum_{i=1}^{n} s_{mi}\, x_{li}$$

where $s_{mi} = \sigma'(\underline{\alpha}_m^T \cdot \underline{x}_i) \cdot \sum_{k=1}^{K} \beta_{km} \cdot \delta_{ki}$.

back propagation of error.

Back propagation

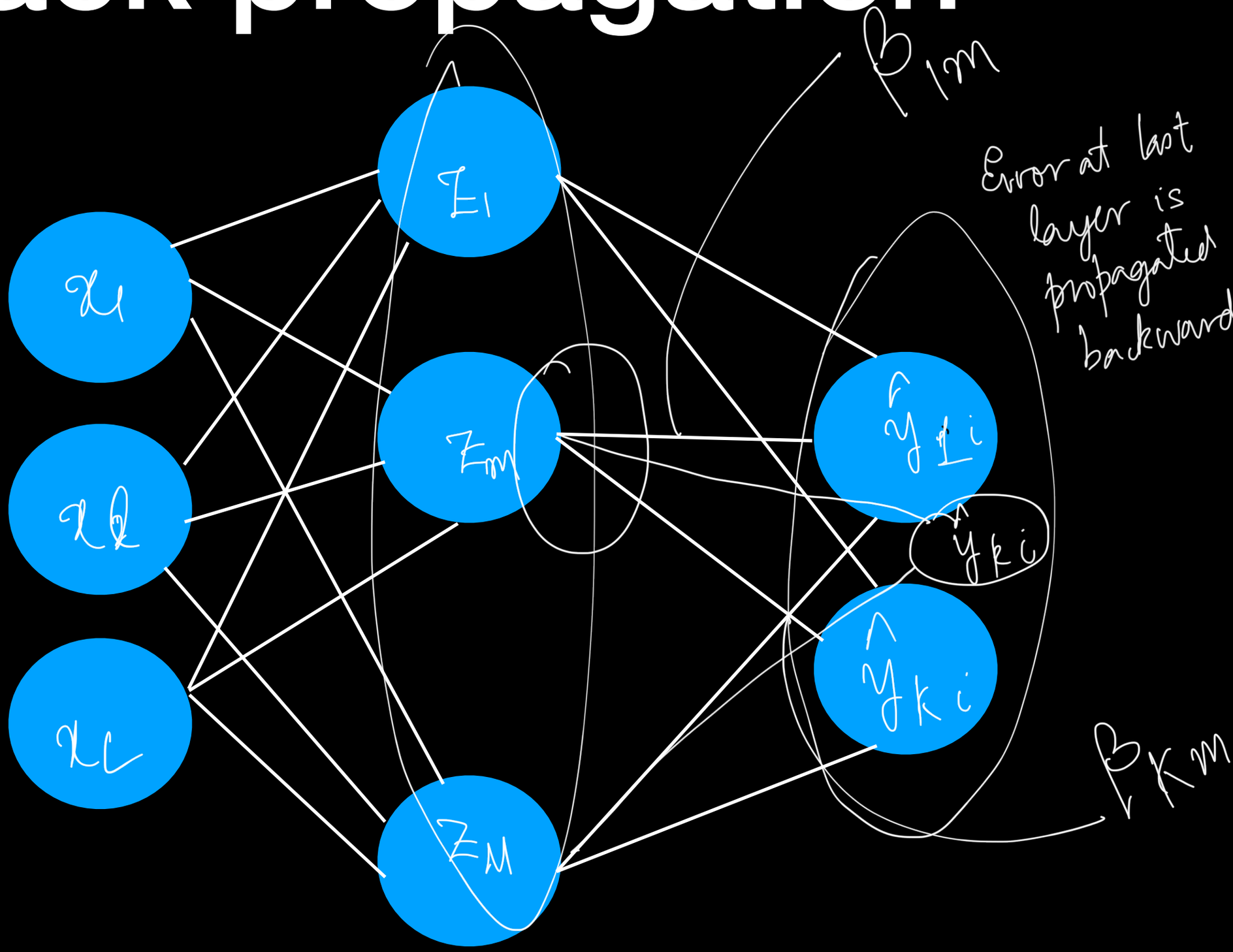
$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \cdot \frac{\partial R(\underline{\theta})}{\partial \beta_{km}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \frac{\partial R(\underline{\theta})}{\partial \alpha_{ml}}$$

r: iteration count

γ_r: learning rate

Back propagation



Challenges

- Initialization
- Overfitting
- Input range
- Network architecture
- Multiple minima