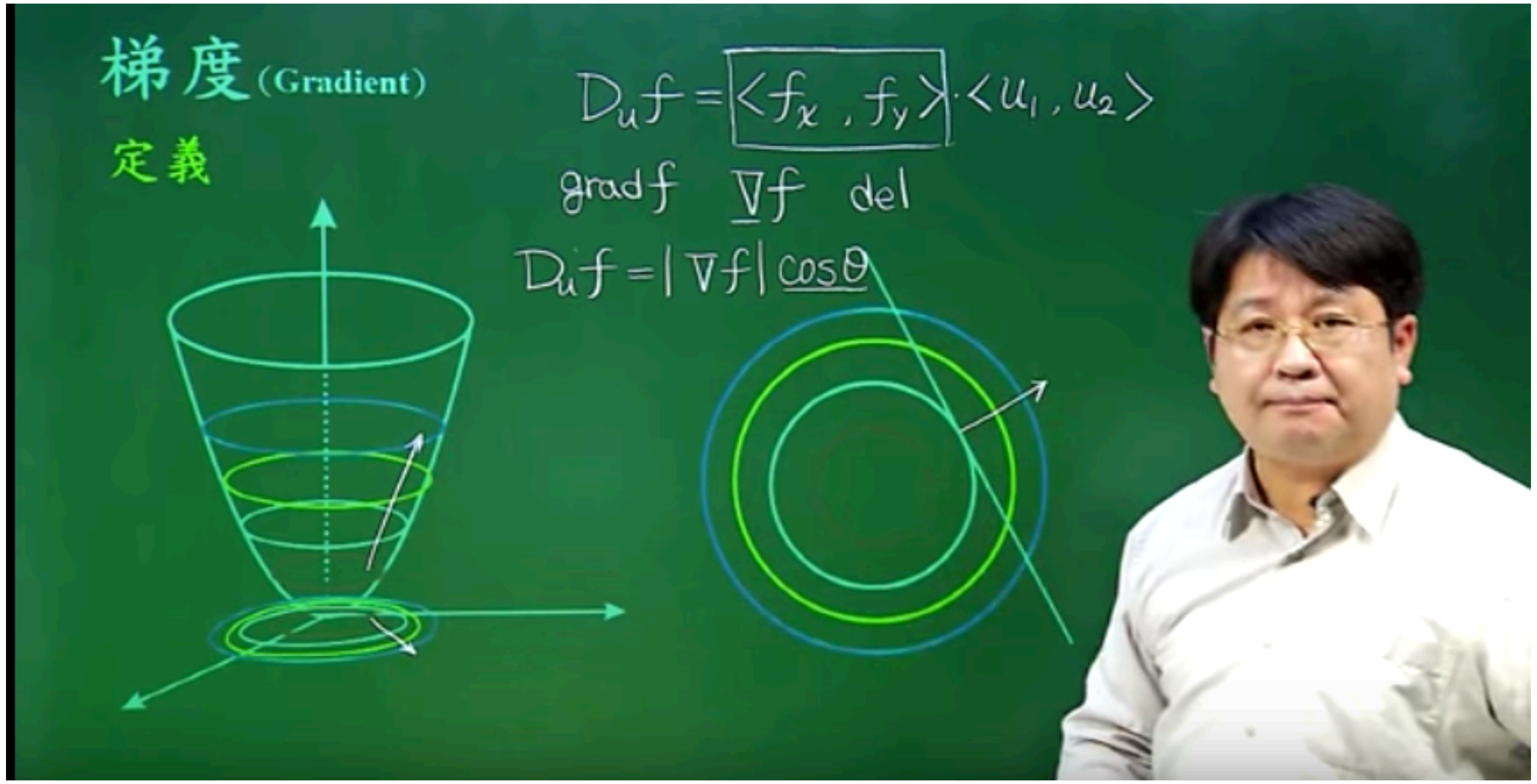
Deep into Neural Network from Gradient Descent

Gradient 梯度



https://www.youtube.com/watch?v=npkI19rcpdY

在向量微積分中，純量場的梯度是一個向量場。 純量場中某一點的梯度指向在這點純量場增長最快的方向（當然要比較的話必須固定方向的長度），梯度的絕對值是長度為1的方向中函數最大的增加率，也就是說 ∇ f.

梯度- 維基百科，自由的百科全書 - Wikipedia
https://zh.wikipedia.org/zh-tw/梯度

f = x^2 + y^2

grad = <2x, 2y>

- **分別對*x, y* 偏微**
- ***example:***

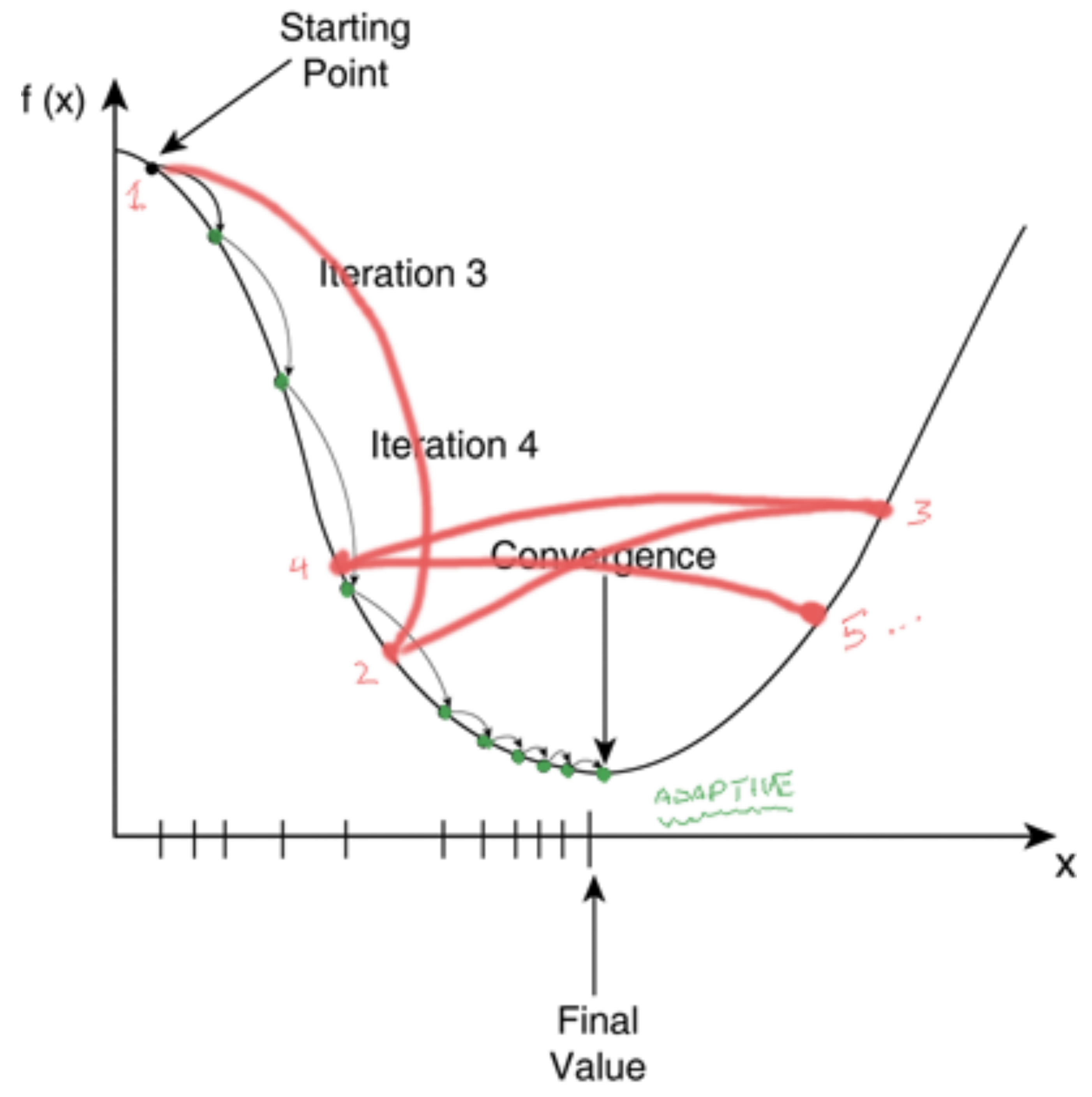
when in point <1, 2>

then grad = vector <2, 4>

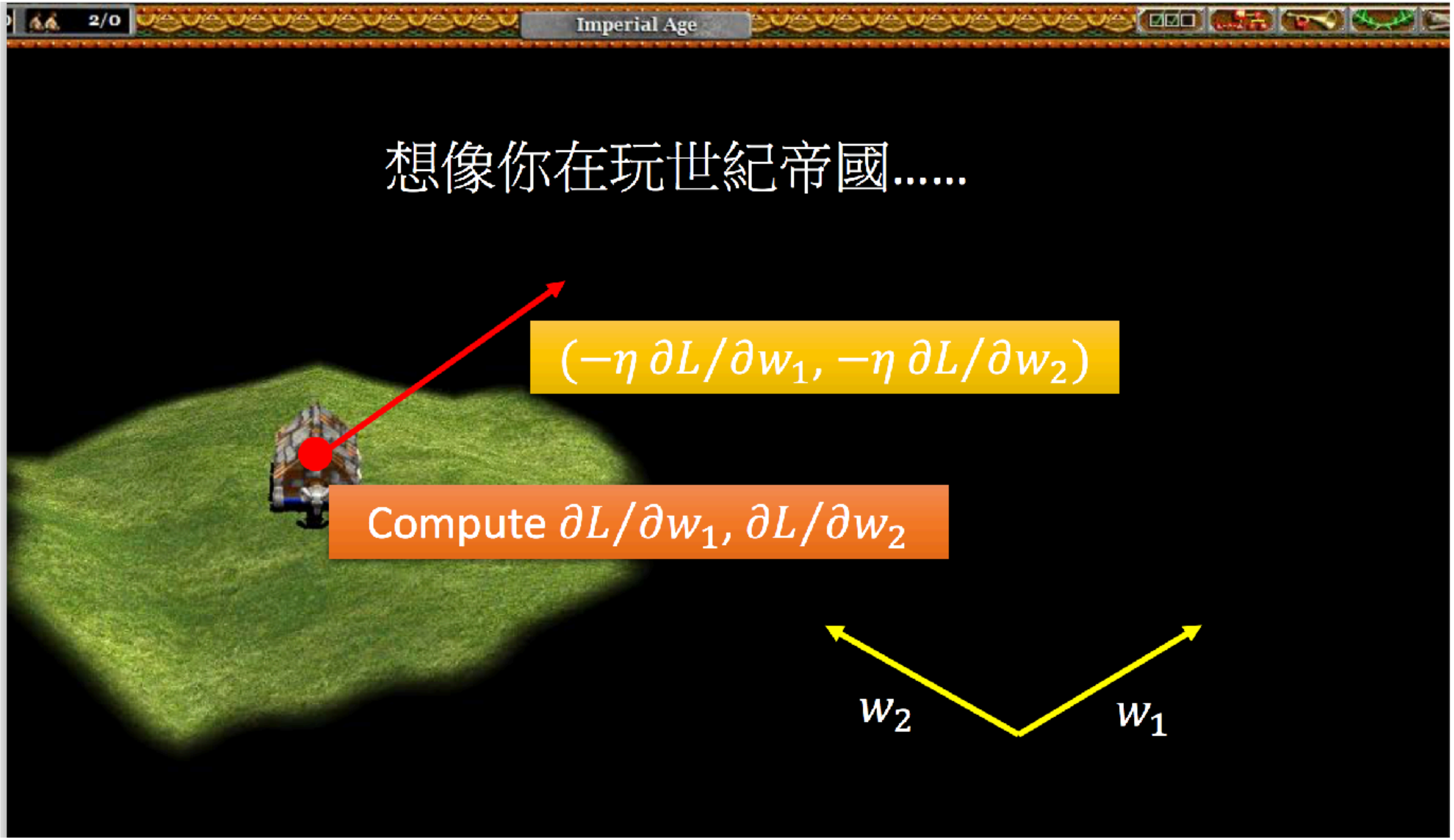
就是在點(1, 2)時, 在<2, 4>的方向坡度最陡 (最難爬的方向)

Gradient Descent 梯度下降法

- 所以在X, Y點時, gradient的相反方向就是最好下降的方向
- 但跳太大步可能會跳到坑洞的另一方
- 跳太小步又很孬種(太慢了！)
- 跳的大小程度就是learning rate
- 每次都跳一樣大、越跳越小等等就是策略
- AdamOptimizer就是一種策略
- 每次前進前要用多少資料描繪當時可能的地形(1 vs mini-batch)



往當時下坡度最大的方向跳的程度


2/0
Imperial Age
想像你在玩世紀帝國……
$(-\eta\, \partial L/\partial w_1, -\eta\, \partial L/\partial w_2)$
Compute $\partial L/\partial w_1, \partial L/\partial w_2$
w_2
w_1

對方程式做Gradient Descent

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)})^2$$

w1*x(i)1 + w2*x(i)2 + ..wj*x(i)j + …

$$\frac{\partial}{\partial \theta_j}\mathrm{MSE}(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)})\, x_j^{(i)}$$

$$\nabla_\theta \mathrm{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0}\mathrm{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1}\mathrm{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n}\mathrm{MSE}(\theta) \end{pmatrix} = \frac{2}{m}\mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

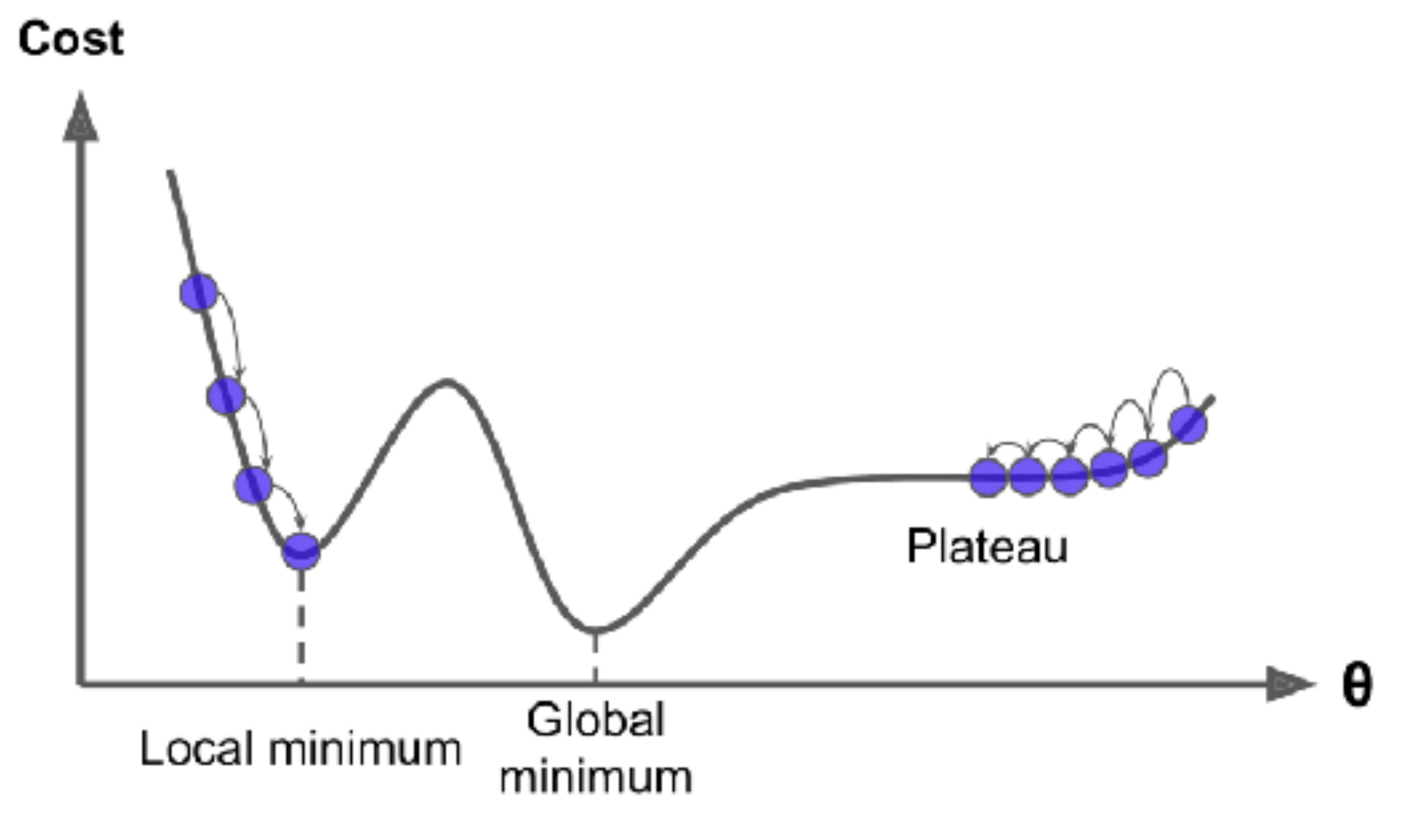


Figure 4-5. Gradient Descent pitfalls

$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$$

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)})^2$$

$$X = \begin{bmatrix} [1, & 2, & 1] \\ [2, & 3, & 1] \end{bmatrix} \quad \theta = \begin{bmatrix} [0.5], \\ [0.75] \\ [0.25] \end{bmatrix} \quad y = \begin{bmatrix} [4] \\ [6] \end{bmatrix} \quad X \cdot \theta = \begin{bmatrix} [2.25] \\ [3.5] \end{bmatrix}$$

$(2, 3) \qquad (3, 1) \qquad (2, 1)$

$$X \cdot \theta - y = \text{error} = \begin{bmatrix} [-1.75] \\ [-2.5] \end{bmatrix}$$

$$\nabla J = \left(\frac{2}{m} \sum_{i=1}^{m=2} \Big(\underbrace{\underbrace{x^{(i)} \cdot w}_{\text{y\_pred}} - y}_{\text{error (都一樣)}} \Big) \times x_1^{(i)} \quad \ldots \ldots \times x_2^{(i)} \quad \vdots \right) = \frac{2}{m} \begin{bmatrix} [-1.75 \times 1 + -2.5 \times 2] \\ [-1.75 \times 2 + -2.5 \times 3] \\ [-1.75 \times 1 + -2.5 \times 1] \end{bmatrix}$$

$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)})\, x_j^{(i)}$$

$$= \begin{bmatrix} [-6.75] \\ [-11] \\ [-4.25] \end{bmatrix}$$

$\Rightarrow$ if η (learning rate) $= 0.01$

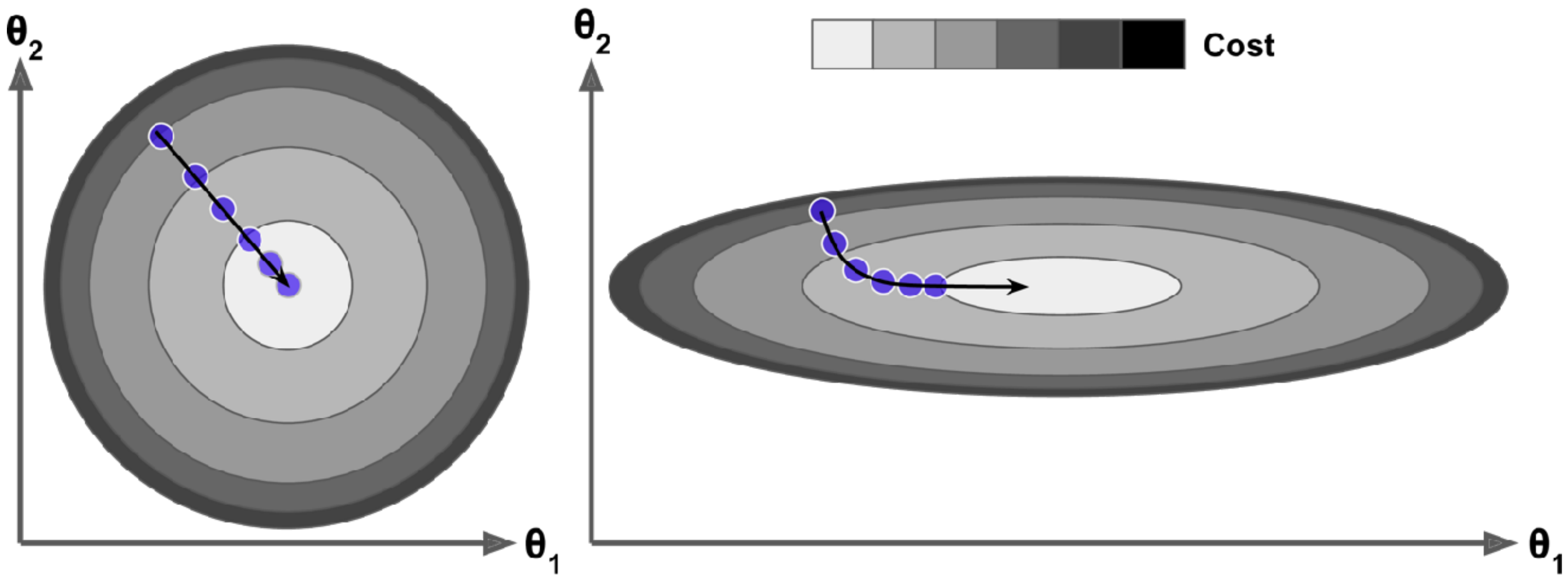
$$\theta \leftarrow \theta - 0.01 \times \nabla J$$

$$= \begin{bmatrix} [0.5] \\ [0.75] \\ [0.25] \end{bmatrix} - 0.01 \begin{bmatrix} [-6.75] \\ [-11] \\ [-4.25] \end{bmatrix} = \begin{bmatrix} [0.5675] \\ [0.86] \\ [0.2925] \end{bmatrix}$$

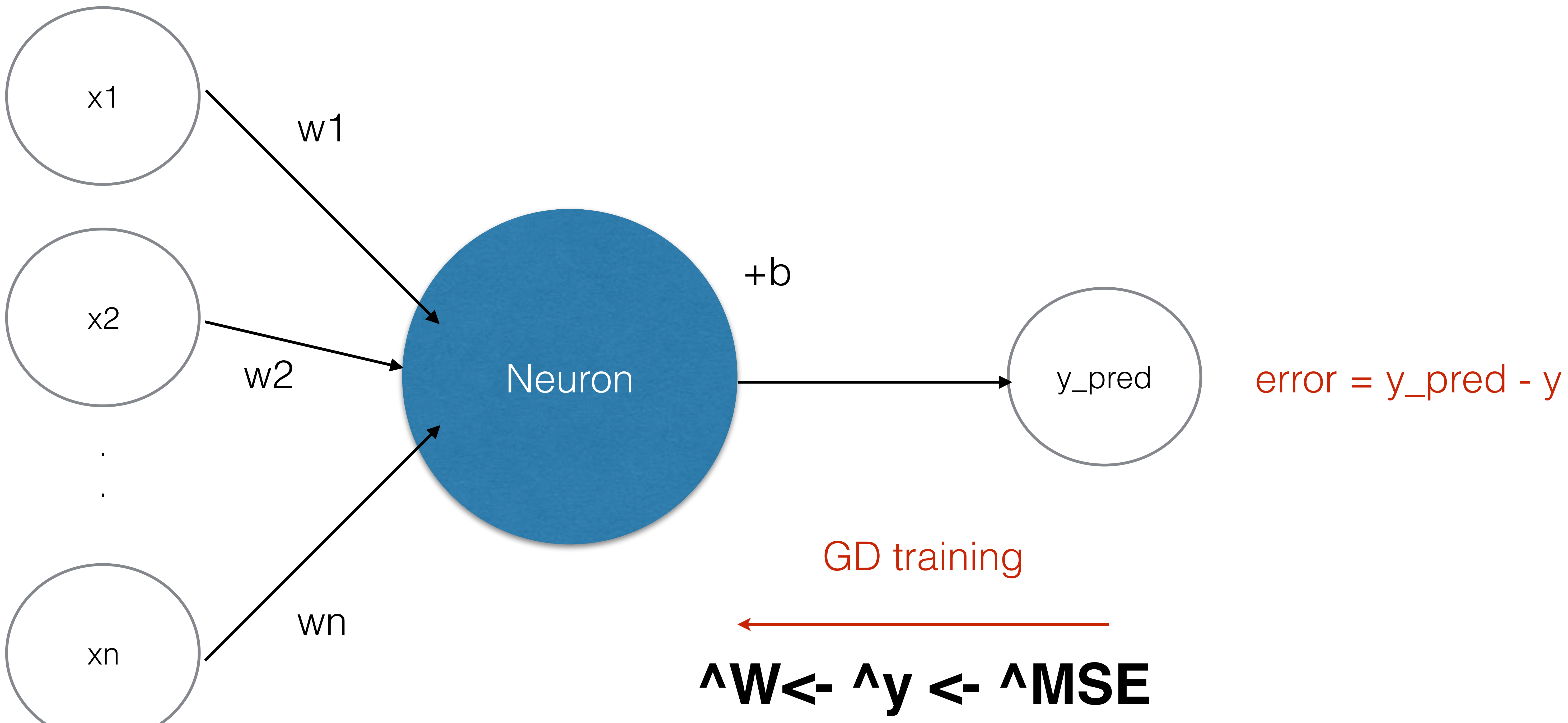
$$\theta^{(\text{next step})} = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$$

θ next step $\longrightarrow \cdots$

做Gradient Descent前, 要scaled features!

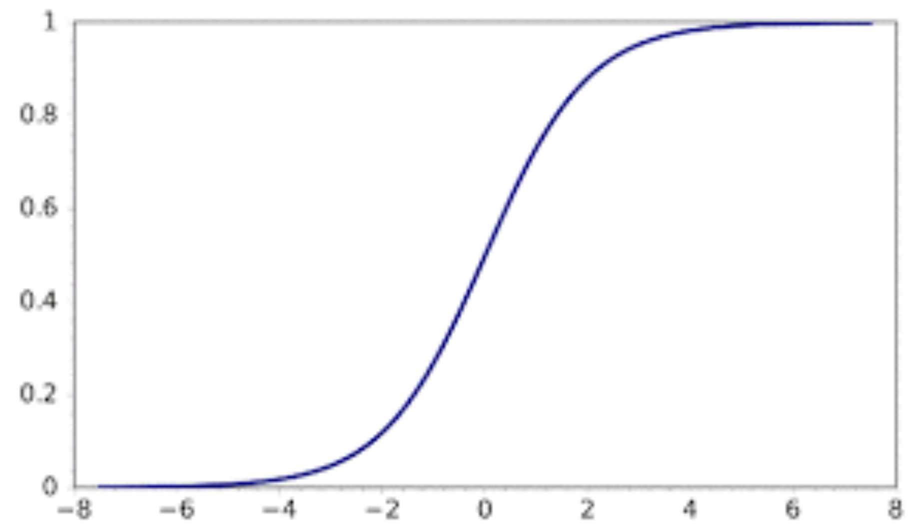


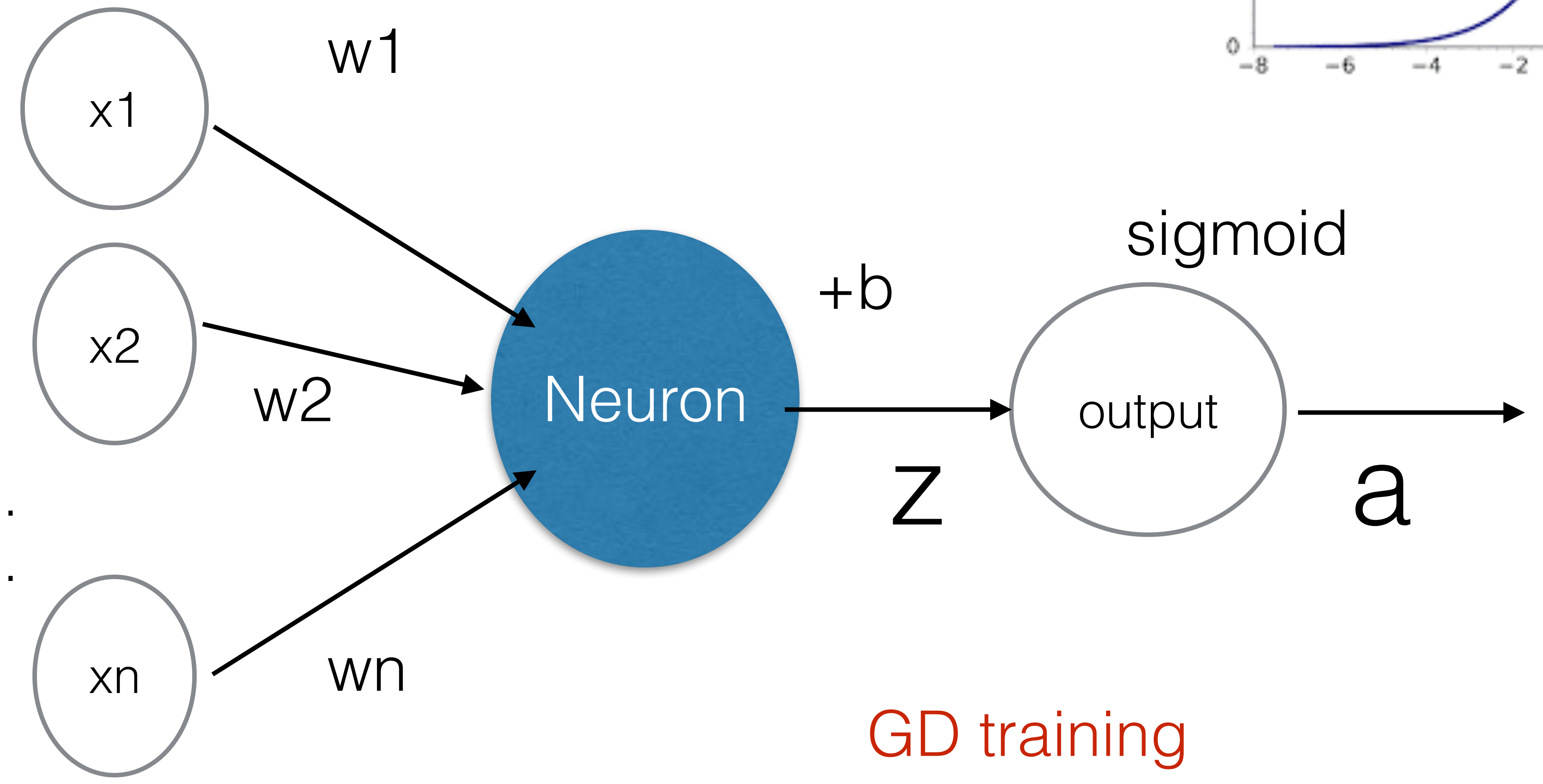
練習1: Linear Regression



最小化MSE訓練出W

Logistic Regression





if >0.5
then 1
else 0

$$H = -\sum_i (p_i . log_2(q_i))$$

-5*5 = -25
-4*6 = -24
-3*7 = -21

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\hat{p}^{(i)}) + (1 - y^{(i)})log(1 - \hat{p}^{(i)})]$$

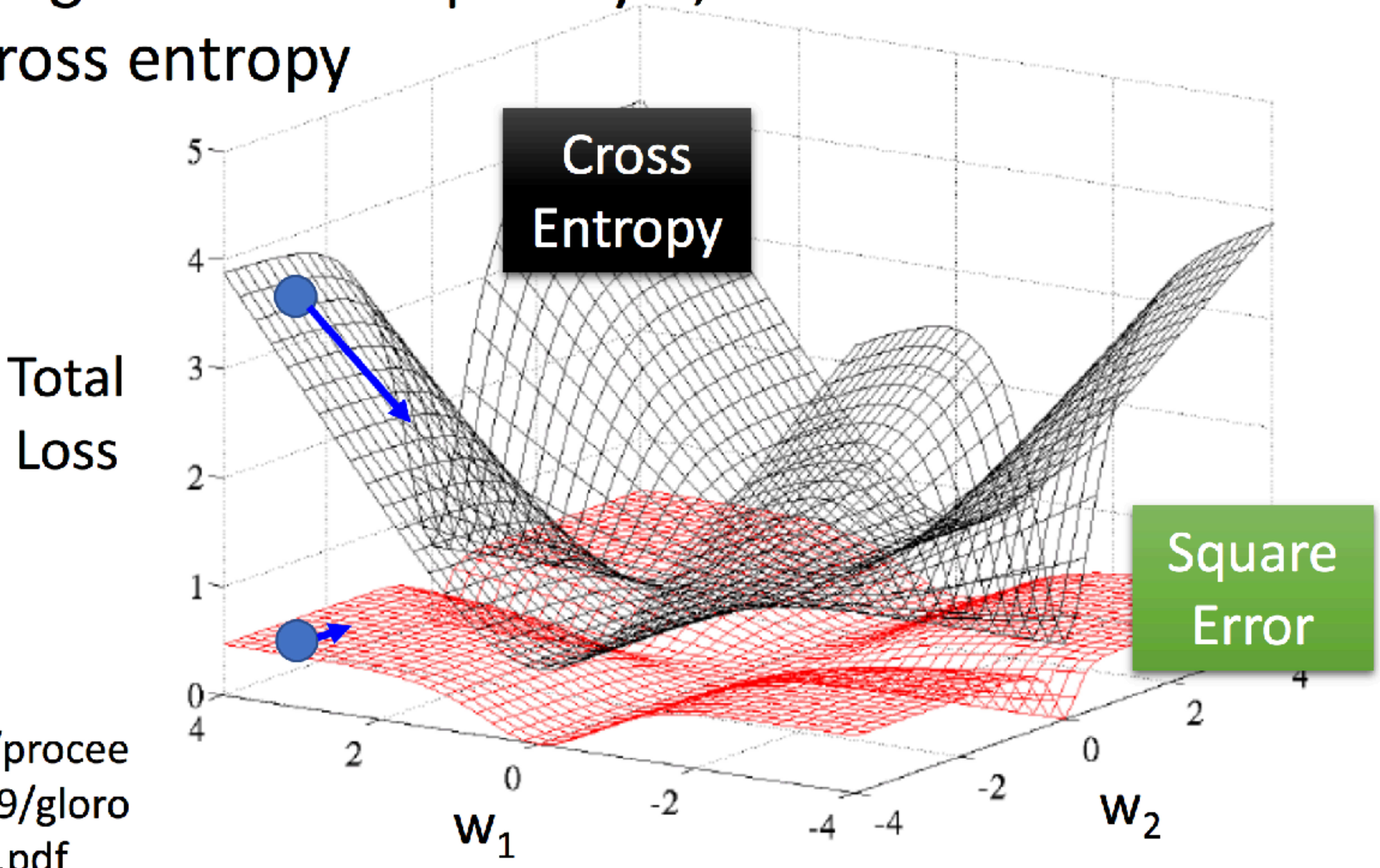
GD training

^W<-^z <- ^a(sigmoid 斜率) <- ^loss

最小化Cross Entropy訓練出W

Choosing Proper Loss

When using softmax output layer, choose cross entropy

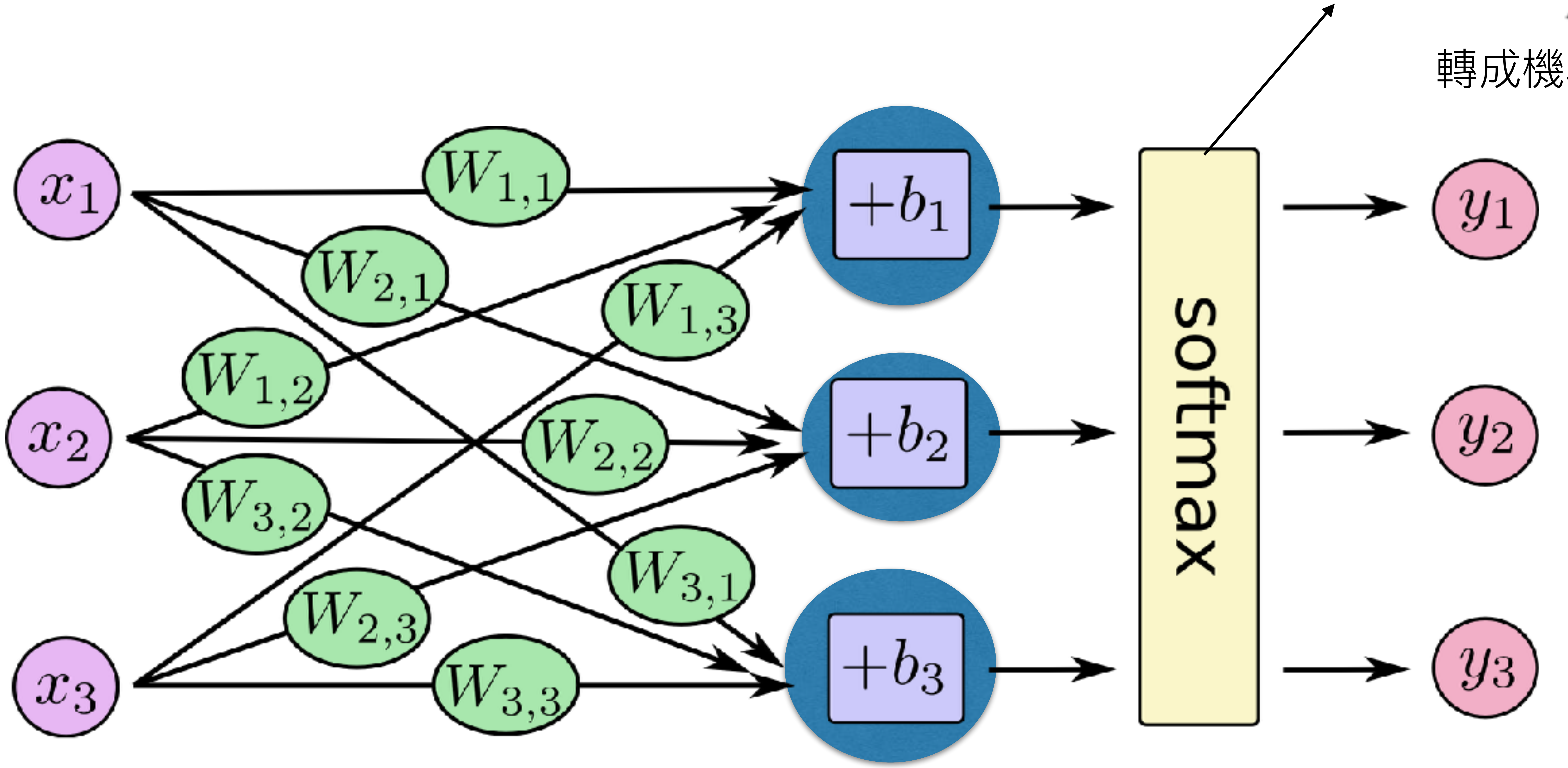


http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

練習2: Softmax classification

$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

轉成機率



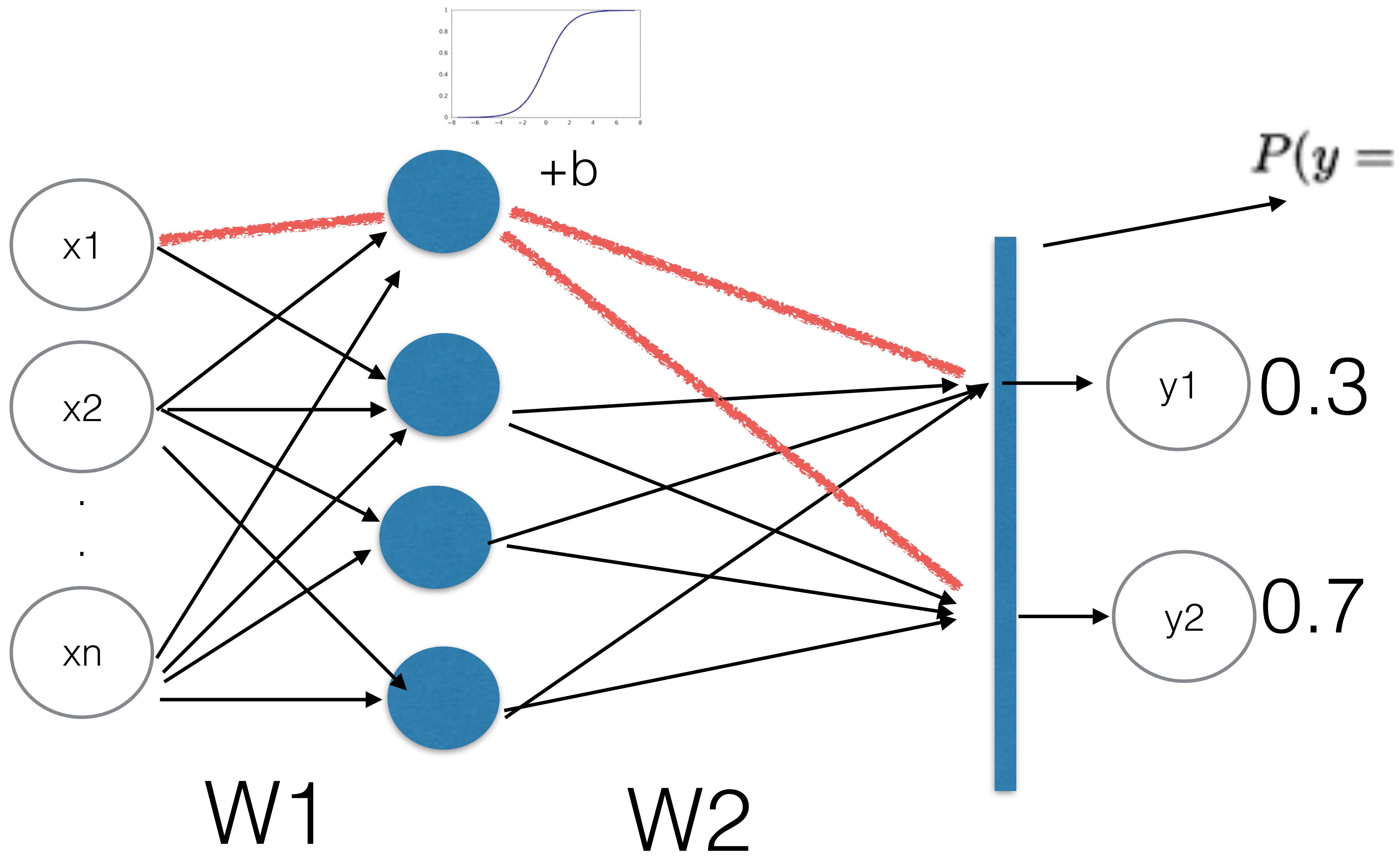
$$loss = -\sum_i \sum_c y_{c_i} \cdot log(y\_predicted_{c_i})$$

最小化Cross Entropy訓練出W

Neural Network

GD training

^z <- ^a <- ^w21 + ^w22 <- ^C



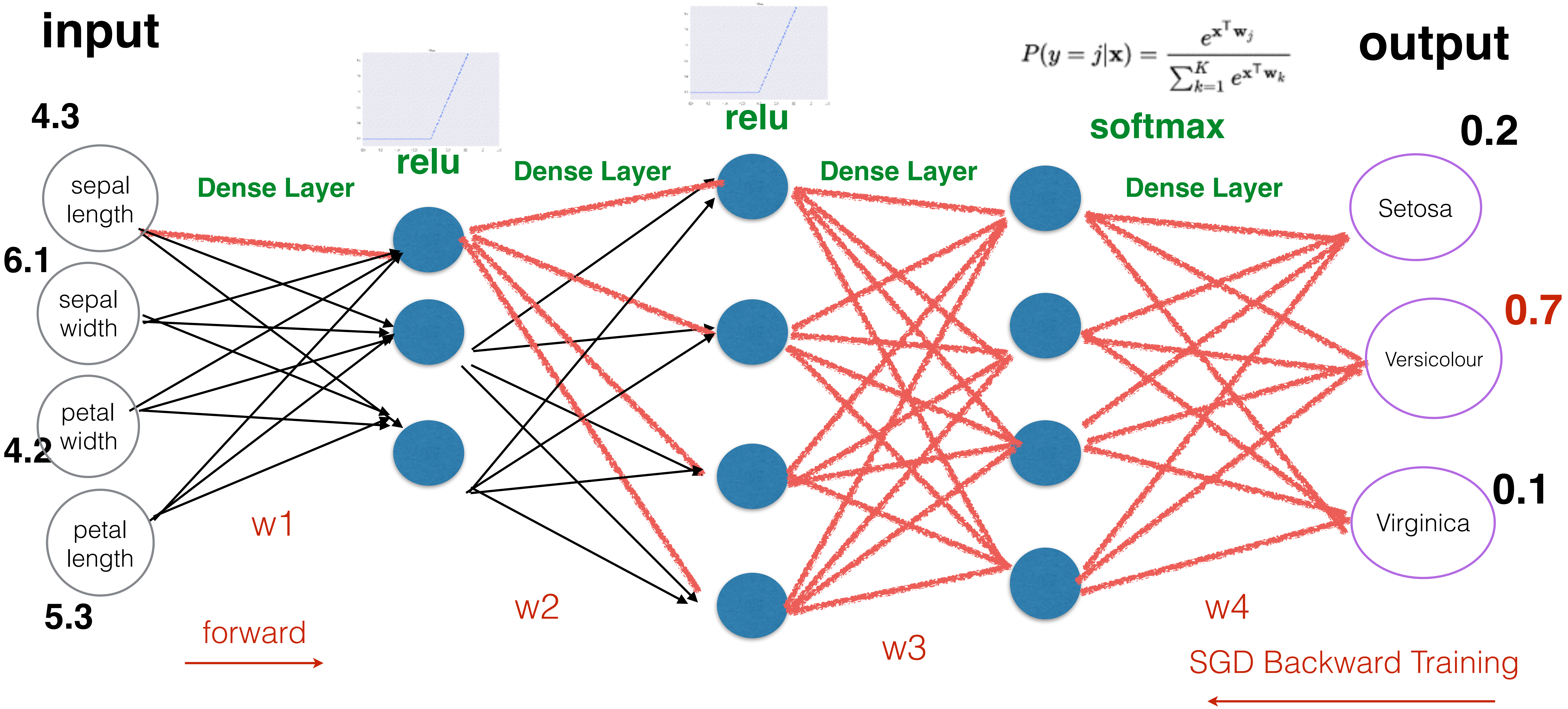
$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

Softmax function
轉成機率

最小化Cross Entropy訓練出W2

W1的改變會擴散到W1 -> Backpropagation

練習3: Build your neural network

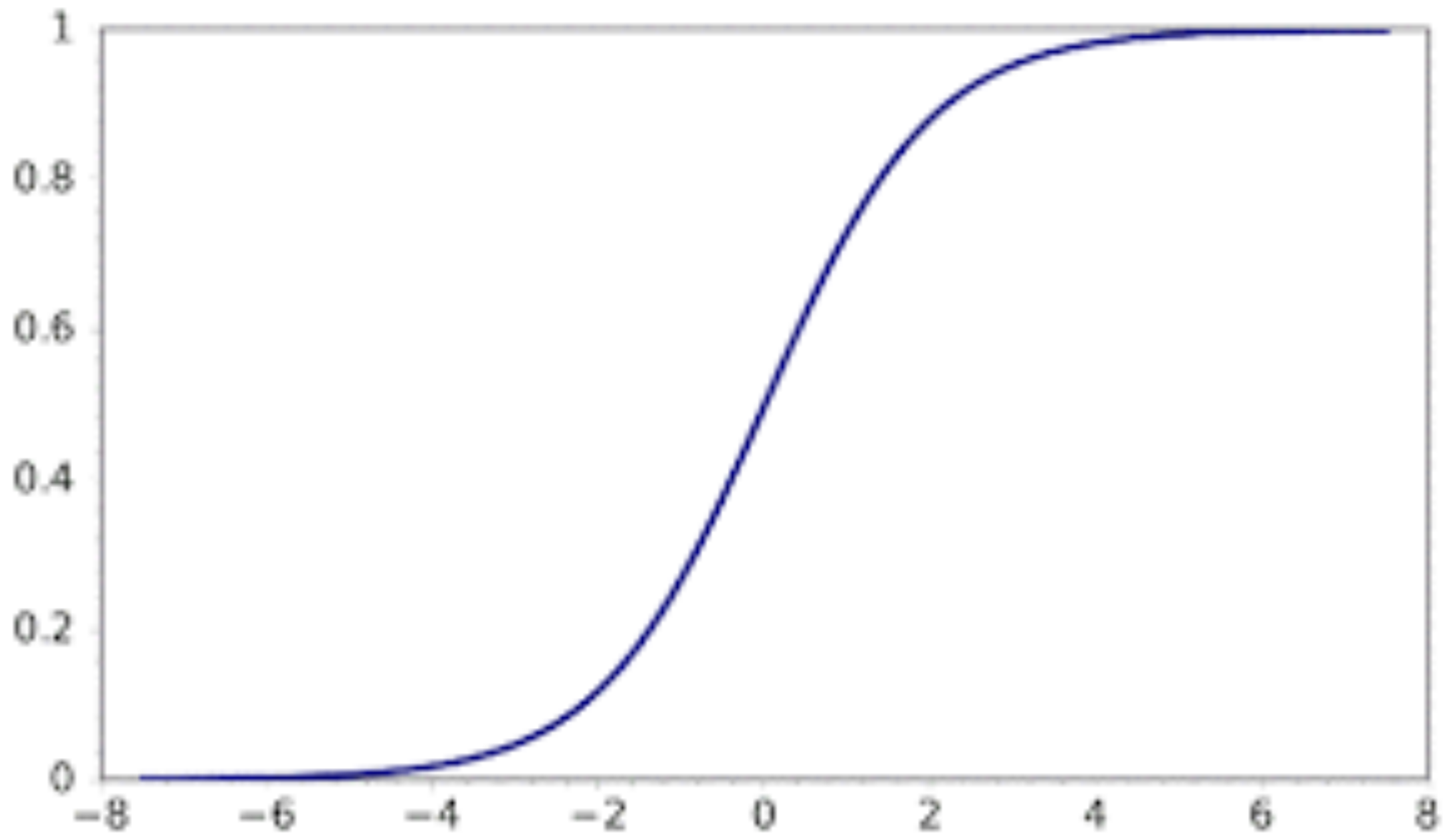


loss function to minimize:
cross entropy

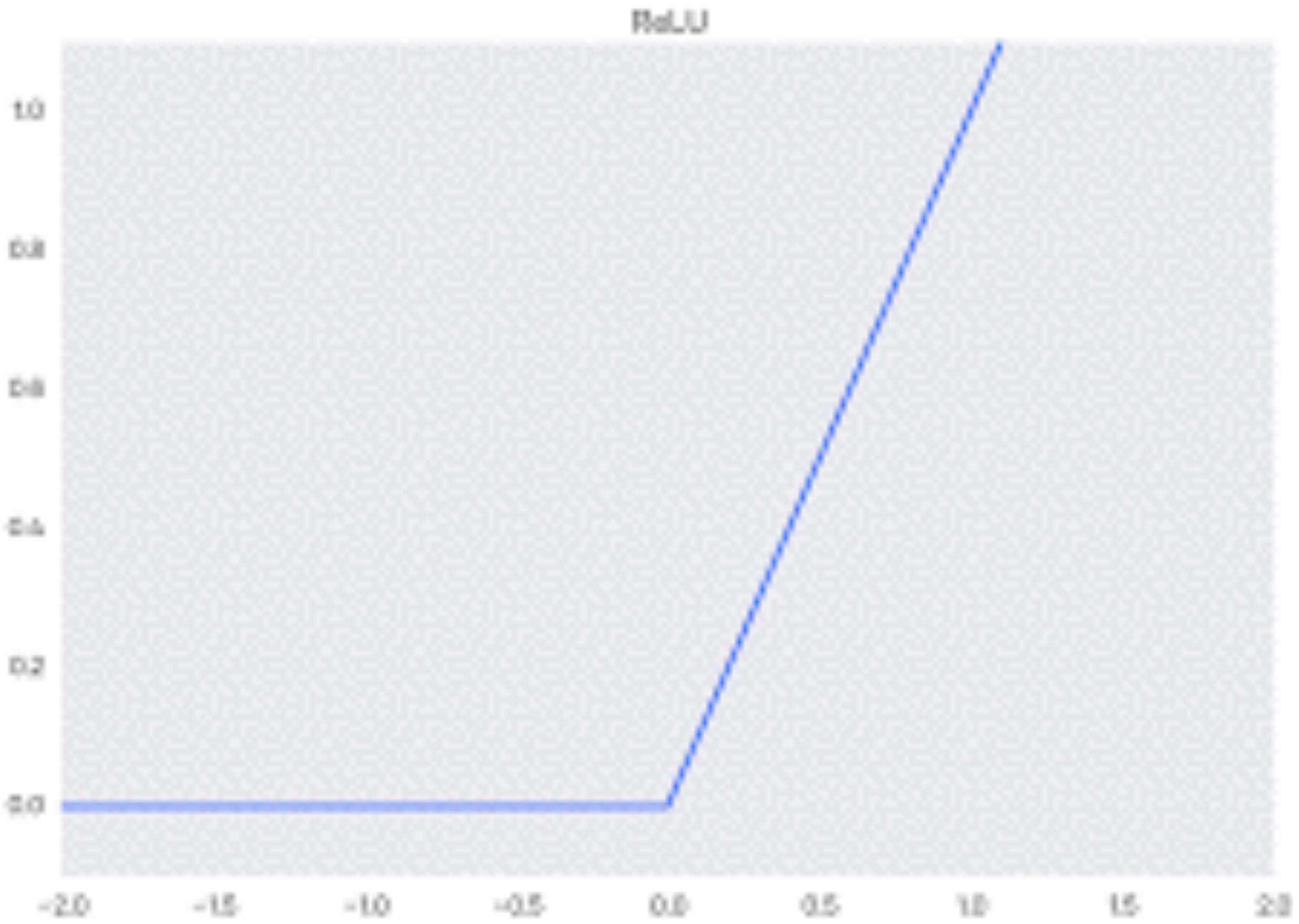
深層網路的BP會層層往後更新

Question

Why from Sigmoid to Relu?

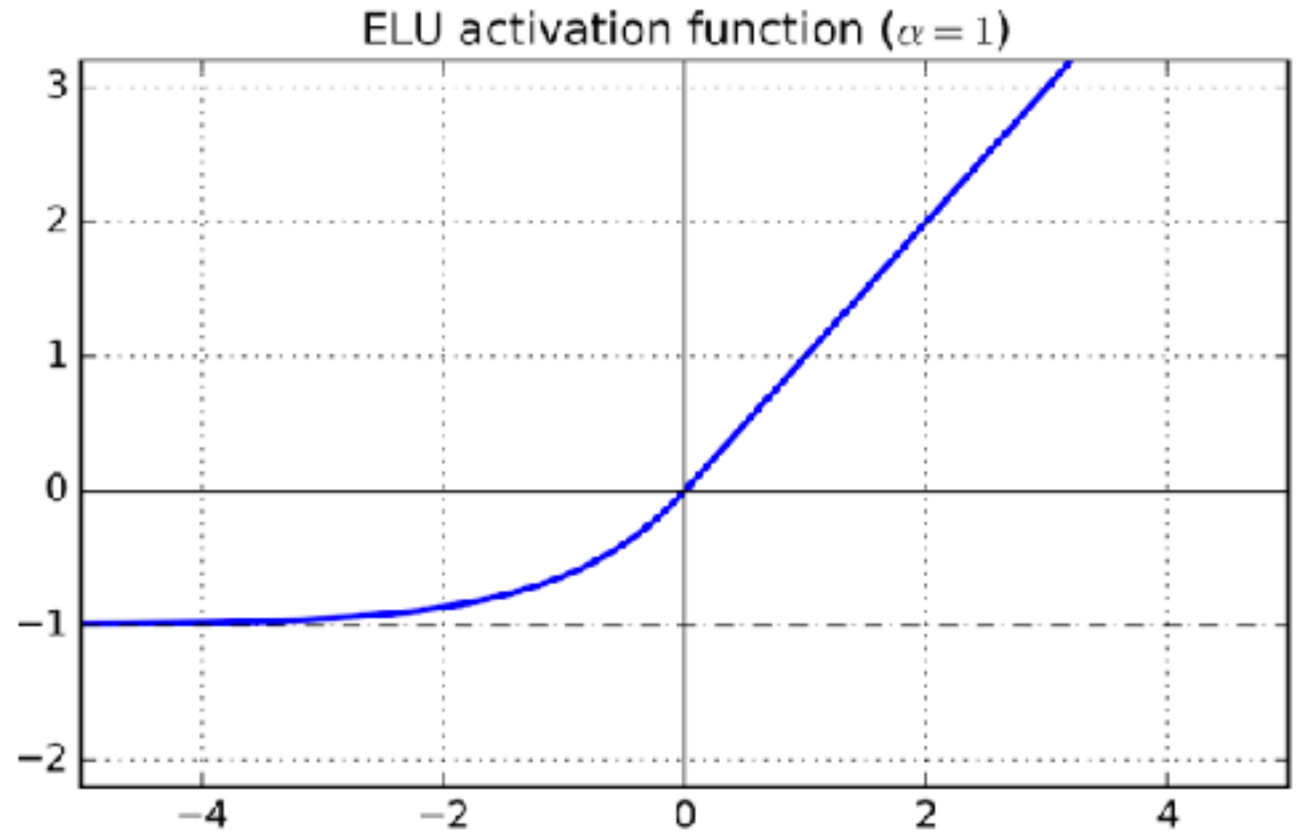


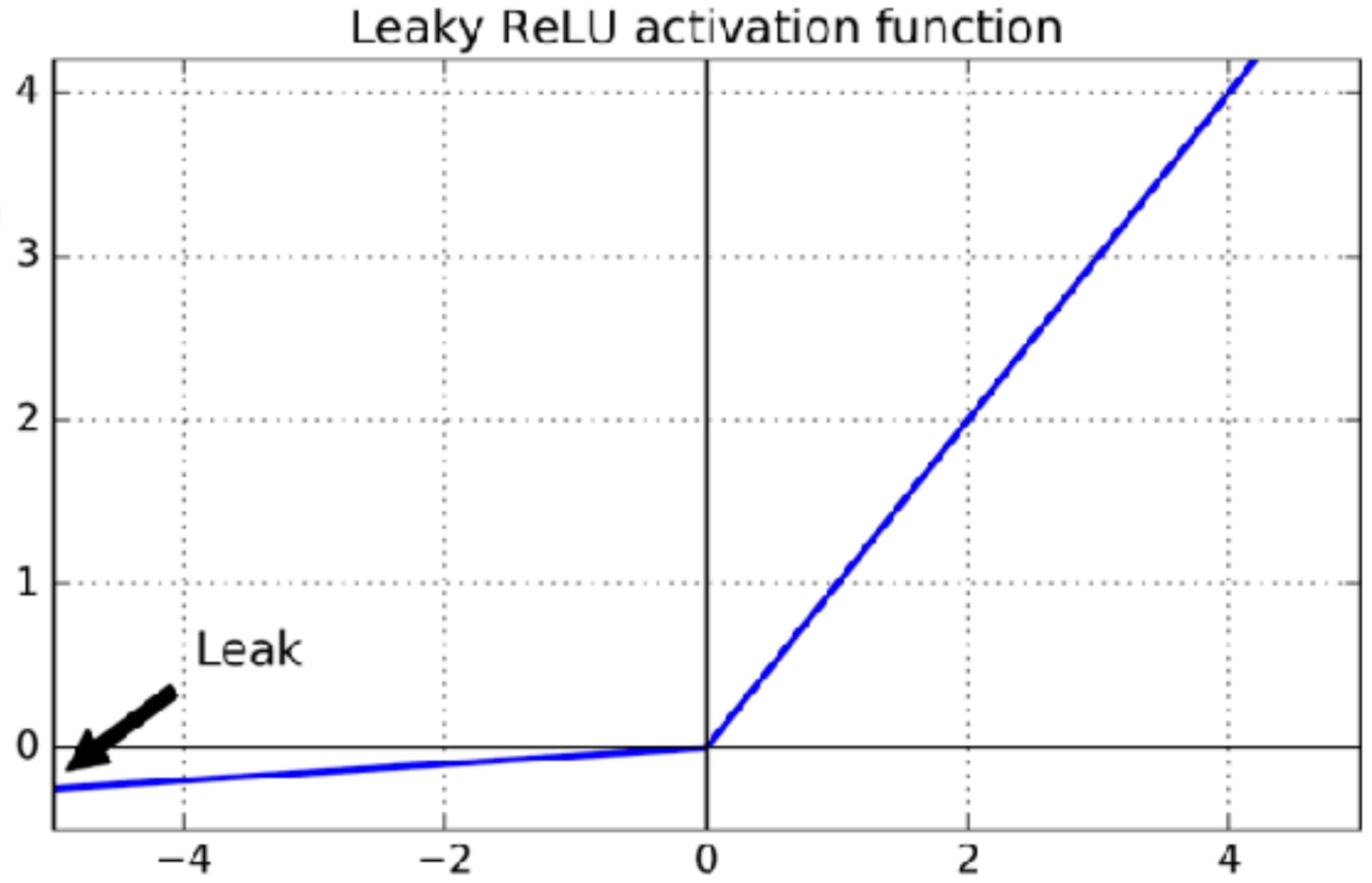
sigmoid



relu

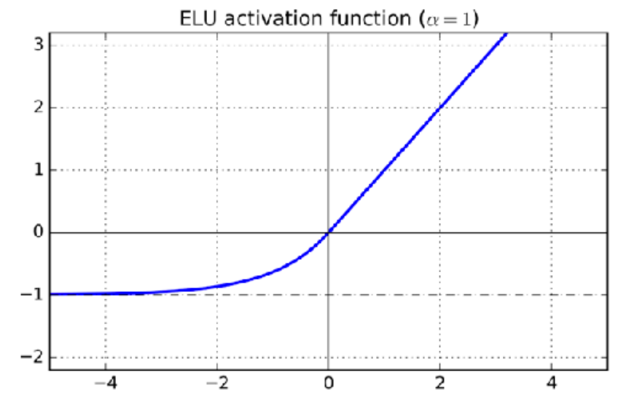
$$\mathrm{ELU}_{\alpha}(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & if z \geq 0 \end{cases}$$





Ways to improve DNN

$$\mathrm{ELU}_\alpha(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$



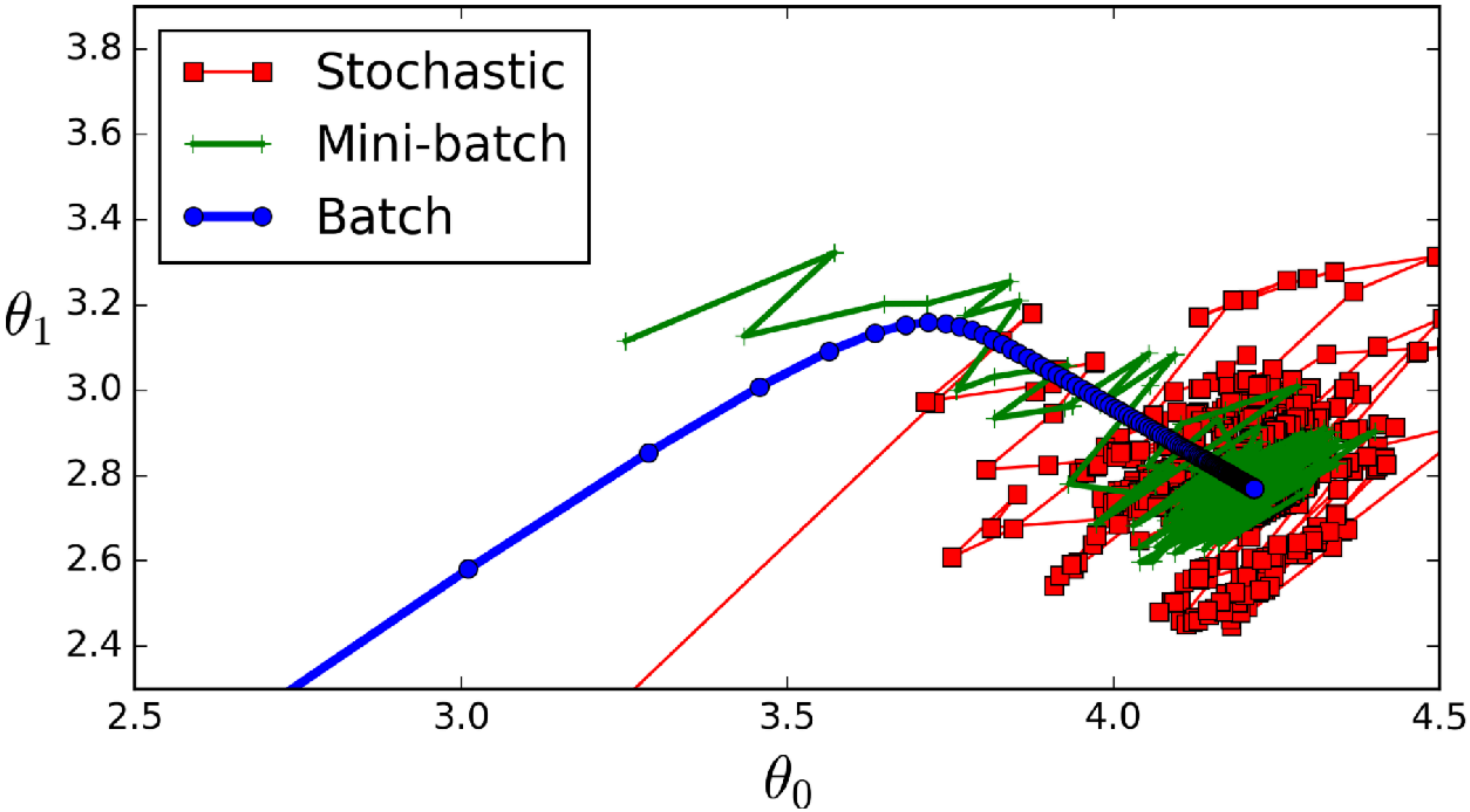
- Weight Initialization - He initialization
- Activation function - ELU
- Normalization - Batch Normalization
- Regularization - Dropout
- Optimizer - Adam
- Check 一天搞懂深度學習

Activation function	Uniform distribution [-r, r]	Normal distribution
Logistic	$r = \sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = \sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$
Hyperbolic tangent	$r = 4\sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = 4\sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$
ReLU (and its variants)	$r = \sqrt{2}\sqrt{\frac{6}{n_{\text{inputs}} + n_{\text{outputs}}}}$	$\sigma = \sqrt{2}\sqrt{\frac{2}{n_{\text{inputs}} + n_{\text{outputs}}}}$

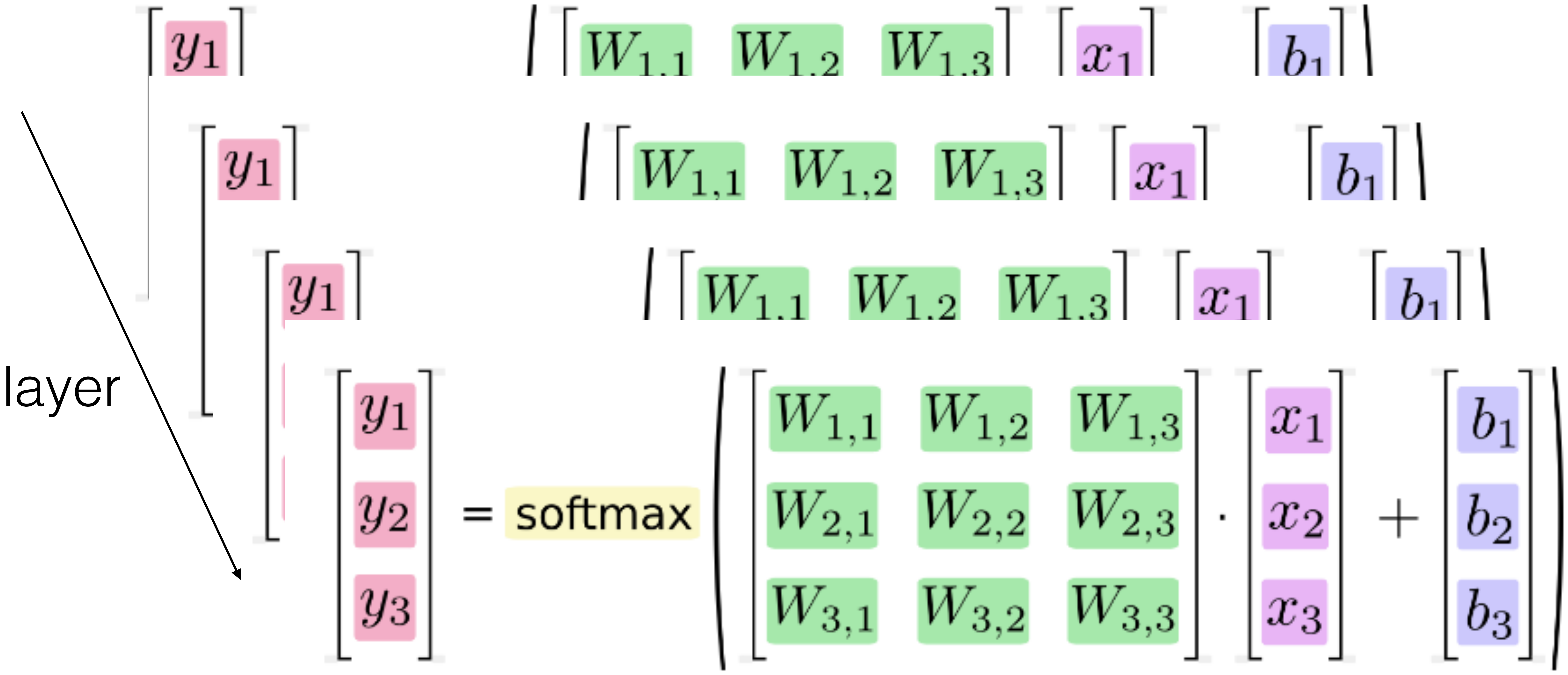
Equation 11-3. Batch Normalization algorithm

1. $\mu_B = \frac{1}{m_B} \sum_{i=1}^{m_B} \mathbf{x}^{(i)}$
2. ${\sigma_B}^2 = \frac{1}{m_B} \sum_{i=1}^{m_B} \left(\mathbf{x}^{(i)} - \mu_B\right)^2$
3. $\hat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \mu_B}{\sqrt{{\sigma_B}^2 + \epsilon}}$
4. $\mathbf{z}^{(i)} = \gamma \hat{\mathbf{x}}^{(i)} + \beta$

Why using mini-batch?



Too many matrix multiplication



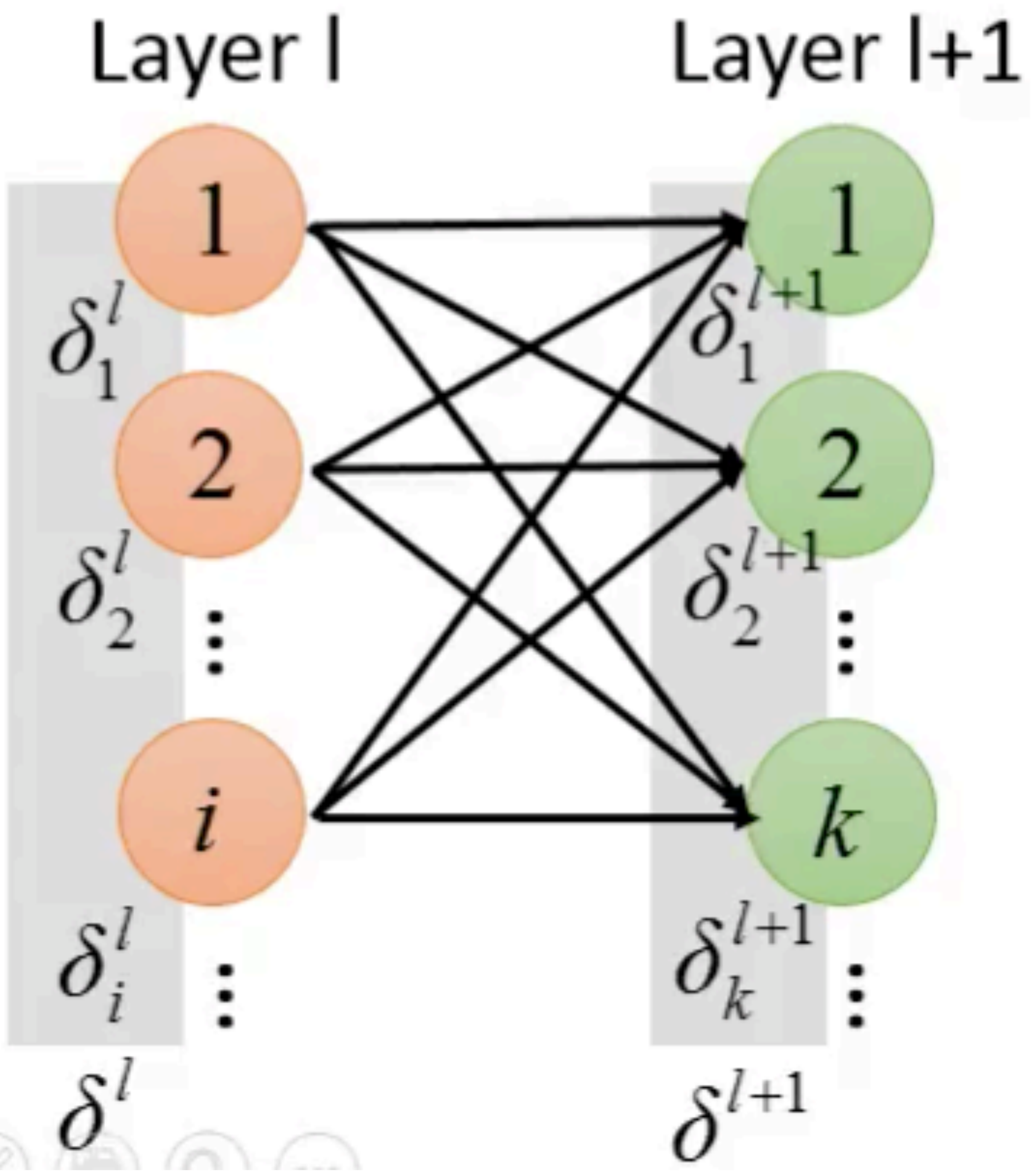
What is your question?

Others

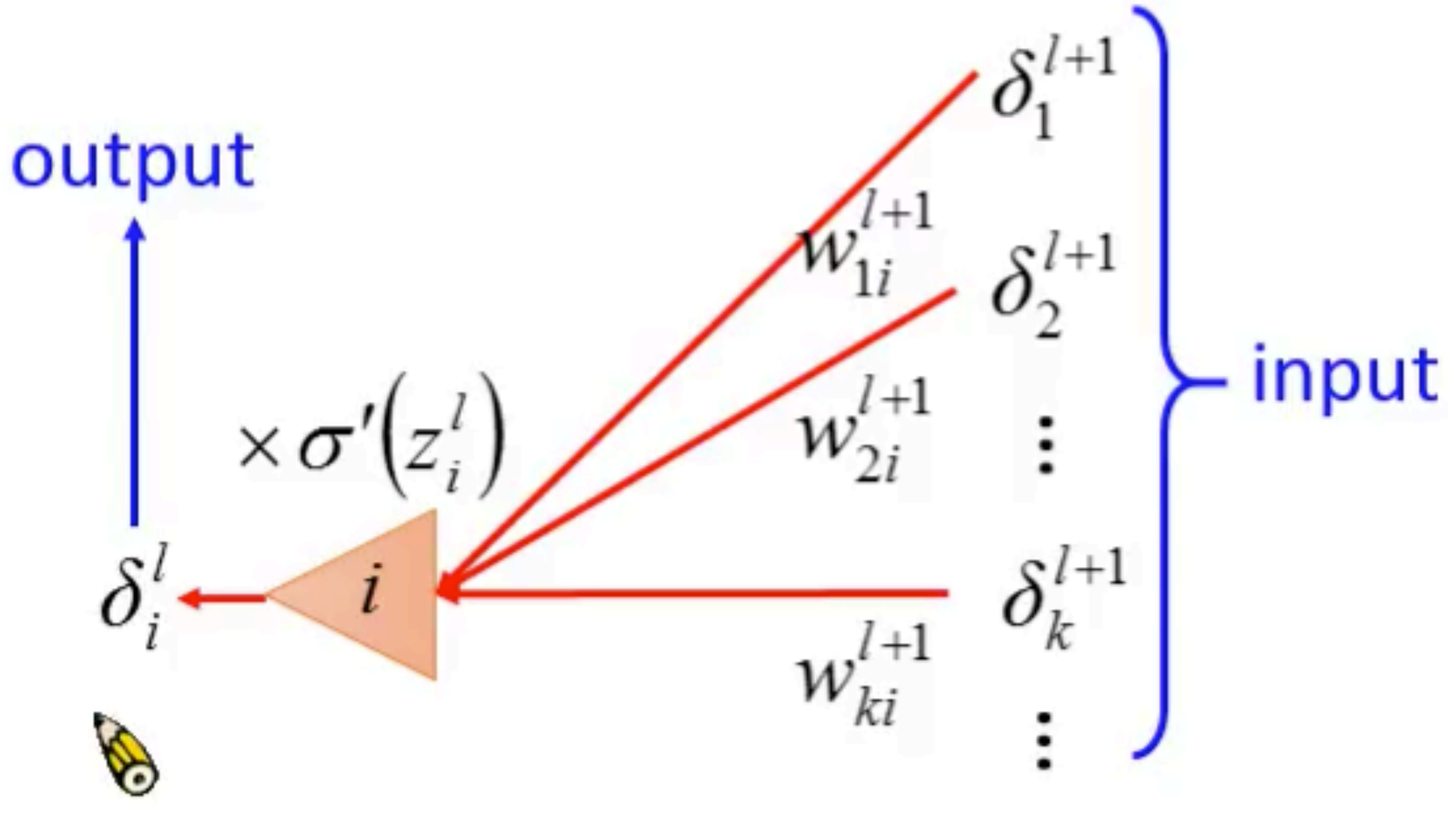
Backpropagation

$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l \qquad \delta_i^l = \sigma'(z_i^l) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$



new type of neuron

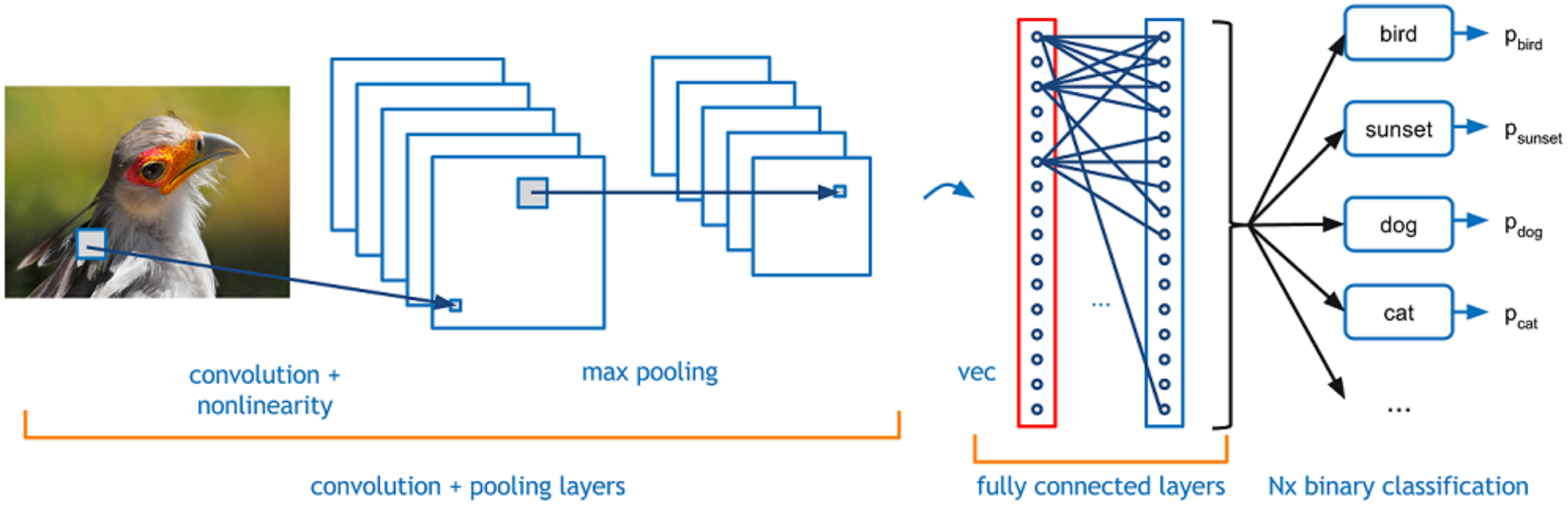


http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html

Homework

- Try classify mnist with Keras
- Check the tensorflow tutorials - https://www.tensorflow.org/tutorials/mnist/beginners/
- Check the tensorflow template (in templates folder)
- Check CNN concept - https://www.youtube.com/watch?v=JiN9p5vWHDY

Next time!



Convolution Neural Network