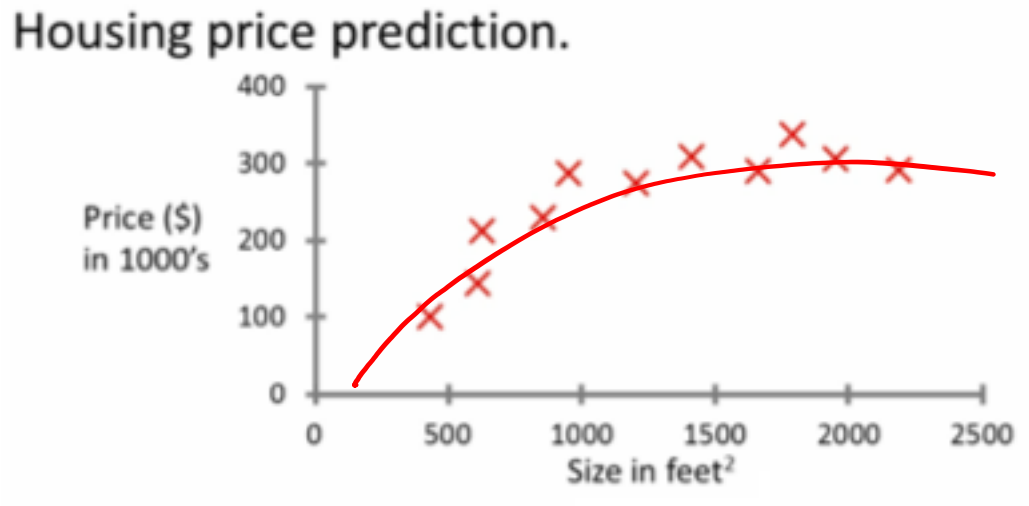
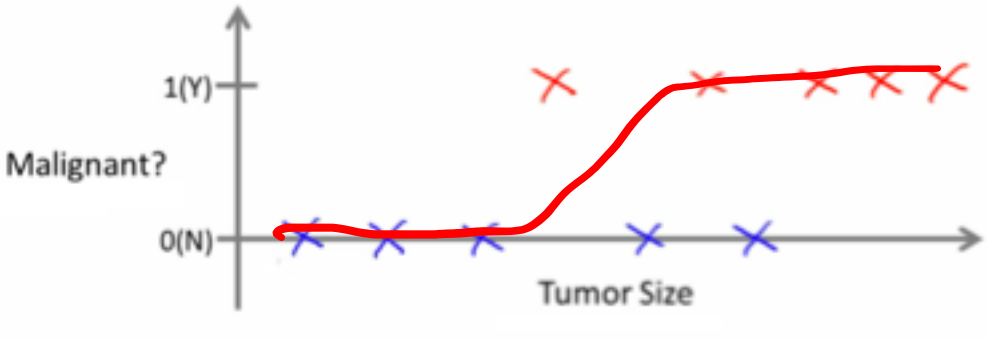
深度学习

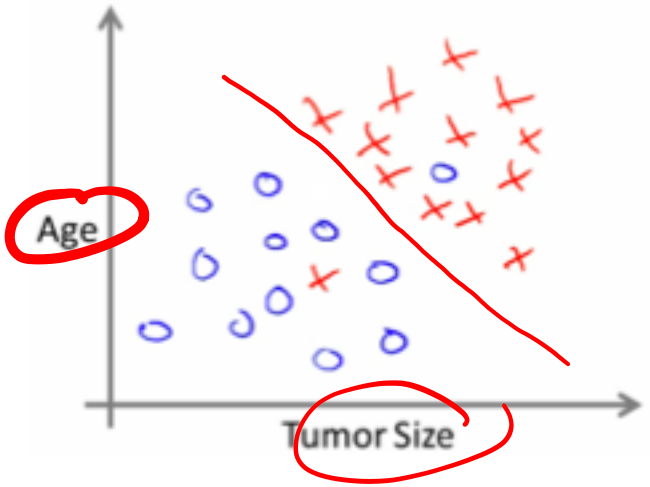
第三讲

王胤

Recap: Regression vs. Classification



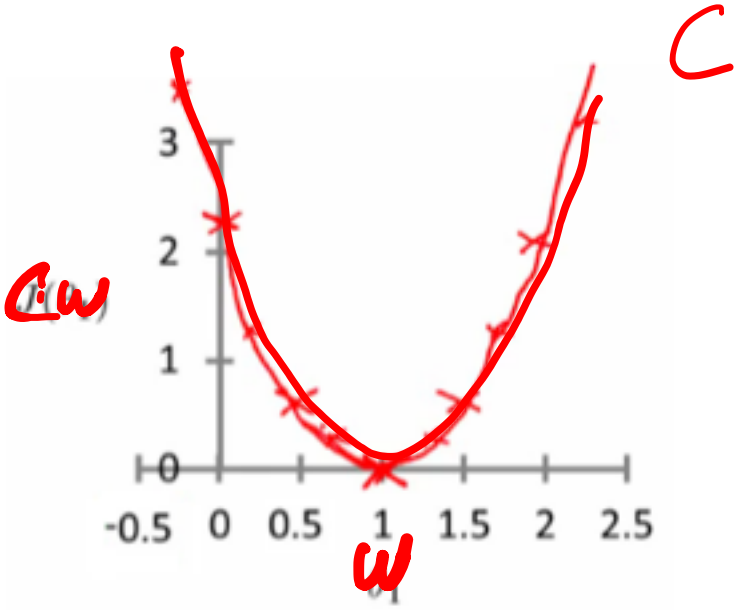


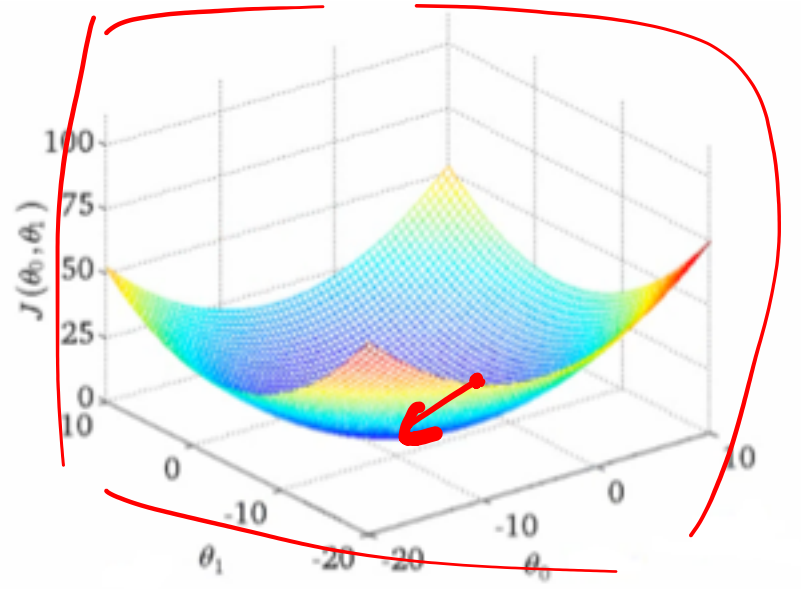


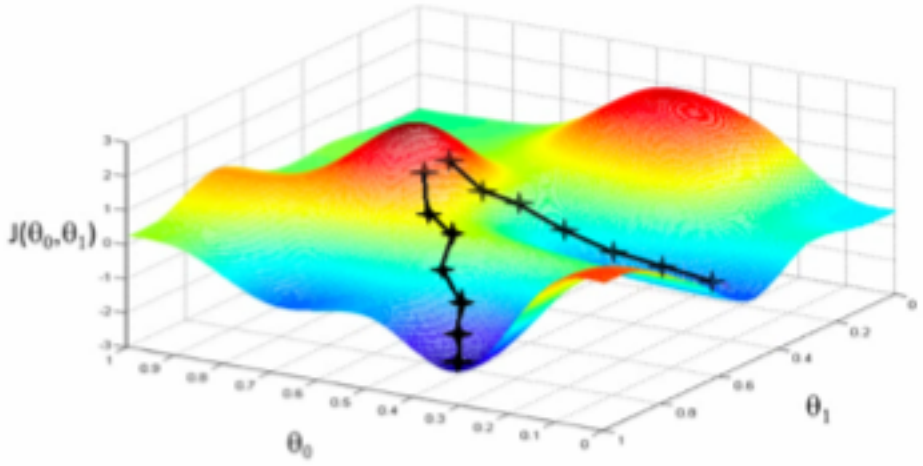
Recap: Linear Regression

- 训练数据是m个 (x^i, t^i) 数据对， X^i 是 n 纬向量
- $y_\omega(x) =$ $w^T x = w_0 + w_1 x_1 + \cdots w_n x_n$
- $C(\omega) =$ $\frac{1}{2m} \sum_i (y_i - t_i)^2$
- 目标？ $\min C(w)$

Recap: 1D case

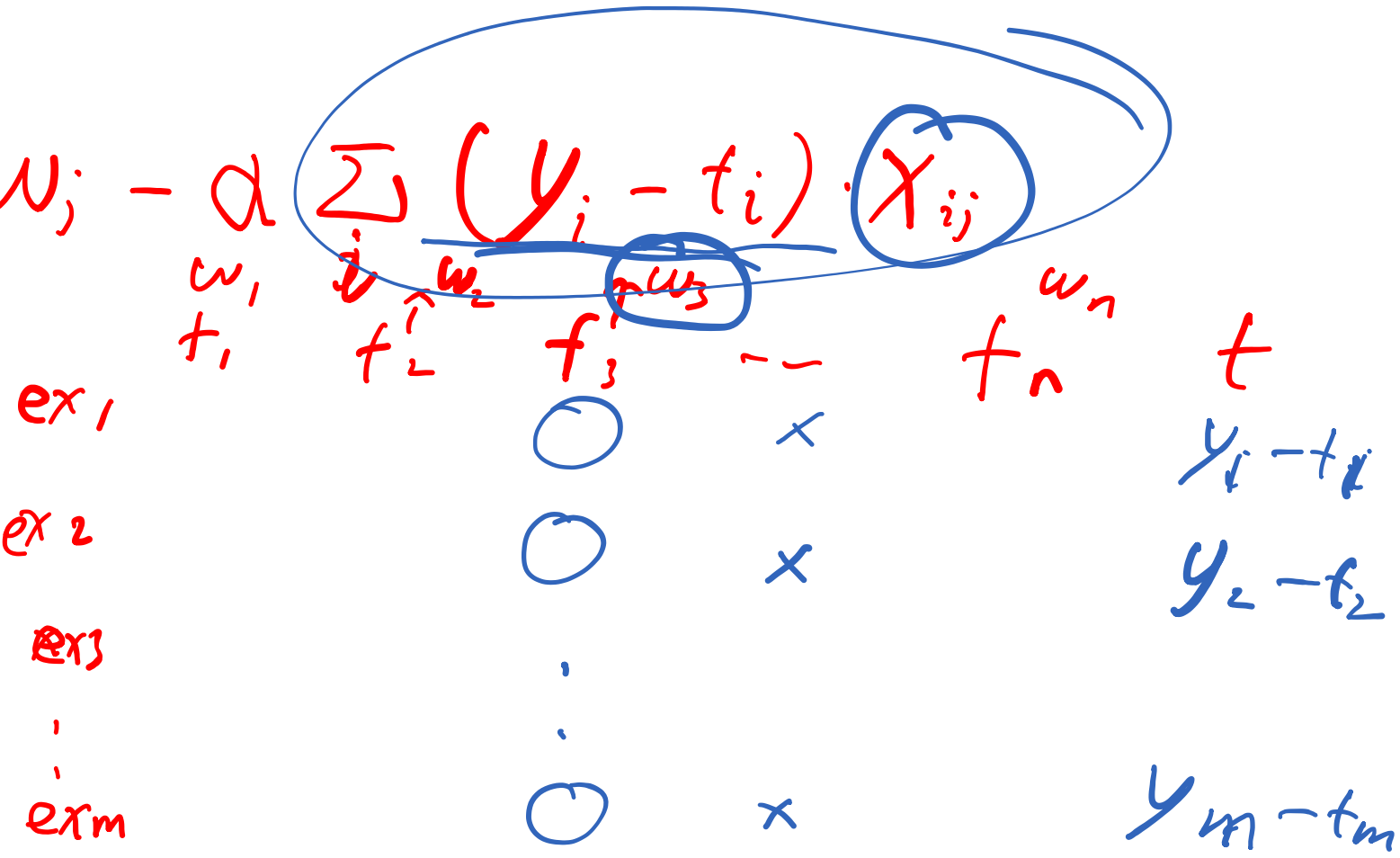




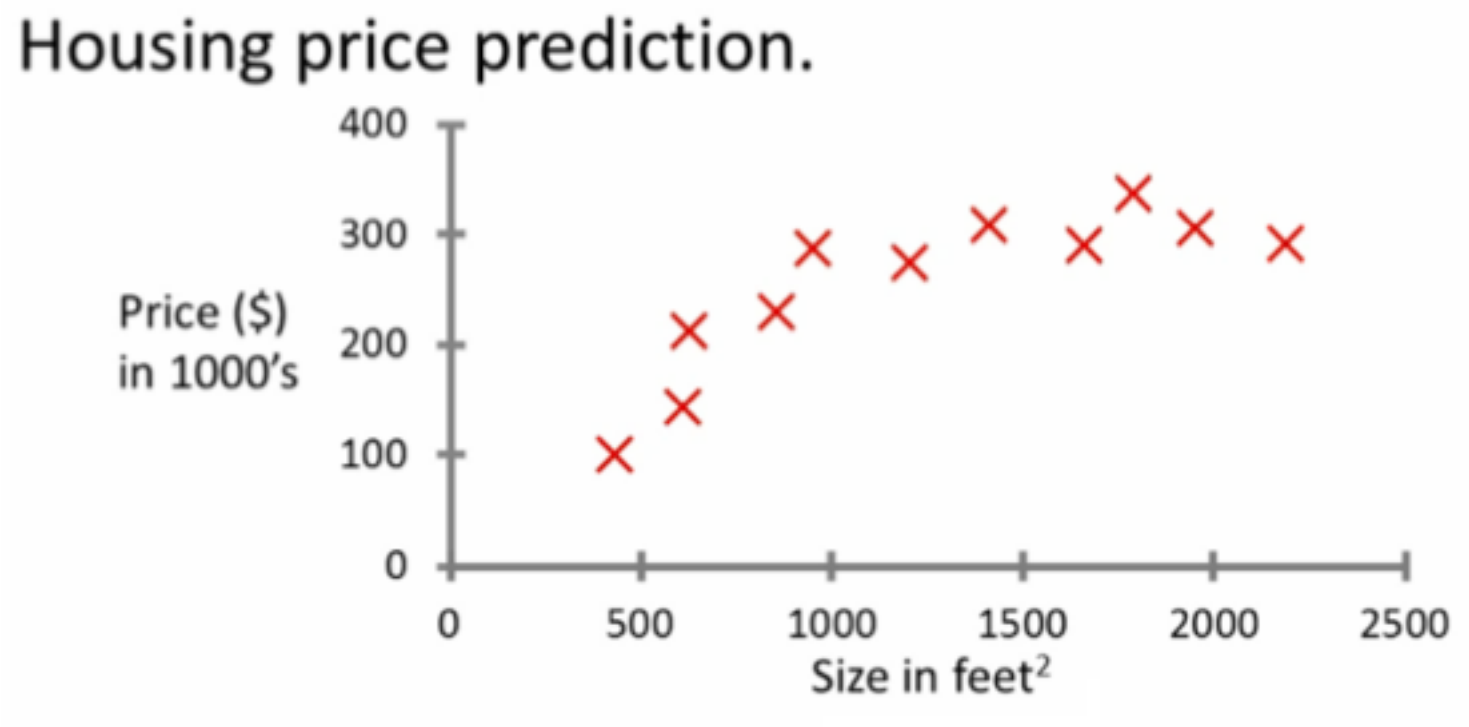


Recap: Gradient Descent

- $\omega_j :=$ $\omega_j - \alpha \sum_i (y_i - t_i) \cdot x_{ij}$

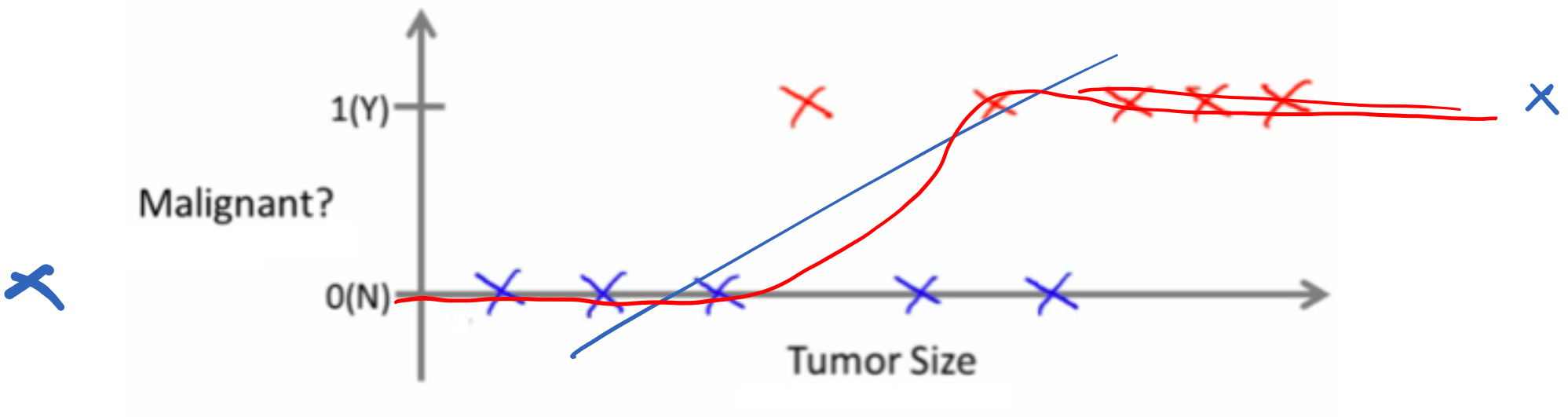


Recap: Polynomial Regression



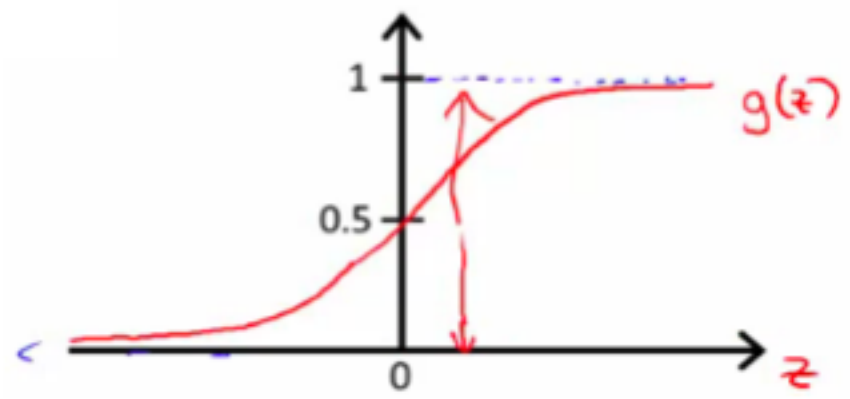
x_1 x_1^2 x_1^3 ——

Recap: Logistic Regression

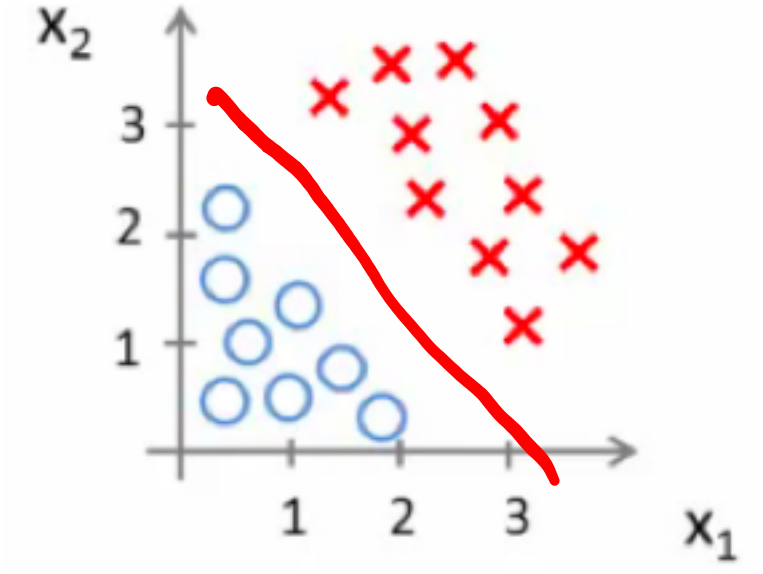


Recap: Logistic Function

$$g(z) = 1/(1 + e^{-z})$$



$$y_\omega(x) = \frac{1}{1 + e^{-\omega^T x}}$$



Recap: Logistic Regression

- 训练数据是m个 (x^i, y^i) 数据对， x^i 是 n 纬向量
- $y_\omega(x) =$ $\frac{1}{1+e^{-\omega^T x}}$
- $C(\omega) = \frac{1}{m}\sum_i \left(t_i \log y_\omega(x_i) + (1-t_i) \log(1-y_\omega(x_i)) \right)$
  $= -\frac{1}{m}\sum_i t_i \log y_i$
- 目标？

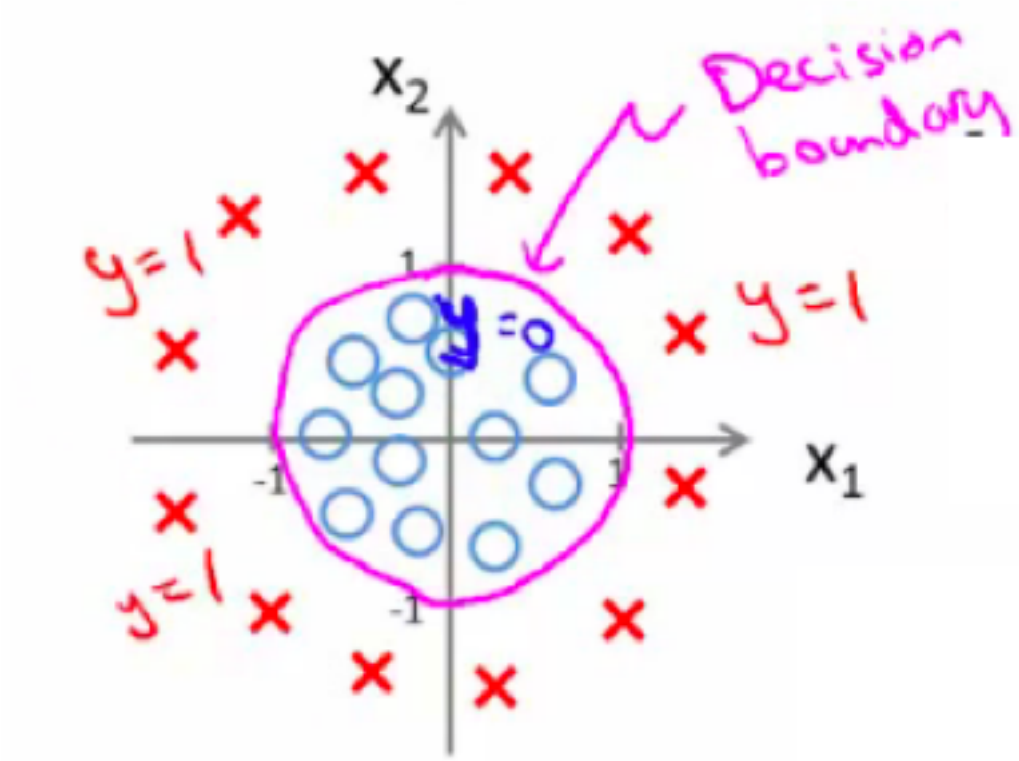
$\min_\omega C(\omega)$

Recap: Gradient Descent

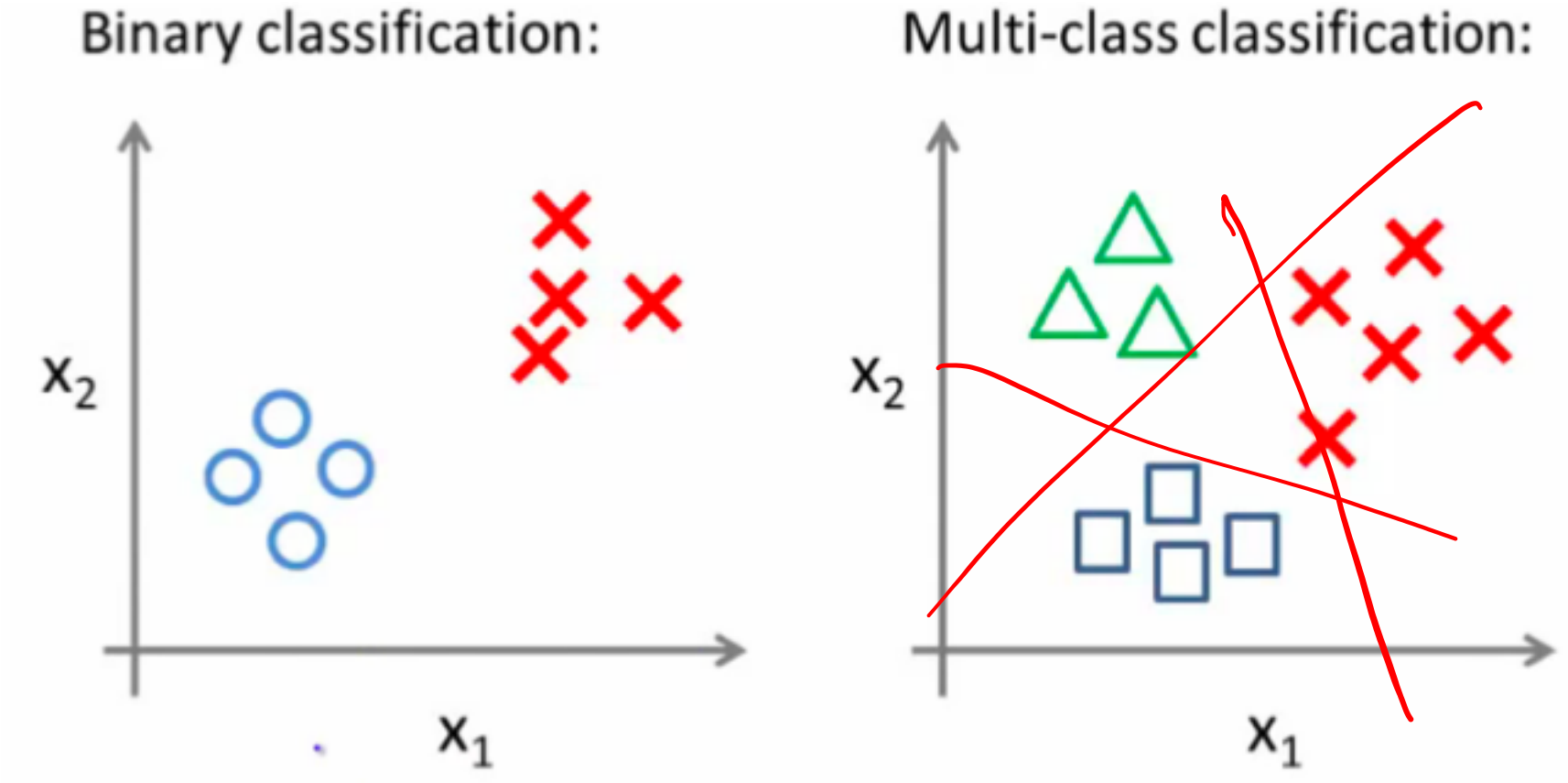
- $\omega_j := \omega_j - \alpha \Delta \omega_j$

$= \omega_j - \alpha \sum_i (y_i - t_i) x_{ij}$

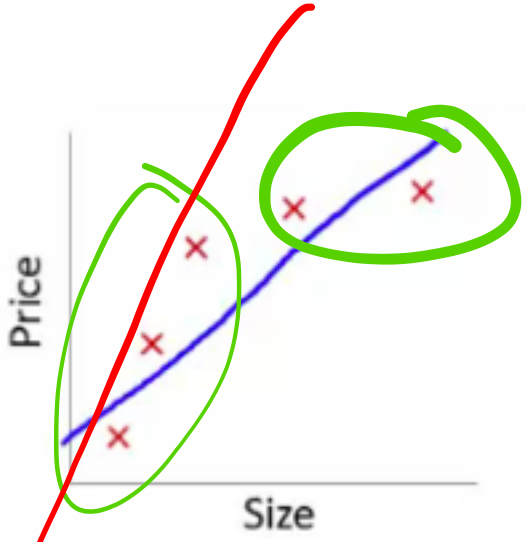
Recap: Non-Linear



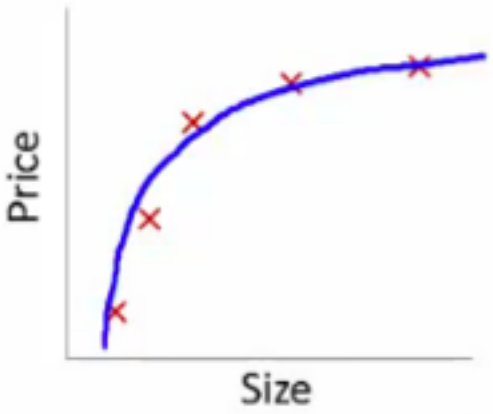
Recap: Multiclass Classification



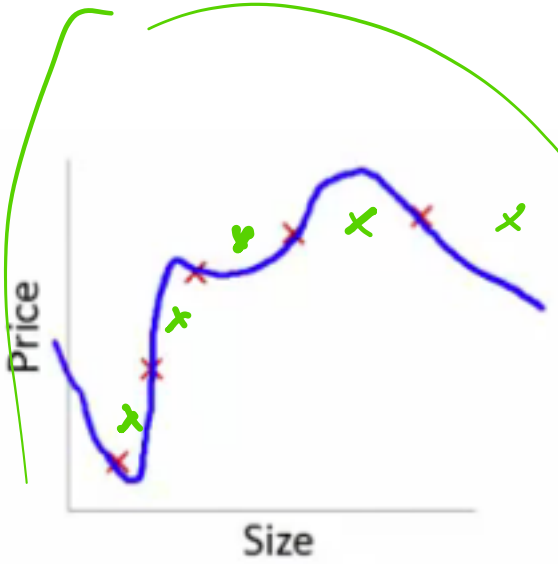
Overfitting: Linear Regression



$y = w_0 + w_1 x$



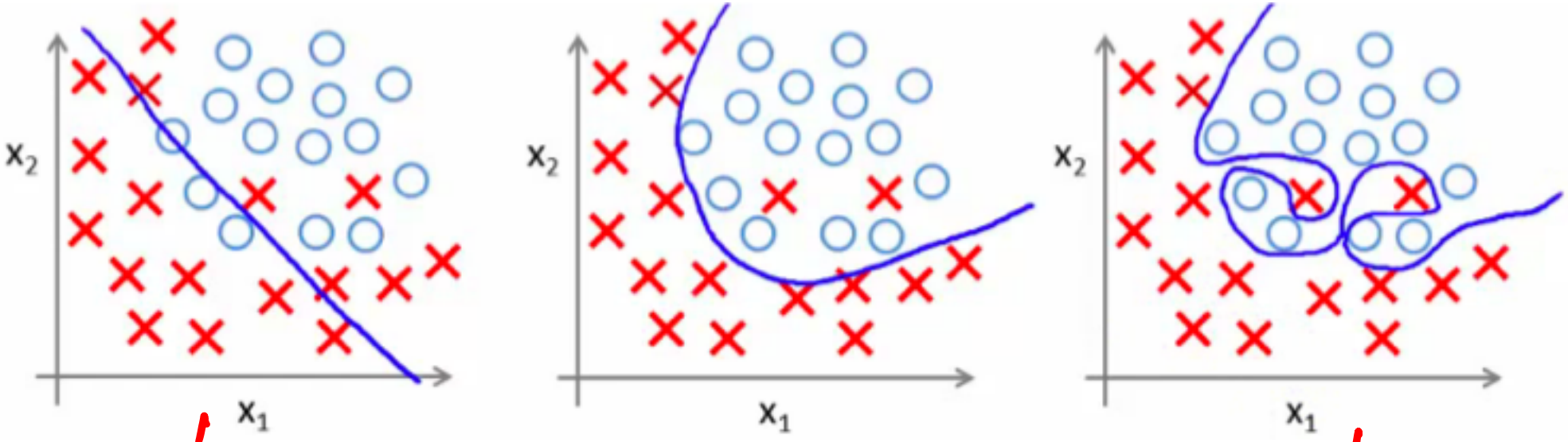
$y = w_0 + w_1 x + w_2 x^2$



$y = - - - - w_3 x^3 \ldots - - w_n x^n$

5-fold CV

Overfitting: Logistic Regression



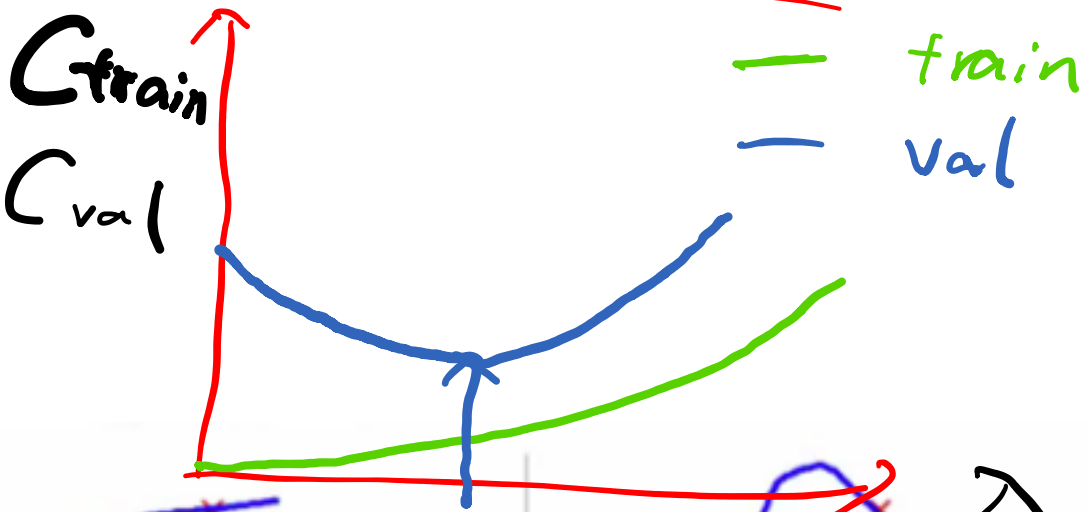
$$\frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2)}}$$

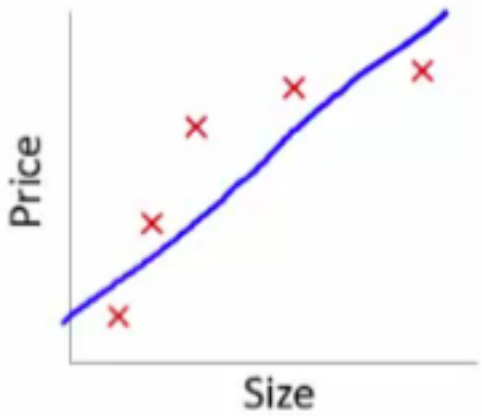
$$w_0+w_1x_1+w_2x_2+w_3x_1^2+w_4x_1x_2+w_5x_2^2$$

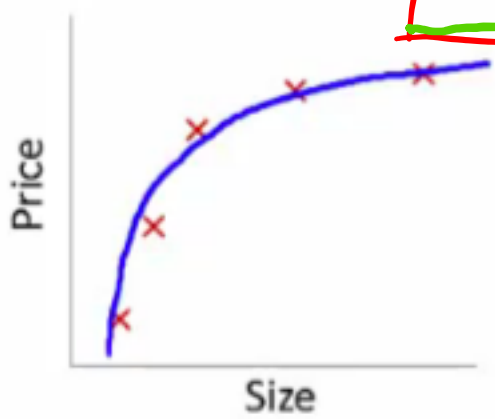
Regularization

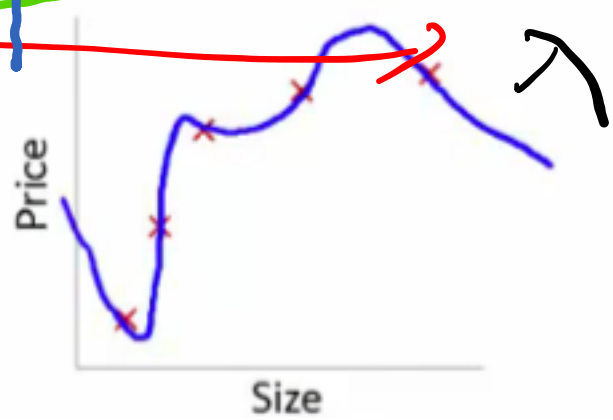
$$C(\omega) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_\omega(x^{(i)}) - t^{(i)}\right)^2 + \lambda \sum_j \omega_j^2$$

$\min_\omega C(\omega)$









Regularized Linear Regression

$$C(\omega) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(y_\omega(x^{(i)}) - t^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \omega_j^2\right]$$

$$w_j := w_j - \alpha \Delta w_j$$

$$\omega_j := \omega_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(y_\omega\left(x^{(i)}\right) - t^{(i)}\right)x_j^{(i)}$$

- Is going to be a number less than 1 usually
- Usually learning rate is small and m is large
  - So the term is often around 0.99 to 0.95

Regularized Logistic Regression

$$C(\omega) = -\frac{1}{m}\sum_{i=1}^{m} t^{(i)} \log\left(y_\omega\left(x^{(i)}\right)\right) + (1 - t^{(i)}) \log\left(1 - y_\omega\left(x^{(i)}\right)\right)$$

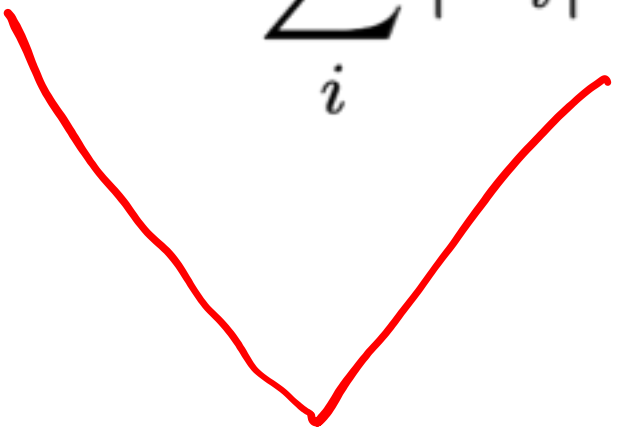
$$+ \lambda \sum_j \omega_j^2$$

$$\omega_j := \omega_j \left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(y_\omega\left(x^{(i)}\right) - t^{(i)}\right)x_j^{(i)}$$

Regularization

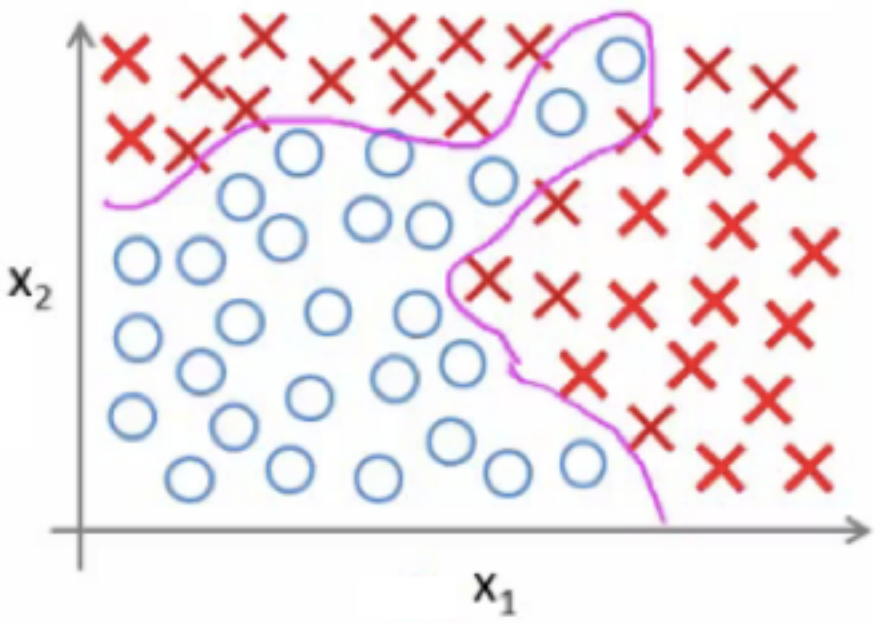
L1 vs. L2

L1: $\min \sum_i |w_i|$



L2: $\min \sum_i w_i^2$

Curse of Dimensionality



x_1^2 x_2^2

$x_1 x_2$

x_1^3 $x_1^2 x_2$ — .

⋮

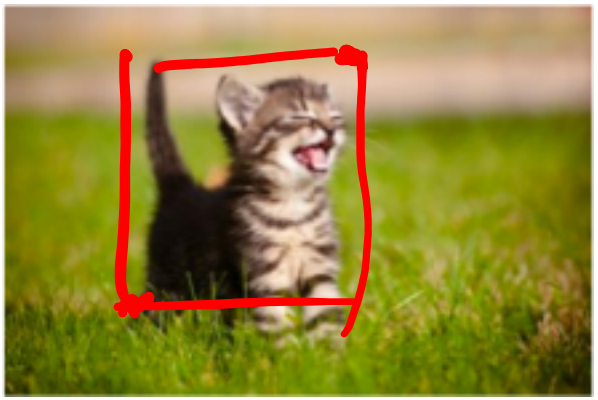
Computer Vision: classification vs. regression

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

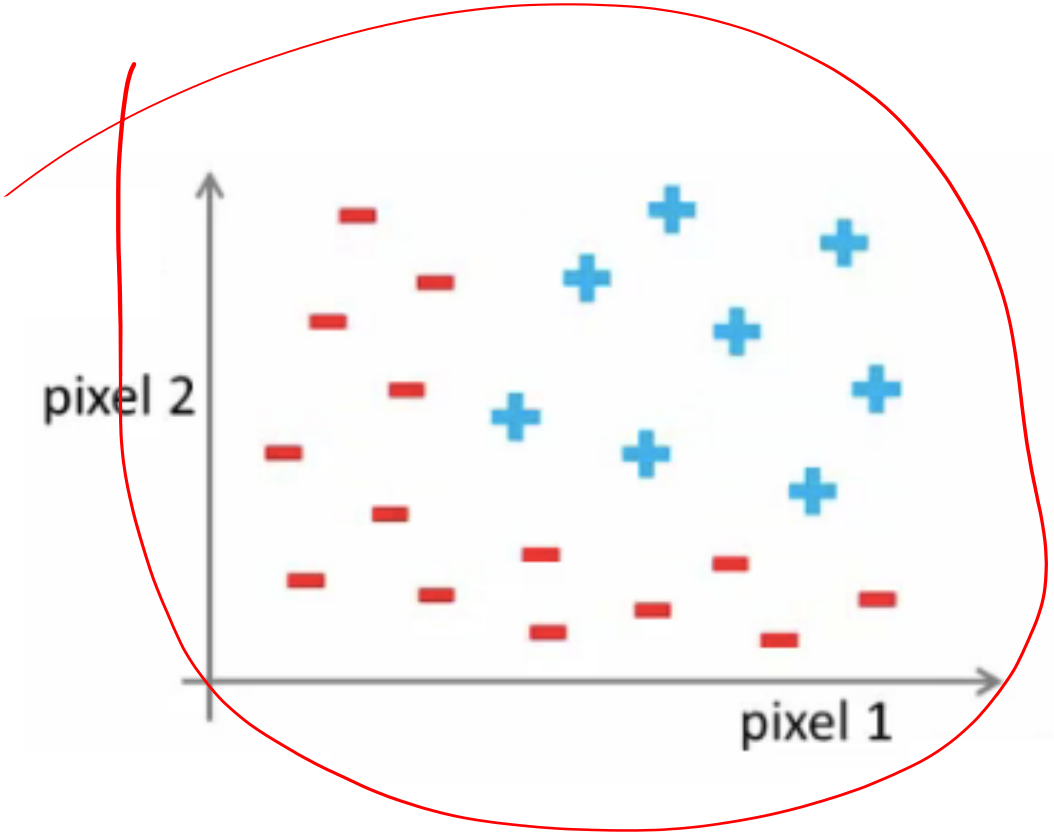
Is there a cat in the image? Yes or No

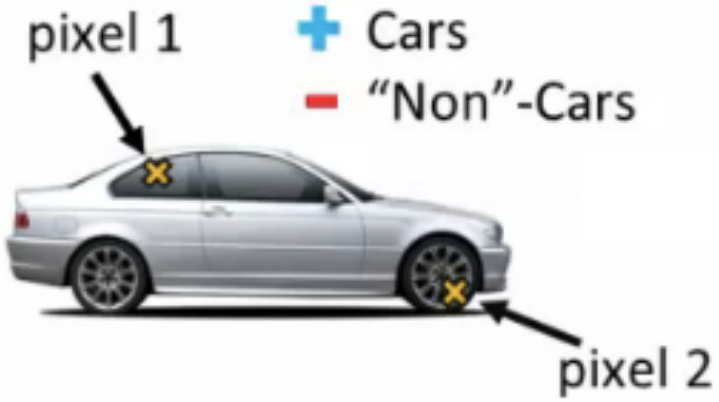
$y \in \Re$ regression

How cute is the cat (scale 1-10)? (9.4)

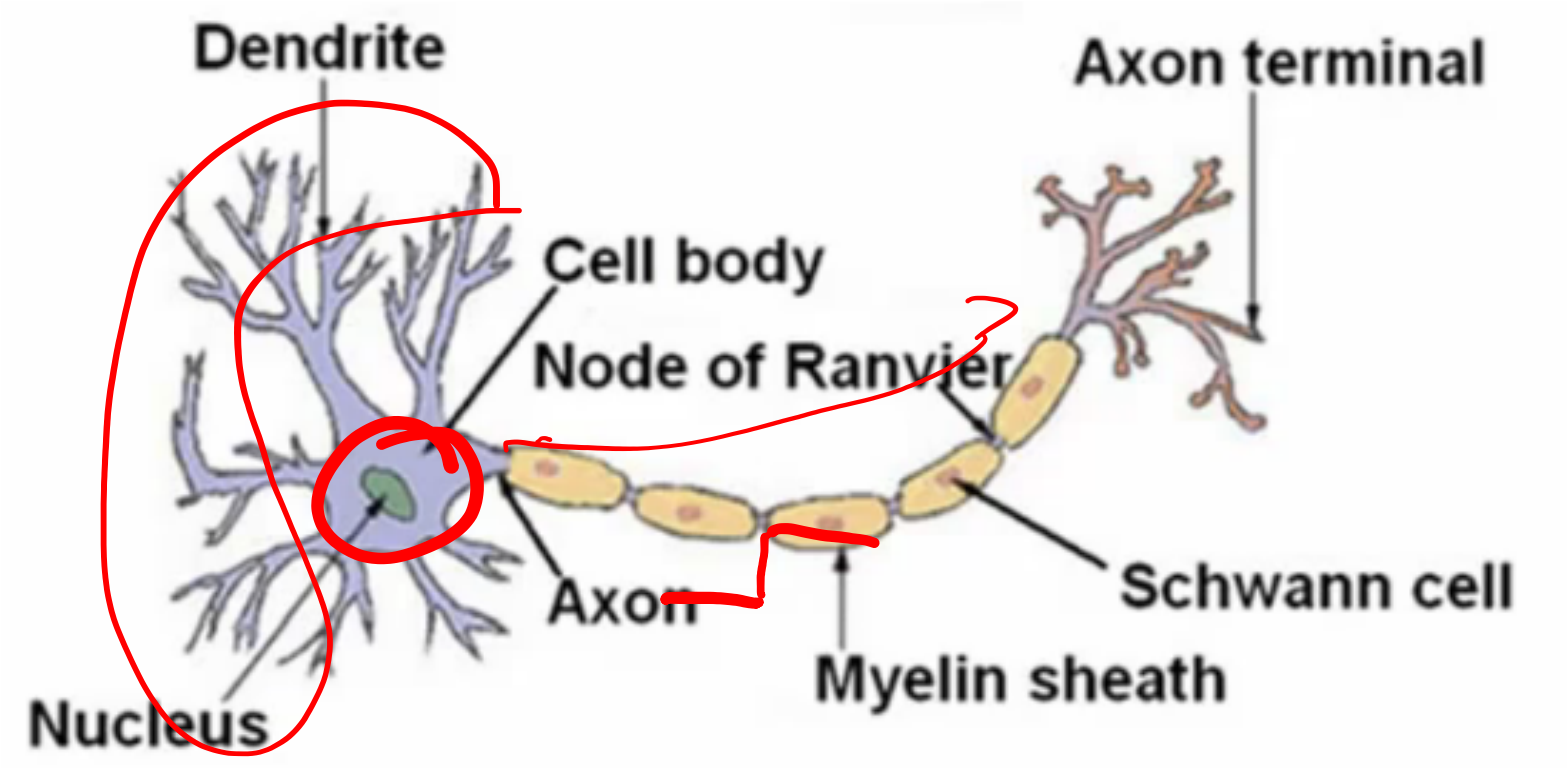


Problems where n is large: Computer Vision





Neuron


Dendrite
Axon terminal
Cell body
Node of Ranvier
Axon
Schwann cell
Myelin sheath
Nucleus

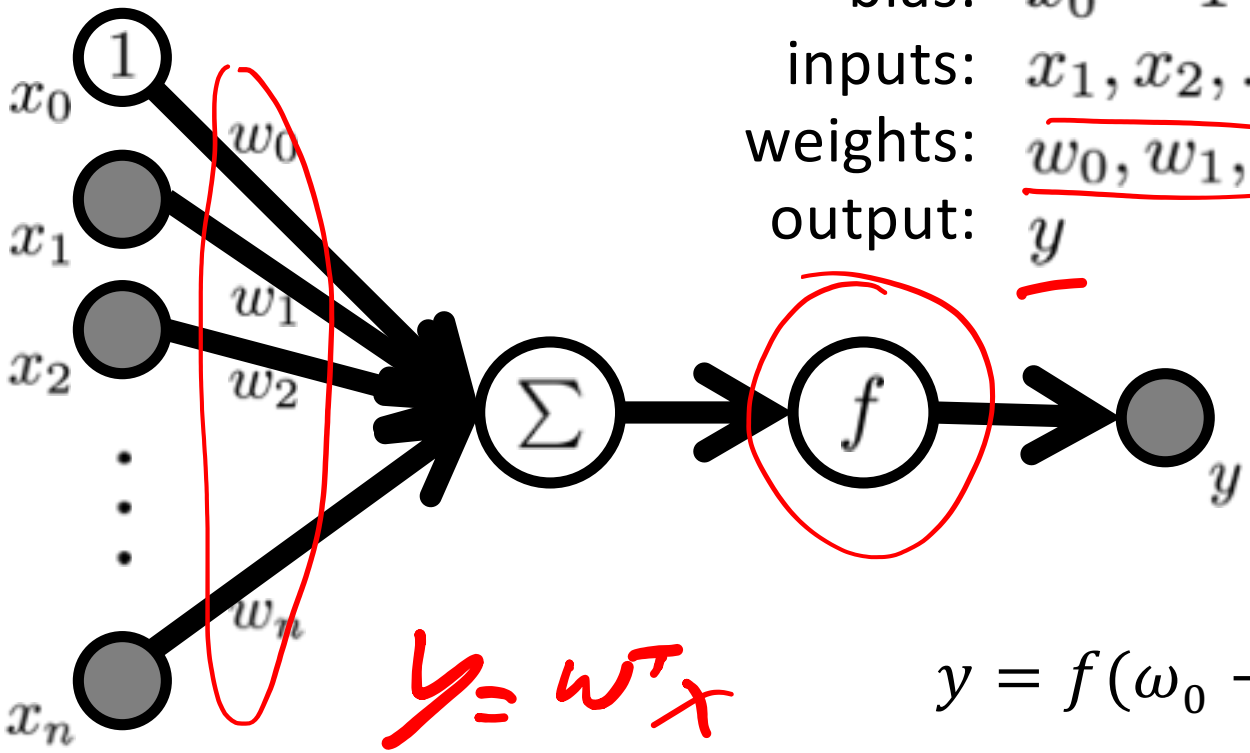
A Single Neuron

bias: $x_0 = 1$

inputs: $x_1, x_2, \ldots, x_n$

weights: $w_0, w_1, \ldots, w_n$

output: y



$y = \omega^T x$

$$y = f(\omega_0 + \omega_1 x_1 + \cdots + \omega_n x_n)$$

$$y = f\left(\sum_{i=0}^{n} \omega_i x_i\right)$$

$$y = f(\omega^T x)$$

Activation function: f

$$y = f(\sum_{i=0}^{n} w_i x_i)$$

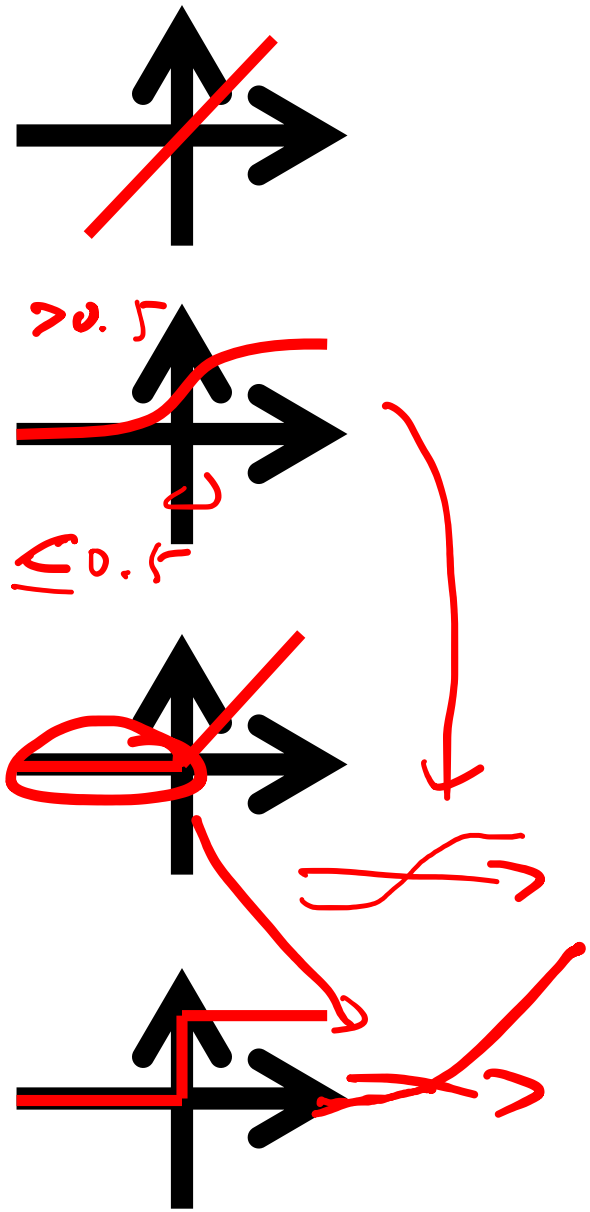
linear: $f(x) = x$

sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

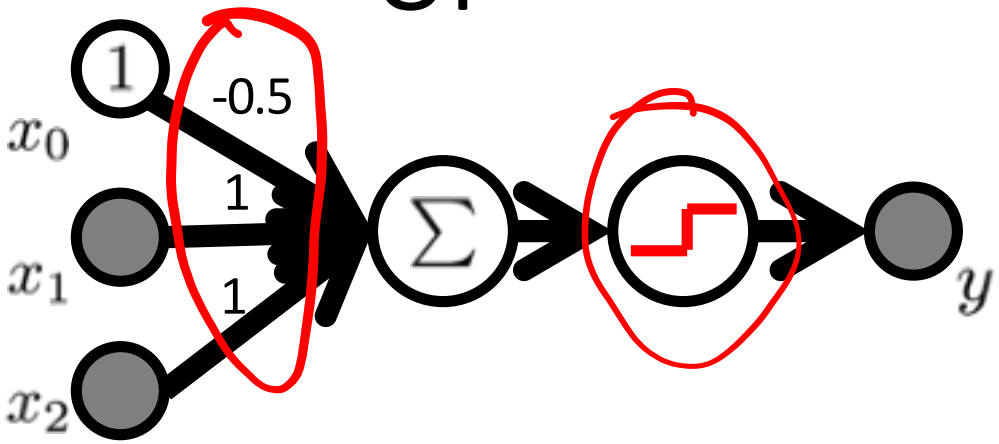
ReLU: $f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$

Rectified Linear Unit

unit step: $f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$

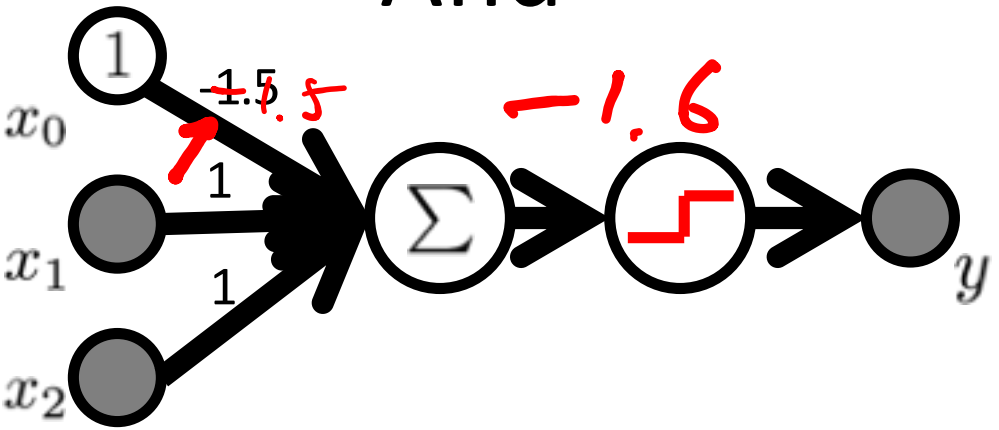


Or



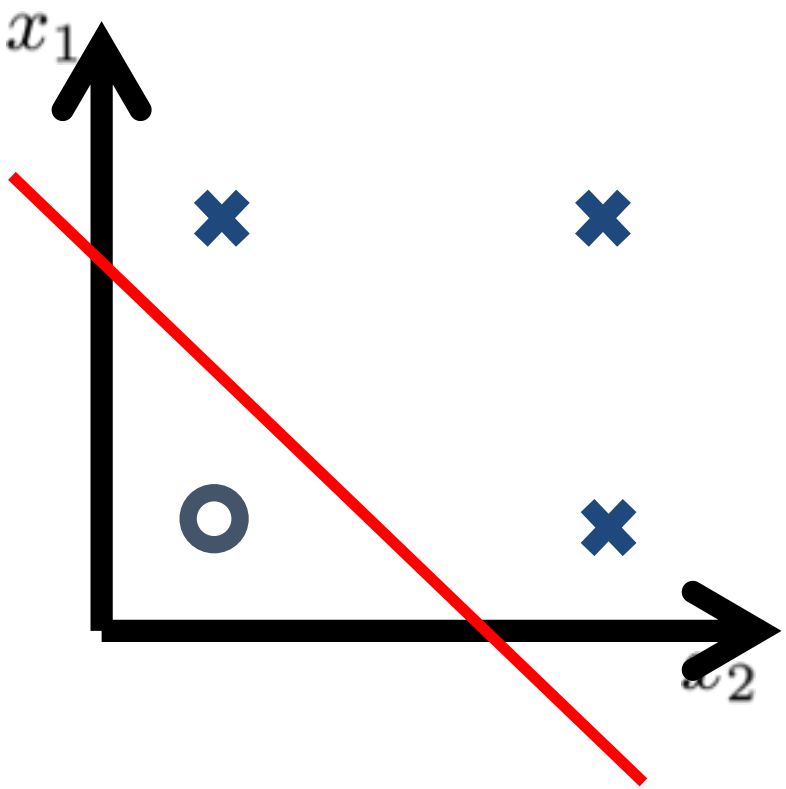
x_1	x_2	$\sum$	y
0	0	-0.5	0
0	1	0.5	1
1	0	0.5	1
1	1	1.5	1

And



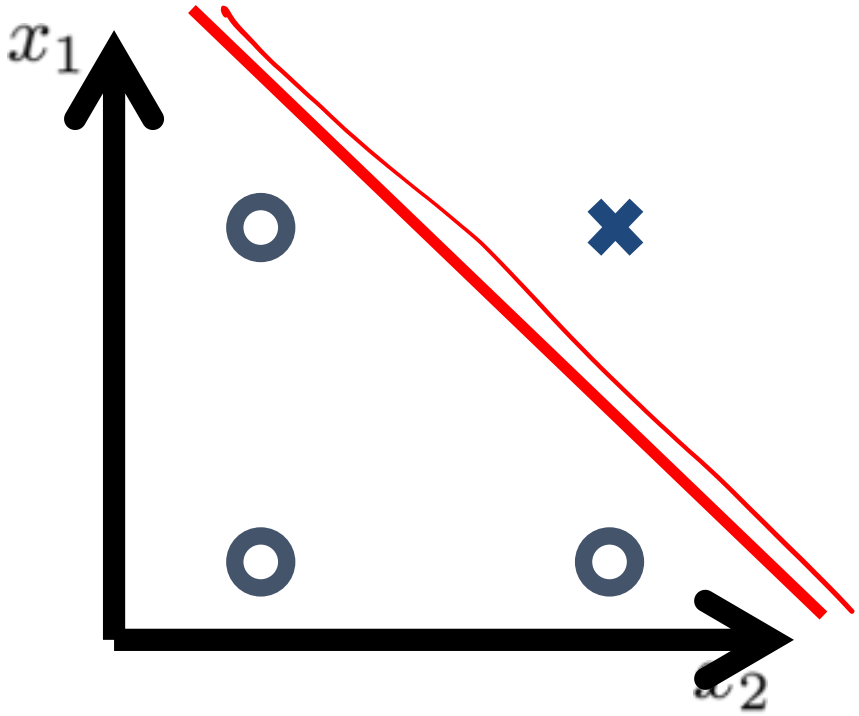
x_1	x_2	$\sum$	y
0	0	-1.5	0
0	1	-0.5	0
1	0	-0.5	0
1	1	0.5	1

Or



$$y = x_2 + x_1 - 0.5$$

And



$$y = x_2 + x_1 - 1.5$$

Objective function

How do we measure our error?

training example: $\{\mathbf{x}_j, t_j\}$

target output: t_j

network parameters: ω

$C(\omega)$

$$\min_{\omega} \sum_{j} (y_j - t_j)^2$$

Also called the error or loss function.

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_j (t_j - y_j)^2$$

$$= (t_j - y_j)\sum_j \frac{\partial}{\partial w_i}(t_j - y_j)$$

$$= (t_j - y_j)\sum_j \frac{\partial}{\partial w_i} f\left(\sum_k w_k x_k\right)$$

$$= -(t_j - y_j)\sum_j f'(\cdot)\, x_i$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_j (t_j - y_j)^2$$

$$= \frac{1}{2}\sum_j \frac{\partial}{\partial w_i}(t_j - y_j)^2$$

$$= \frac{1}{2}\sum_j 2(t_j - y_j)\frac{\partial}{\partial w_i}(t_j - y_j)$$

$$= \sum_j (t_j - y_j)(-\frac{\partial}{\partial w_i} y_j)$$

$$= \sum_j (t_j - y_j)(-f'(net_j)x_{ij})$$

$$= -\sum_j (t_j - y_j) f'(net_j) x_{ij}$$

Function derivatives

		Derivatives
linear:	$f(x) = x$	$f'(x) = 1$
sigmoid:	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
ReLU:	$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$	$f'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$
unit step:	$f(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$	$f'(x) = 0$?

Weight update

new weight

old weight

increment

$$w_i \leftarrow w_i + \Delta w_i$$

step size

derivative of activation function

$$\Delta w_i = \alpha (t_j - y_j) f'(net_j) x_{ij}$$

target

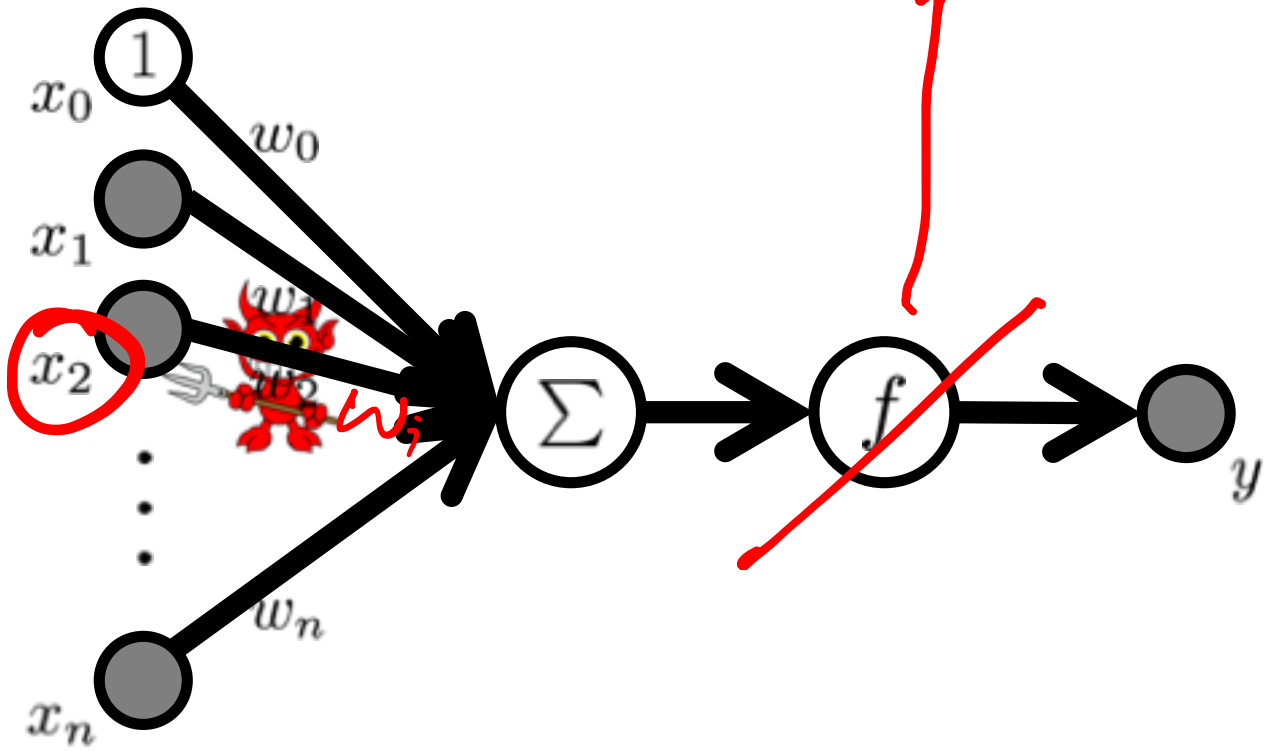
output

input

$$net_j = \sum_{i=0}^{n} w_i x_{ij}$$

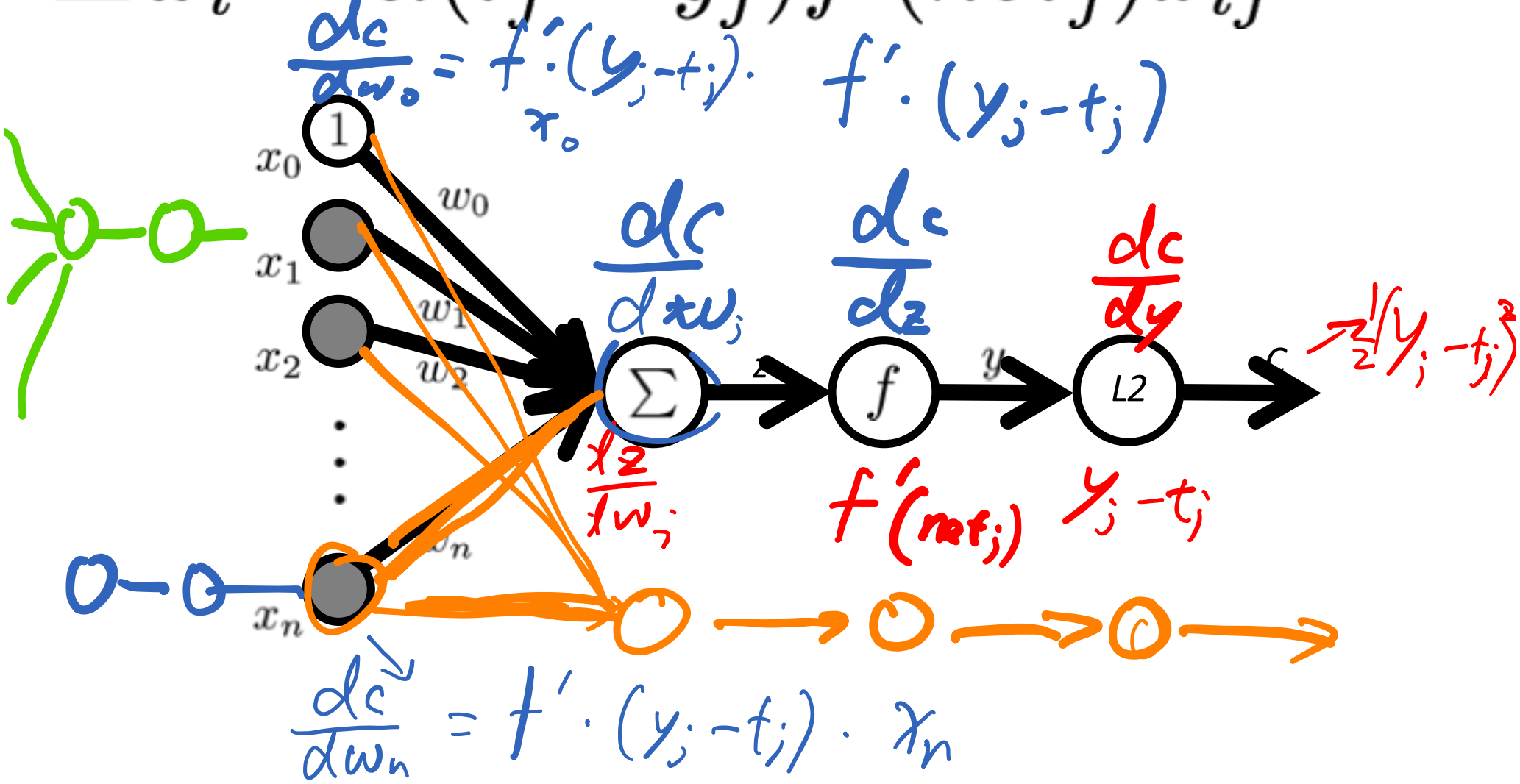
Reducing the error

$$\Delta w_i = \alpha(t_j - y_j) f'(net_j) x_{ij}$$

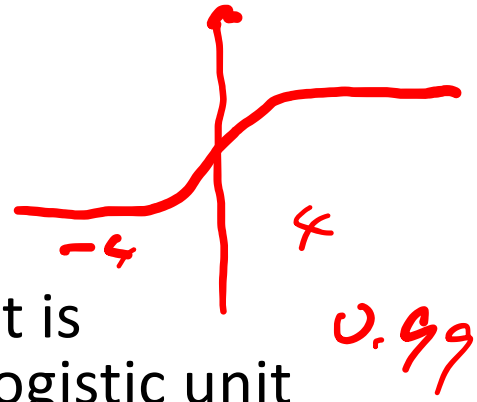


Chain Rule

$$\Delta w_i = \alpha(t_j - y_j) f'(net_j) x_{ij}$$



Problems with logistic activation and squared error



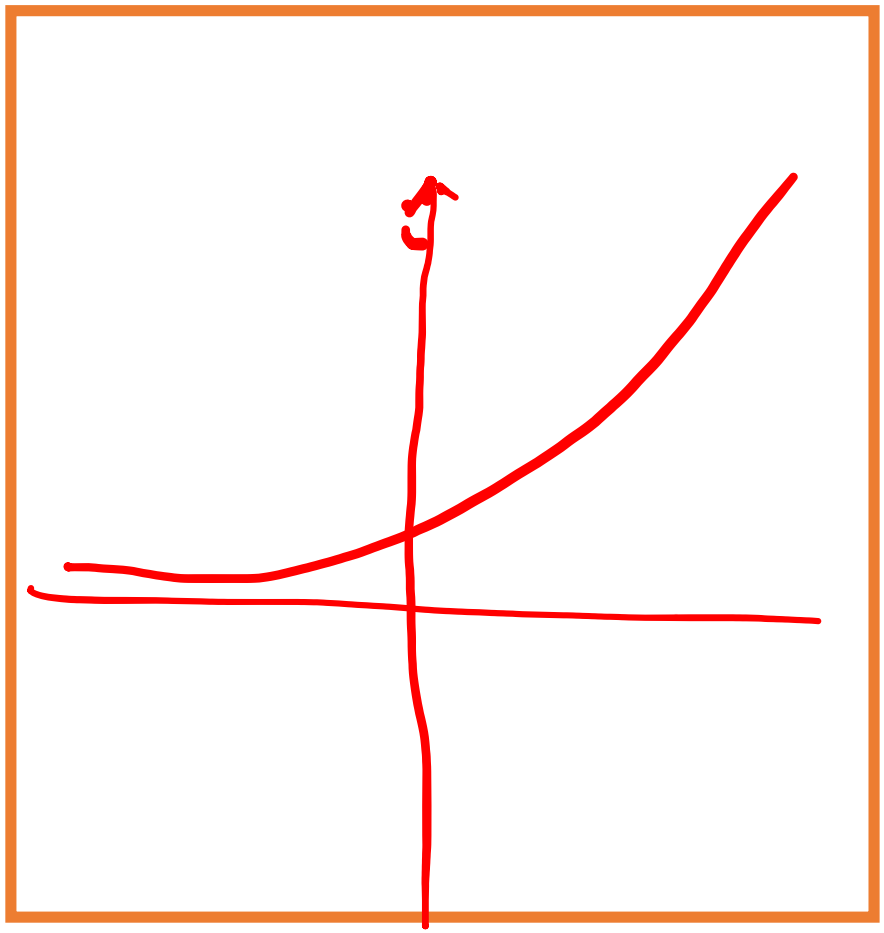
- The squared error measure has some drawbacks:
  - If the desired output is 1 and the actual output is 0.00000001 there is almost no gradient for a logistic unit to fix up the error.
  - If we are trying to assign probabilities to mutually exclusive class labels, we know that the outputs should sum to 1, but we are depriving the network of this knowledge.
- Is there a different cost function that works better?
  - Yes: Force the outputs to represent a probability distribution across discrete alternatives.

Softmax

Used for most classification problems:

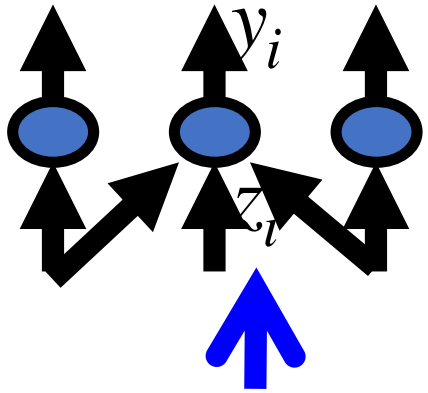
$$f(net_k) = \frac{e^{net_k}}{\sum_l e^{net_l}}$$

$$\sum_{k=1}^{c} y_k = 1$$



Softmax

The output units in a softmax group use a non-local non-linearity:



softmax group

this is called the "logit"

$$z_i = \sum_j w_i x$$

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

$$\frac{\partial y_i}{\partial z_i} = y_i (1 - y_i)$$

Cross-entropy: the right cost function to use with softmax

- The right cost function is the negative log probability of the right answer.
- C has a very big gradient when the target value is 1 and the output is almost zero.
  - A value of 0.000001 is much better than 0.000000001
  - The steepness of dC/dy exactly balances the flatness of dy/dz

$$C = -\sum_j t_j \log y_j$$

target value

$$\frac{\partial C}{\partial z_i} = \sum_j \frac{\partial C}{\partial y_j} \frac{\partial y_j}{\partial z_i} = y_i - t_i$$

$$
\begin{aligned}
& \frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} - \log f(\mathbf{x})_y \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} f(\mathbf{x})_y \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} \text{softmax}(\mathbf{a}^{(L+1)}(\mathbf{x}))_y \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} \frac{\exp(a^{(L+1)}(\mathbf{x})_y)}{\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'})} \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \left(\frac{\frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} \exp(a^{(L+1)}(\mathbf{x})_y)}{\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'})} - \frac{\exp(a^{(L+1)}(\mathbf{x})_y) \left(\frac{\partial}{\partial a^{(L+1)}(\mathbf{x})_c} \sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'}) \right)}{\left(\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'}) \right)^2} \right) \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \left(\frac{1_{(y=c)} \exp(a^{(L+1)}(\mathbf{x})_y)}{\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'})} - \frac{\exp(a^{(L+1)}(\mathbf{x})_y)}{\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'})} \frac{\exp(a^{(L+1)}(\mathbf{x})_c)}{\sum_{c'} \exp(a^{(L+1)}(\mathbf{x})_{c'})} \right) \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \left(1_{(y=c)} \text{softmax}(\mathbf{a}^{(L+1)}(\mathbf{x}))_y - \text{softmax}(\mathbf{a}^{(L+1)}(\mathbf{x}))_y \, \text{softmax}(\mathbf{a}^{(L+1)}(\mathbf{x}))_c \right) \\
= \quad & \frac{-1}{f(\mathbf{x})_y} \left(1_{(y=c)} f(\mathbf{x})_y - f(\mathbf{x})_y \, f(\mathbf{x})_c \right) \\
= \quad & -\left(1_{(y=c)} - f(\mathbf{x})_c \right)
\end{aligned}
$$

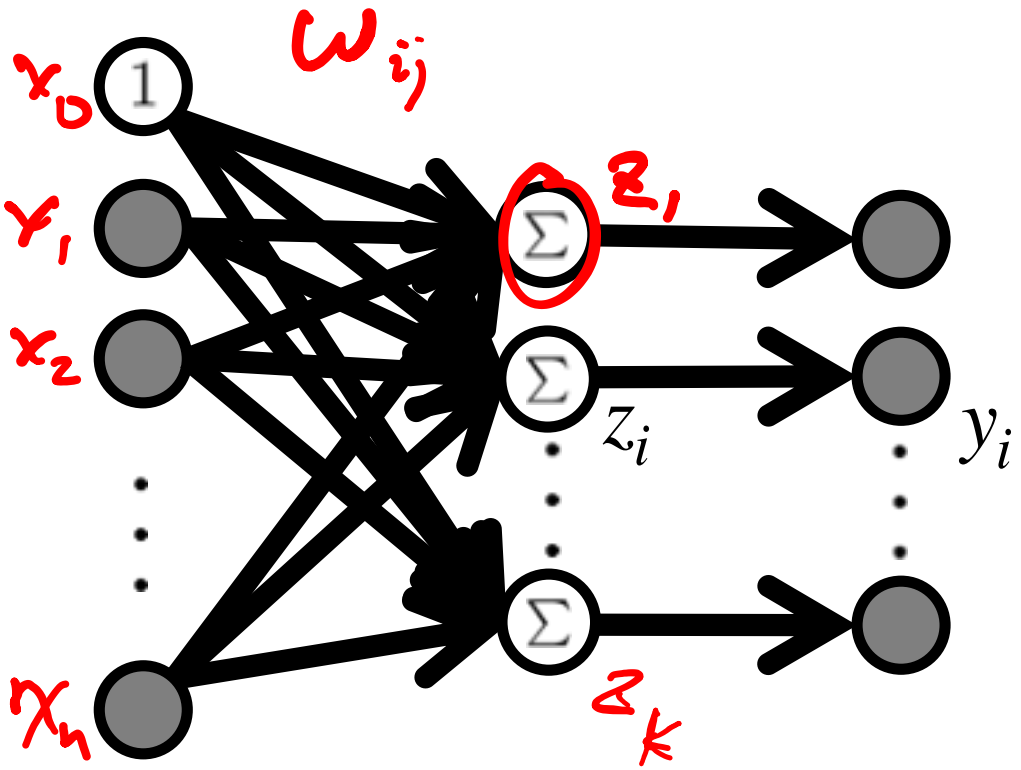
$t_i - y_i$

$$
\frac{\partial \frac{g(x)}{h(x)}}{\partial x} = \frac{\partial g(x)}{\partial x} \frac{1}{h(x)} - \frac{g(x)}{h(x)^2} \frac{\partial h(x)}{\partial x}
$$

(from Hugo Larochelle)

Softmax

$$y_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$



$$\frac{\partial y_i}{\partial z_i} = y_i\,(1 - y_i)$$

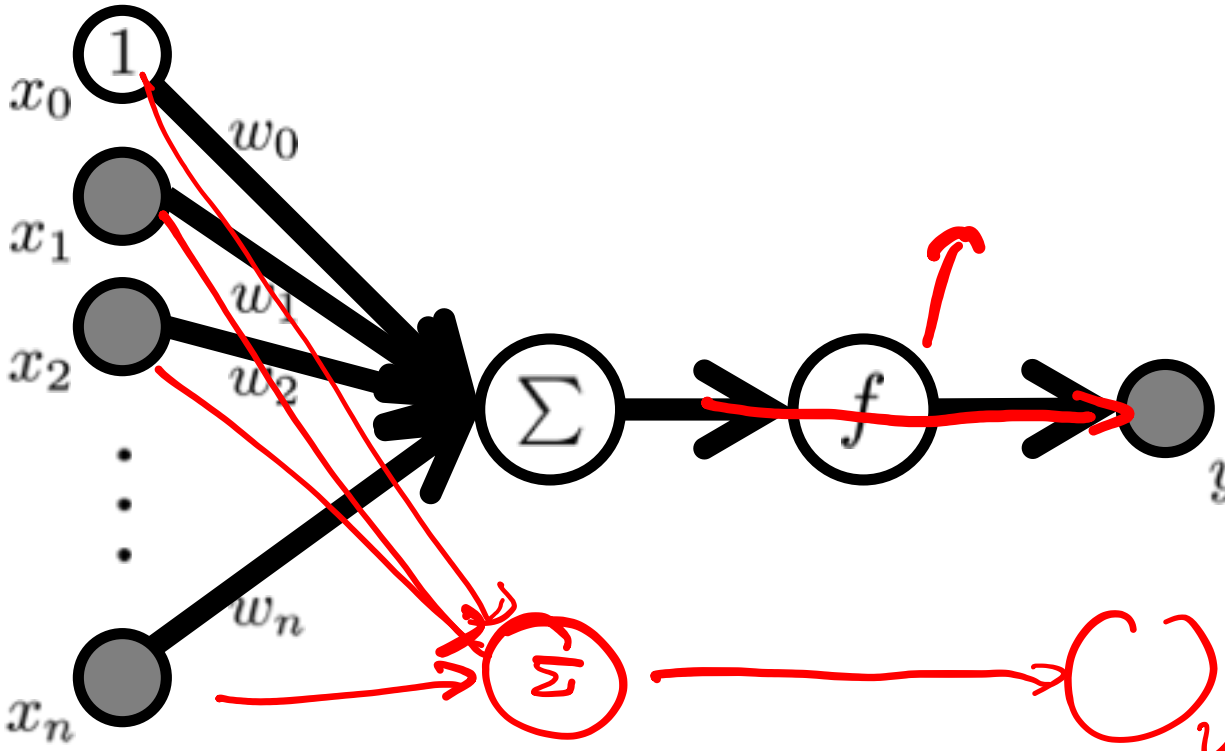
$$C = -\sum_j t_j \log y_j$$

target value

$$\Delta w_i = \alpha(t_j - y_j) f'(net_j) x_{ij}$$

$$\frac{\partial C}{\partial z_i} = \sum_j \frac{\partial C}{\partial y_j}\frac{\partial y_j}{\partial z_i} = y_i - t_i$$

Logistic Regression (special case)



$$\frac{1}{1+e^{-(w_1-w_2)^T x}}$$

$$=$$

$$y_1 = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x}}$$

$$y_2 = \frac{e^{w_2^T x}}{e^{w_1^T x} + e^{w_2^T x}}$$


Q&A