overfitting → too many

- high bias → linear
- high variance → high order

reduce features.

regularization: penalize the params.

(penalize term) $\lambda \sum_{j=1}^{n} \theta_j^2$ → regularization term

regularization parameters

- low → ineffective
- high → underfitting

$$\theta_j = \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta - y)x_j + \frac{\lambda}{m}\theta_j\right]$$

$$= \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta - y)x_j$$

↙ shrinking

gradient descent

normal equation:

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ & 1 & & & 0 \\ & & 1 & & 0 \\ & & & \ddots & \vdots \\ 0 & & & & 1 \end{bmatrix}\right)^{-1} X^T y$$

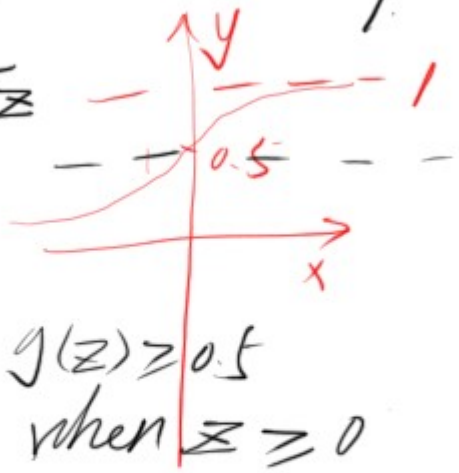
invertible

logistic classification

$y \in \{0, 1\}$ multi $y \in \{0, 1, 2 \dots k\}$

$h_\theta(x) = g(\theta^T x)$ $\quad g = \frac{1}{1+e^{-z}}$

→ estimate "P" of $y=1$



$h_\theta(x) = P(y=1 \mid x; \theta)$

$g(z) \geq 0.5$ when $z \geq 0$

For $\theta^T x \geq 0 \Rightarrow h_\theta \geq 0.5$

cost function

$$\text{cost} = \begin{cases} -\log(h_\theta) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & y=0 \end{cases}$$

simplified cost function

$$\text{cost}(h_\theta(x), y) = -y \log(h_\theta) - (1-y)\log(1-h_\theta)$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} \text{cost}\right]$$

Gradient Descent: $\theta_j = \theta_j - \alpha \sum_{i=1}^{m} (h_\theta - y) x_j$

Neuron model:
logistic unit

$$\frac{1}{1+e^{-\theta^T x}}$$

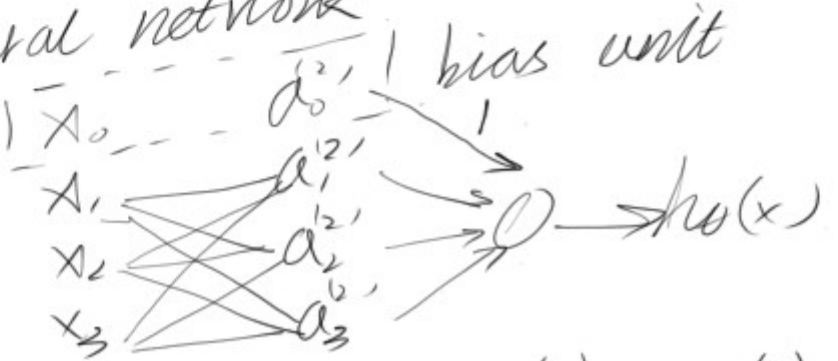
$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$$

"weights"

input

activation function

neural network



$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

$z_1^{(2)}$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$z^{(2)} = \theta^{(1)} x$ ← activation layer $a^{(1)}$

$a^{(2)} = g(z^{(2)})$

add $a_0^{(2)} = 1$ ← bias unit

$z^{(3)} = \theta^{(2)} a^{(2)}$

$h_\theta(x) = a^{(3)} = g(z^{(3)})$

$x \longrightarrow a \longrightarrow h_\theta$
logR logR

layer$_1$ → layer$_2$ → $\cdots$ $h_\theta(x)$

final result

x_1
x_2 → a_i
x_3

multi-class classification

neural network with multiple outputs

$h_\theta = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ $h_\theta = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$

Type 1 Type 2 — —

Training set $(x^{(1)}, y^{(1)})$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix} \cdots \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

cost function

Training set, L, total no of layers

s_l: No of unit in layer l

$h_\theta(x) \in \mathbb{R}^K$ → K outputs

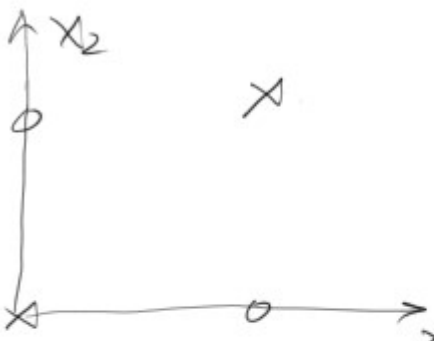
$$J_\theta = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta(x^{(i)})\right)_k + (1-y_k^{(i)}) \log\left(1-(h_\theta(x^{(i)}))_k\right)\right]$$

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

layer → no bias term

Neural network example

non-linear classification



x_1 and x_2 are binary

$y = x_1$ XOR x_2

x_1 XNOR x_2

$x_1, x_2 \in \{0, 1\}$

$y = x_1$ AND x_2

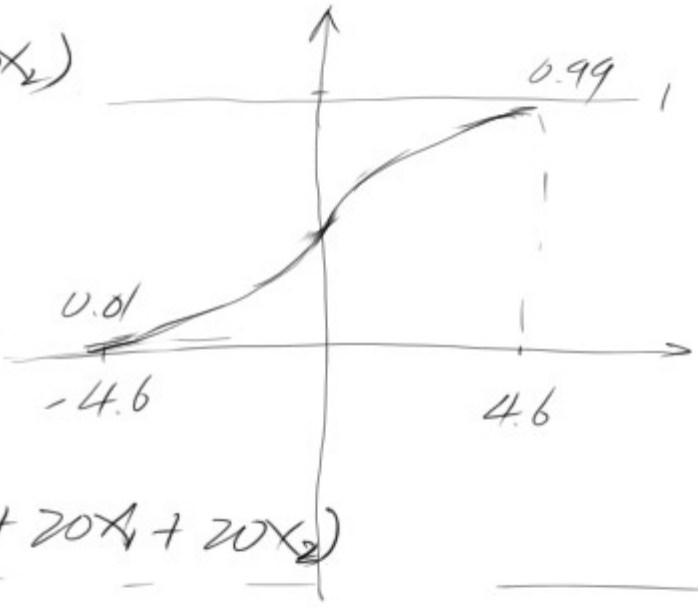
$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$

input all x_1 and x_2

the result will be

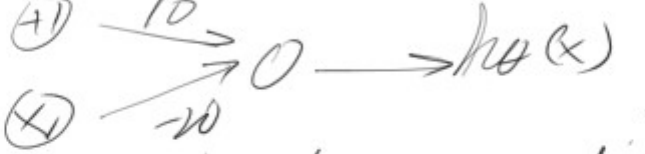
the same as

x_1 AND x_2



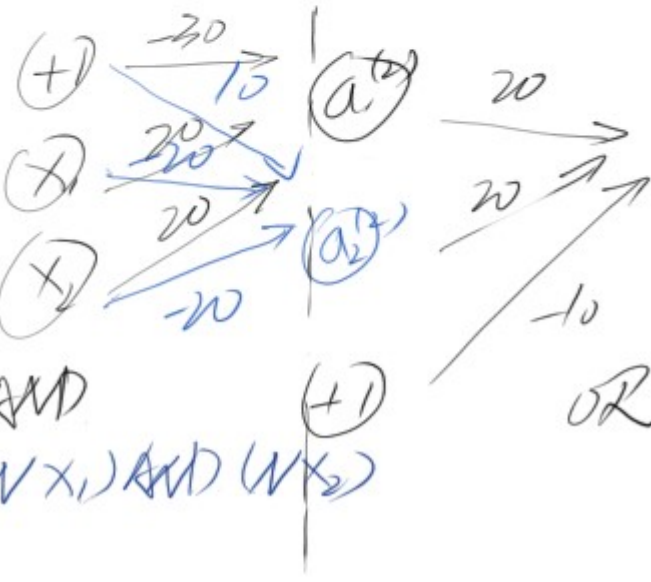
OR $\Leftrightarrow h_\theta(x) = g(-10 + 20x_1 + 20x_2)$

Negation



$h_\theta = g(10 - 20x_1)$

→ large negating factor



AND

(NX_1) AND (NX_2)

OR

NOR

Backpropagation.

$J(\theta)$. $\frac{\partial}{\partial \theta_{ij}} J(\theta)$. required.

δ_j^l = "error" of node j in layer l

$\delta_j^4 = a_j^4 - y_j$ → the value observed, actual

↓ calculated by the hypothesis

$\delta^3 = (\theta^3)^T \delta^4 .* g'(z^3)$ ← $a^3 .* (1 - a^3)$

$\frac{\partial}{\partial \theta_{ij}^l} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$

procedure: $\begin{cases} a^1 = x^{(i)} \\ \text{forward propagation, using hypo.} \\ \text{compute } \delta^{(L)} \cdots \delta^2 \end{cases}$

$\Delta_{ij}^l = \Delta_{ij}^l + a_j^l \delta_i^{l+1}$

$= \Delta^l + \delta^{l+1} (a^l)^T$.

$D_{ij}^l = \frac{1}{m} \Delta_{ij}^l + \lambda \theta_{ij}^l$ for $j \neq 1$.

$\frac{\partial}{\partial \theta_{ij}} J(\theta) = D_{ij}^l$.

unrolling Parameters.

matrix ⇒ vector for optimization.

thetaVec = [theta1(:) ... thetan(:)]

θ^1 θ^n.

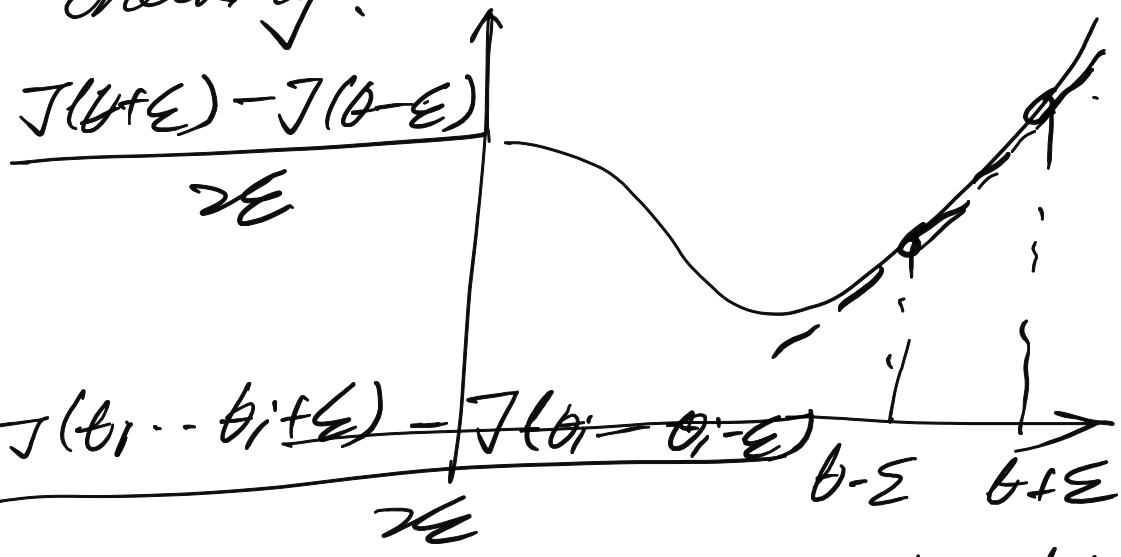
thetaVec ⟹ Theta1

= reshape(thetaVec(1:total #),
hJ, nj)

Gradient checking:

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$$

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{J(\theta_1 \cdots \theta_i + \varepsilon) - J(\theta_i - \theta_i - \varepsilon)}{2\varepsilon}$$



Random Initialization: Symmetry breaking

Small value

1. pick a network architecture.
2. random initial weight
3. FP
4. cost function.
5. BP $\frac{\partial}{\partial \theta} J(\theta)$. — loop $\Delta = \Delta + \delta (a^l)^T$.
6. Gradient checking, numeric checking.
7. minimize cost function.

machine learning diagnostic:
evaluating hypothesis.
split into "training set" and "test set"
"shuffle" the data
$J(\theta)_{training}$ $J(\theta)_{test}$ both low
$\frac{1}{m_{test}} \sum (h_\theta(x) - y_{test})$
For logistic regression same method.
misclassification error, prediction accuracy.

model selection: