SCHEDULE

	MON	TUE	WED	THU	FRI

Important dates

Time Zones (US)

International

New York	1 p.m.
Los Angeles	10 a.m.
London	6 p.m.
Moscow	9 p.m.
Tokyo	3 a.m.
Beijing	2 a.m.
Rome	7 p.m.
Hong Kong	2 a.m.
Bangkok	1 a.m.
Sydney	5 a.m.
Paris	7 p.m.
Cairo	8 p.m.
Mumbai	12 a.m.
Shenzen	2 a.m.
Delhi	12 a.m.
Istanbul	9 p.m.
Dubai	10 p.m.
Vienna	7 p.m.

*Does not take US daylight savings into account

this is so thin, what the fuck
check dis

Week 1 notes

$h_\theta(x)$ → prediction model

y → actual model

$$\underset{\theta_0,\theta_1}{\text{minimize}} \ \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$$

m ← # of training set

most commonly used → squared error function

$x^{(i)}$ ← training set

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 \cdot x^{(i)}$$

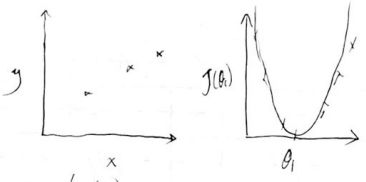
$$\text{cost function} = J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$$

$$\underset{\theta_0,\theta_1}{\text{minimize}} \ J(\theta_0,\theta_1)$$

Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$ **Parameters** θ_0, θ_1

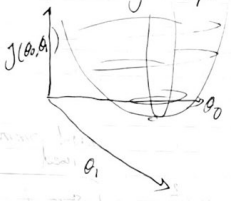
cost function $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Goal : $\underset{\theta_0,\theta_1}{\text{minimize}} \ J(\theta_0,\theta_1)$



this is changing θ_1.

if we change multiple θ, then we get high dimensional graphs



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$\theta_0 := temp0$

$\theta_1 := temp1$

this reposition θ_0, θ_1, and give us a new value for $J(\theta_0, \theta_1)$

A' → transpose

$inv(A)$ → inverse

Week 2 Notes

multiple features ie: $x^{(i)}$ = input (features) of i^{th} training example

$x_j^{(i)}$ = value of feature j

size x_1	# of rooms x_2	floors x_3	age of home x_4	price y
2104	5	1	45	460
1416	3	2	40	232
1534				
852				

$x_1^{(4)} = 852$ → 4^{th} row of feature 1.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \cdots \theta_n x_n$$

example $= 80 + 0.1 \cdot x_1 + 0.01 \cdot x_2 + 3 \cdot x_3 - 2 \cdot x_4$

$x_0^{(i)} = x_0 = 1$

$1^* n+1 \times n+1 \times 1 = 1 \times 1$

if $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$ then $h_\theta(x) = \theta^T \cdot x$

$(n+1) \times (1)$ $\quad$ $(n+1) \times (1)$

Hypothesis $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 \cdots \theta_n \cdot x_n$

Parameters : $\theta_0, \theta_1, \theta_2 \cdots$ aka θ

Cost function : $J(\theta_0, \theta_1, \theta_2 \cdots \theta_n) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$$= \frac{1}{2m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^i \right) - y^{(i)} \right)^2$$

Gradient descent

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

Partial Derivative Demonstration

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

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2 \right]$$

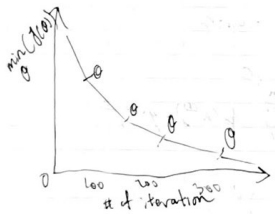
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 x_0 + \theta_1 \cdot x_1 + \ldots + \theta_n \cdot x_n) - y^{(i)} \right)^2 \right]$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left((\theta_0 x_0 + \theta_1 \cdot x_1 + \ldots + \theta_n \cdot x_n) - y^{(i)} \right) \cdot x_j^{(i)} \right]$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

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



brah, this shit trash.

$$\theta = (X^T X)^{-1} X^T y$$

pinv(x' * x) * x' * y

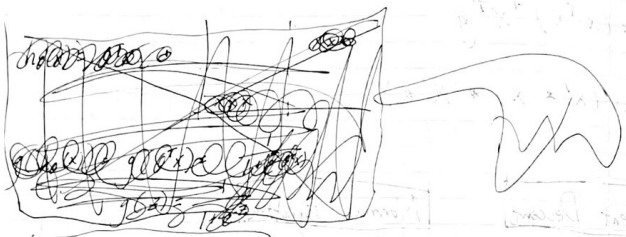
Gradient Descent

Σ, need α,
works well n large.

Normal Equation

matrix, find θ.
no good if n large.

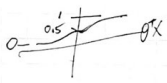
Week 3 Notes



$h_\theta(x) = g(\theta^T x)$

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

$\frac{1}{1+e^{-0}} = \frac{1}{1+1} = \frac{1}{2} = 0.5$



example:

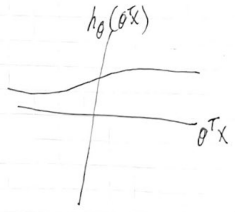
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

example:

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

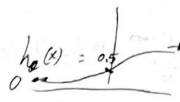
$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$ predict "$y = 1$" if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 \geq 0$



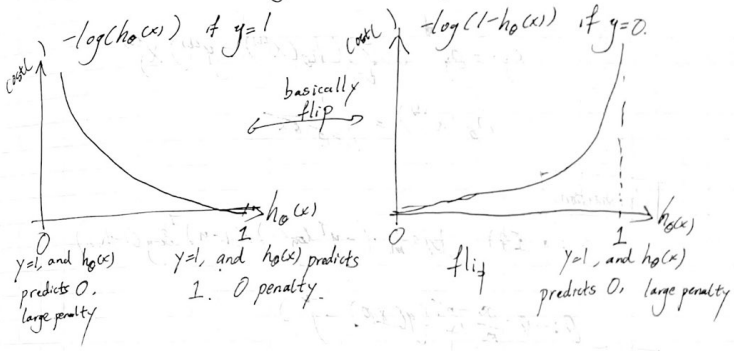
$h_\theta(x) \geq 0.5 \rightarrow y = 1$

$h_\theta(x) < 0.5 \rightarrow y = 0$

Logistic Regression Cost Function



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



$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

no negative

$\rightarrow y \cdot \log(h_\theta(x)) - (1 - y) \cdot \log(1 - h_\theta(x))$

negative

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right)$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1-y^{(i)}) \log\left(1-h_\theta(x^{(i)})\right)\right]$$

$\min_\theta J(\theta)$ repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ $\quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$

}

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

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

Vectorization

$h = g(X\theta)$ $\quad J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$

$$\theta := \theta - \frac{\alpha}{m} X^T \left(g(X\theta) - \vec{y}\right)$$

Regularization

$$J(\theta) = \frac{1}{2m}\left[\left(\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2\right) + \left(\lambda \sum_{j=1}^{n} \theta_j^2\right)\right]$$

$\theta_0 + \theta_1 x + \theta_2 x^2 + 0 \cdot x^3 + 0 \cdot x^4$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ } same format

Repeat {

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_0^{(i)}$ $\quad j=0$

$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$ $\quad j = 1, 2, 3, \ldots, n$

}

↓ can be rewritten as

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

$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}$ → first data.

$m \le n$

\# examples $\quad$ \# features

$$\theta = (X^T X)^{-1} X^T y$$

if $\lambda > 0$

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

use this if no inverse. pinv vs inv

Logistic regression without regularization

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right]$$

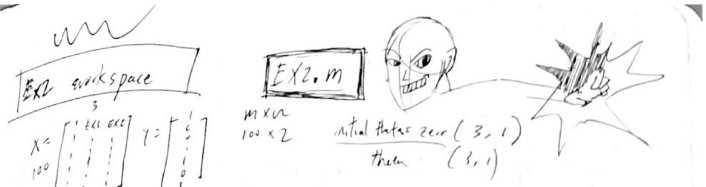
With regularization

$$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

derivative:

$$\theta_0 \rightarrow \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_0^{(i)}$$

$$\theta_j \rightarrow \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)} + \frac{\lambda}{m}\theta_j$$



$X = \begin{bmatrix} 1 & ex1 & ex2 \\ \vdots & \vdots & \vdots \end{bmatrix}$ $y = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$

$m \times n$ 100×2

$h = sigmoid(X \cdot theta)$;

$h = sigmoid\begin{bmatrix} \theta_0 + ex1\cdot\theta_1 + ex2\cdot\theta_2 \\ \vdots \end{bmatrix} \rightarrow h = \begin{bmatrix} \frac{1}{1+e^{-(\theta_0 + ex1\cdot\theta_1 + ex2\cdot\theta_2)}} \\ \vdots \end{bmatrix}$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y^{(i)})$$

$$= \frac{1}{m}\sum_{i=1}^{m} [-y \cdot \log(h_\theta(x)) - (1-y)\cdot \log(1-h_\theta(x))] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Regularization

$$\frac{\partial J(\theta)}{\partial \theta_j} \rightarrow \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)}) \cdot x_{0,j}^{(i)}$$

$h = sigmoid(X * theta)$

octave code

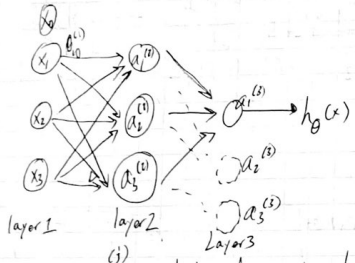
$J(\theta) = \frac{1}{m} sum[-y .* log(h) - (1-y) .* log(1-h)]$

$\frac{\partial J(\theta)}{\partial \theta_j} \quad (\frac{1}{m}) sum[(h - y)' .* X]$

$X .* Y = \begin{bmatrix} 7 \\ 6 \\ 8 \end{bmatrix}$ $X .' * Y = 17$

$X = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ $Y = \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$

Week 4 Notes



$a_i^{(j)}$ = activation of unit i in layer j

$a_1^{(2)}$ = activation of unit 1 in layer 2.

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$.

$$a_1^{(2)} = g(\Theta_{10}^{(1)} X_0 + \Theta_{11}^{(1)} X_1 + \Theta_{12}^{(1)} X_2 + \Theta_{13}^{(1)} X_3) \quad z_1^{(2)}$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} X_0 + \Theta_{21}^{(1)} X_1 + \Theta_{22}^{(1)} X_2 + \Theta_{23}^{(1)} X_3) \quad z_2^{(2)}$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} X_0 + \Theta_{31}^{(1)} X_1 + \Theta_{32}^{(1)} X_2 + \Theta_{33}^{(1)} X_3) \quad z_3^{(2)}$$

$$h_\theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \quad z_1^{(3)}$$

$a_1^{(2)} = g(z_1^{(2)})$ $\quad z_1^{(2)} = \Theta_{10}^{(1)} X_0 + \Theta_{11}^{(1)} X_1 + \Theta_{12}^{(1)} X_2 + \Theta_{13}^{(1)} X_3$

$a_2^{(2)} = g(z_2^{(2)})$

$a_3^{(2)} = g(z_3^{(2)})$

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad a^{(2)} = g(z^{(2)})$$

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

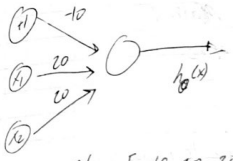
$z_k^{(j)} = \Theta_{k,0}^{(j-1)} X_0 + \Theta_{k,1}^{(j-1)} X_1 + \dots + \Theta_{k,n}^{(j-1)} X_n$

layer $j=2$ node k, variable z.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \qquad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \dots \\ z_n^{(j)} \end{bmatrix}$$

$z_1^{(3)} = \Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}$

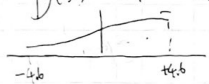
$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$



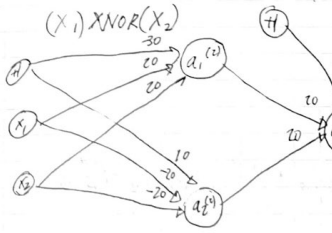
$\Theta^1 = [-10, 20, 20]$

$x_1, x_2 \in \{0, 1\}$ $\quad g(-10 + 20x_1 + 20x_2)$

x_1	x_2	$h_\theta(x)$			
0	0	$g(-10)$	$\to 0$	11	1
0	1	$g(10)$	$\to 1$	10	0
1	0	$g(10)$	$\to 1$	01	0
1	1	$g(30)$	$\to 1$	00	0



(X_1) XNOR (X_2)



x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\theta(X)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

EX3 Workspace

oneVsAll (X, y, num_labels, lambda)

Imagedata — whether it is X or y — 10 — 0.1

5000 × 400 → each image 20x20 Grayscale pixel

5000 × 1

m = 5000
n = 400

all_theta = zeroes Matrix
10 × 402

X = 5000 × 401

10 classes for 10 numbers

$a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$, $a_4^{(2)}$

$a_1^{(2)} = g(\theta_{10}^{(1)} X_0 + \theta_{11}^{(1)} X_1 + \theta_{12}^{(1)} X_2 + \theta_{13}^{(1)} X_3)$

classifier — lrCostFunction (theta, X, y, lambda)

10 different classifier for each number

lrCostFunction (theta, X, y, lambda)

@t   I   X   y==c   lambda

X = [1 2 3 4 ; 5 6 7 8]   X(:) = 1 5 2 6 3 7 4 8

X(1,:) = 1, 2, 3, 4
X(2,:) = 5, 6, 7, 8

options = optimset ('GradObj', 'on', 'MaxIter', 50);
max iteration 50

for c = 1 : num_labels = for c = 1:10.   all_theta = zeros(10, 401)
all_theta (c, :) = ... ← this means continue
fmincg (

401 θs for each pixel intensity

10 × 401
return the θ parameters for each of the 10 classes

for peepee = 1 : size(X, 1)
X(PeePee, :)
1 × 401

all_theta 10×401

Predict One Vs All (all_theta , X)

X = 5000 x 400

↓

X = 5000 x 401 → 1 x 401

all_theta = 10 x 401

X = 5000 x 400

m = 3000

num_labels = 10

p = 5000 x 1

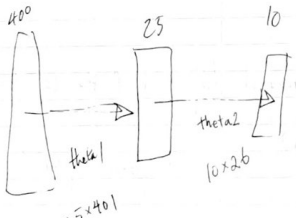
theta1 = 25 x 401

theta2 = 10 x 26

X * all_theta'

p(1,:) = X(1,:) * (all_theta')

gives you a 1x10 matrix of probabilities for the first image 20x20.



25 x 401

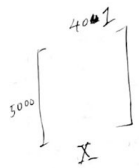
↓

25 x 1

25 [sig(401)]

↓

25 [1] → 26 [one()]



→ X 401 [X(bigpenis, :)]

theta 1 * 25 x 401.

sig(25 [1]) ⇒ 25 [1]

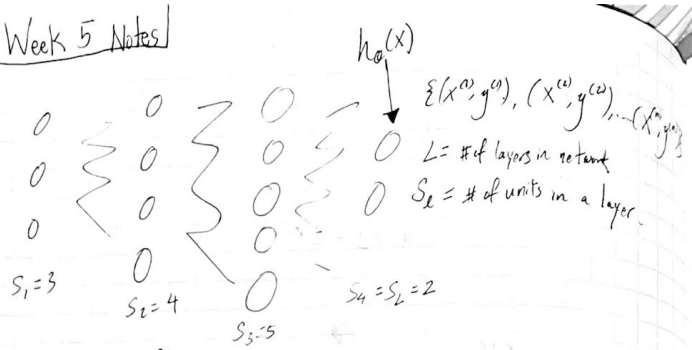
sigmoid [25 x 401 (theta 1) , X(bigpenis, :)']

~~TEXT~~ 26 [1] 10 [26]

bigger penis. theta 2

(Theta 2 * biggerpenis) = 10 x 1 10 [1]

Week 5 Notes



Binary Classification

$y = 0$ or 1

$S_L = 1$ $\quad h_\theta(x) \in \mathbb{R}$

Multi Class classification (k classes)

$y \in \mathbb{R}^k$ $\quad \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

$S_L = K$ $\quad k \geq 3$

$K \times K_k$

Logistic Regression

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1-y^{(i)}) \log\left(1-h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Neural Network

$h_\theta(x) \in \mathbb{R}^K$ $\quad (h_\theta(x))_i = i^{th}$ output

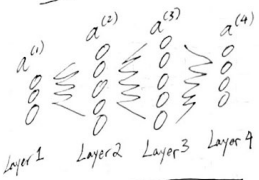
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta(x^{(i)})\right)_k + (1-y_k^{(i)}) \log\left(1-h_\theta(x^{(i)})\right)_k\right]$$
$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}} \left(\theta_{ji}^{(l)}\right)^2$$

Gradient computation:

$\min_\theta J(\theta)$ $\quad$ compute $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$ $\quad \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

Given one training set (x, y)

Forward propagation



activation values

$a^{(1)} = x$

$z^{(2)} = \theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^{(2)})$ (add $a_0^{(2)}$)

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

$a^{(3)} = g(z^{(3)})$ (add $a_0^{(3)}$)

$z^{(4)} = \theta^{(3)} a^{(3)}$

$a^{(4)} = h_\theta(x) = g(z^{(4)})$

Backward Propagation

$\delta_j^{(l)}$ = the error of $a_j^{(l)}$ ← j^{th} node, l^{th} layer

$\delta_j^{(4)} = a_j^{(4)} - y_j$ $\quad$ vectorized → $\delta^{(4)} = a^{(4)} - y$

$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$ ← derivative

$z^{(3)} = \begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \\ \vdots \\ z_n^{(3)} \end{bmatrix}$ → $\theta_{10}^{(1)} a_0^{(2)} + \theta_{11}^{(1)} a_1^{(2)} \ldots$

$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$

$g'(z^{(3)}) = a^{(3)} .* (1 - a^{(3)})$

$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$ $\quad$ ignore λ if $\lambda = 0$

$\Delta_{ij}^{(l)}$ ← capital δ, use to capture $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

for $i = 1$ to m

set $a^{(1)} = x^{(i)}$

forward propagation to compute $a^{(l)}$ $l = 2, 3, \ldots, L$

compute $\delta^{(L)} = a^{(L)} - y^{(i)}$ ← actual answer

then $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$ ← going backwards.

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

Two cases

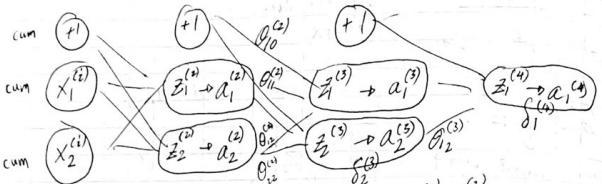
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

complex math proof shows that $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{ij}^{(l)}$

Forward Propagation

$a_0 = 1$ activation value



$(x^{(i)}, y^{(i)})$ $\quad z_1^{(3)} = \theta_{10}^{(2)} \cdot 1 + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} \cdot a_2^{(2)}$

$\delta_j^{(l)}$ = "error" of cost for $a_j^{(l)}$ unit j layer l

$\delta_j^{(4)} = \frac{\partial}{\partial z_j^{(4)}} \cdot \text{cost}(i)$ where $\text{cost}(i) = y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log h_\theta(x^{(i)})$

$\delta_1^{(2)} = g(z^{(2)}) - a_1^{(2)}$ $\quad \delta_2^{(3)} = \theta_{12}^{(3)} \cdot \delta_1^{(4)}$ $\quad \delta_2^{(2)} = \theta_{12}^{(2)} \delta_1^{(3)} + \theta_{22}^{(2)} \cdot \delta_2^{(3)}$

$\delta_2^{(2)} = \delta_2^{(3)} + \theta_{22}^{(2)} \cdot \delta_2^{(3)}$

$\delta_2^{(2)} = \delta_2^{(2)} (1 + \theta_{22}^{(2)})$

↑ whatever output you choose

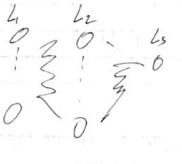
(θ) parameter (D) gradients

$S_1 = 10 \quad S_2 = 10 \quad S_3 = 1$

$\theta^{(1)} \in \mathbb{R}^{10 \times 11} \quad \theta^{(2)} \in \mathbb{R}^{10 \times 11} \quad \theta^{(3)} \in \mathbb{R}^{1 \times 11}$

$S_2 = 10$ nodes $S_1 \to 11$ nodes $+1$

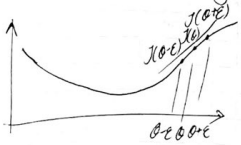
$D^{(1)} \in \mathbb{R}^{10 \times 11} \quad D^{(2)} \in \mathbb{R}^{10 \times 11} \quad D^{(3)} \in \mathbb{R}^{1 \times 11}$




```
thetaVec = [Theta1(:); Theta2(:); Theta3(:)];
DVec = [D1(:); D2(:); D3(:)];
Theta1 = reshape(thetaVec(1:110),10,11);
Theta2 = reshape(thetaVec(111:220),10,11);
Theta3 = reshape(thetaVec(221:231),1,11);
```

← reshape back to original

Gradient could have a bug. So use gradient checking



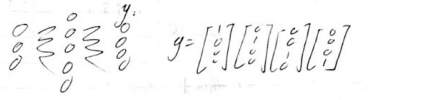
$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$

$\text{gradApprox} = \frac{J(\text{theta} + \text{EPSILON}) - J(\text{theta} - \text{EPSILON})}{2 \cdot \text{EPSILON}}$

$\frac{\partial}{\partial \theta_j} J(\theta) \approx \frac{J(\theta_1 \cdots \theta_j + \varepsilon, \ldots, \theta_n) - J(\theta_1 \cdots \theta_j - \varepsilon \cdots \theta_n)}{2\varepsilon}$

gradApprox ≈ DVec

Putting it together



ex 4 workspace

nnCostFunction(nn_params, input_layer_size, hidden_layer_size, num_label, X, y, lambda)

10

Theta 1 25 × 401

Theta 2 10×26

m = 5000

X = 5000 × 401

g: through X.

1 to 5000

x(i,:) → 1×401

x(1,:) * Theta1 → first 1×25

first * Theta2 = 1×10

yi(1,:) = 1×10

yi(:,1) = 10×1

log(second) * -(yi(:,j))

log(first

X = 5000 × 401

y = 10×10 identity matrix

max(null , 2)

max(null , null)

bak not on deliverable

$500

create practice for XYZ.

x(i,:) = 1×401

Theta1 = 25×401

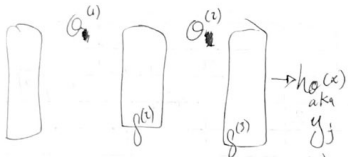
Theta2 = 10×26

25×401 * 1×401 = X = 2×3, Y = 1×3

1×10 * 10×1 = 1×1

10×26 * 26×1

1×26 * 26×10



$$\delta_j^{(3)} = a_j^{(3)} - y_j$$

$$\delta^{(2)} = (\Theta^{(2)})^T \cdot \delta^{(3)} .* g'(z^{(2)})$$

remove $\delta_0^{(2)}$

1 to 5000 t=1:m loop

part 1

part 2

part 3

part 4

end

$(10\times26)^T$ (1×10)

26×10 1×10 $g'(z^{(2)})$ 1×25

00 01 02

10 11 12

20

25 1×25

j=1:end $(10\times25)^T$ (1×10) (1×25)

25×10

Week 6 Notes

Evaluating your hypothesis

size | price

70% training set

30% test set

compute test set error: $J_{test}(\theta) = \frac{1}{2 \cdot m_{test}} \sum_{i=1}^{m_{test}} \left(h_\Theta(X_{test}^{(i)}) - y_{test}^{(i)} \right)^2$

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\Theta(X_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\Theta(X_{test}^{(i)})$$

Model Selection

d = degree of polynomial

d=1 $h_\Theta(x) = \Theta_0 + \Theta_1 x$ $\rightarrow \Theta^{(1)}$ $J_{test}(\Theta^{(1)})$

d=2 $h_\Theta(x) = \Theta_0 + \Theta_1 X + \Theta_2 X^2$ $\rightarrow \Theta^{(2)}$ $J_{test}(\Theta^{(2)})$

d=3 $h_\Theta(x) = \Theta_0 + \Theta_1 X + \Theta_2 X^2 + \Theta_3 X^3$ $\rightarrow \Theta^{(3)}$ $J_{test}(\Theta^{(3)})$

train/validation/test error

Evaluating your hypothesis

size | price

60% training set $(X^{(1)}, y^{(1)})$ $(X^{(2)}, y^{(2)})$ $(X^{(3)}, y^{(3)})$ $\vdots$ $(X^{(m)}, y^{(m)})$

20% cross validation $X_{CV}^{(1)}, y_{CV}^{(1)}$ $\vdots$ $X_{CV}^{(m_{cv})}, y_{CV}^{(m_{cv})}$

20% test set $X_{test}^{(1)}, y_{test}^{(1)}$ $\vdots$ $X_{test}^{(m_{test})}, y_{test}^{(m_{test})}$

Traing error

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\Theta(X^{(i)}) - y^{(i)} \right)^2$$

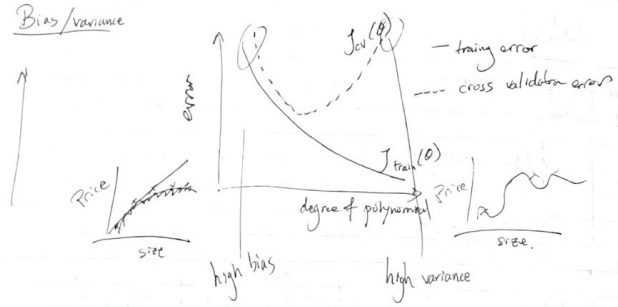
Cross Validation Error

$$J_{cv}(\theta) = \frac{1}{2 \cdot m_{cv}} \sum_{i=1}^{m_{cv}} \left(h_\Theta(X_{CV}^{(i)}) - y_{CV}^{(i)} \right)^2$$

Test Error

$$J_{test}(\theta) = \frac{1}{2 m_{test}} \sum_{i=1}^{m_{test}} \left(h_\Theta(X_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$

Bias/variance

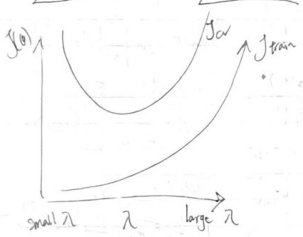


Bias (underfit)
$J_{train}(\theta)$ high
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit)
$J_{train}(\theta)$ low
$J_{cv}(\theta) \gg J_{train}(\theta)$

Linear Regression with Regularization

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \left(\frac{\lambda}{2m}\right)\sum_{j=1}^{n}\theta_j^2$$

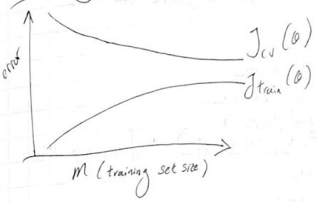


$$J(\theta) = \frac{1}{2m}\cdot\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

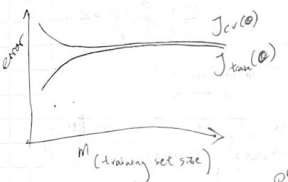
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

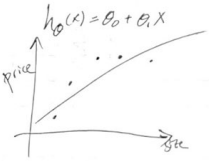
$$J_{cv}(\theta) = \frac{1}{2\cdot m_{cv}}\cdot\sum_{i=1}^{m_{cv}}\left(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)}\right)^2$$

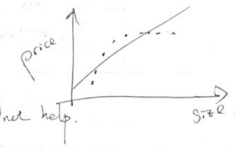
Learning Curves



High Bias

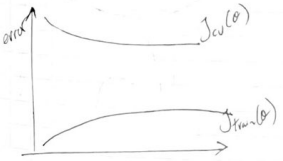






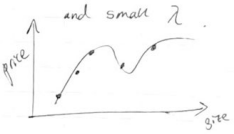
If there is high bias, getting more training data might not help.

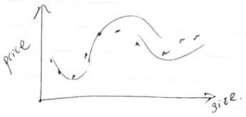
High variance



getting more training data will likely help

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_{100} x^{100}$
and small λ

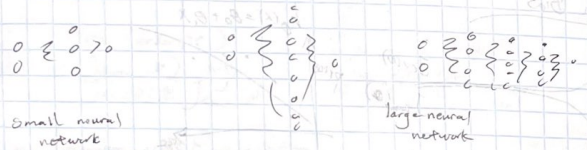




Debugging learning algorithm

Implement regularized linear regression to predict housing prices on new dataset. Not working, now what?

- Get more training examples → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc) → fixes high bias
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance



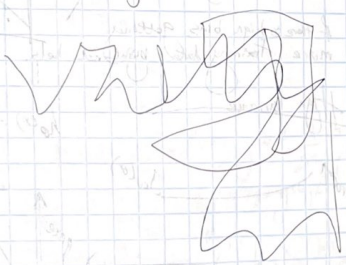
small neural network

prone to underfit, computationally cheap

large neural network

use regularization (λ) to address overfitting

computationally expensive



ex 5 workspace

(12,2) (12,1) (2,1) (1,1)

linearRegCostFunction (X, y, theta, lambda)

Conv (m,1) X

[1;1]

$$J(\theta) = \frac{1}{2m}\left(\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right) + \frac{\lambda}{2m}\left(\sum_{j=1}^{n}\theta_j^2\right)$$

theta = 2×1

copy theta

12×2 · 2×1 − 12×1

$(\frac{1}{2m}$ sum $(((X \cdot theta) - y).^2)) + (\frac{lambda}{2\cdot m}$ (sum

gradient $\begin{bmatrix} (1) \\ (2) \\ \vdots \\ (n) \end{bmatrix}$

$$\rightarrow grad(1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x_j^{(i)}$$

$$\rightarrow grad(i) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\,x_j^{(i)} + \frac{\lambda}{m}\theta_j$$

$x^{(i)}$ $\theta(i)$

12×2 2×1 12×1

$(X \cdot theta - y) .* X$

12×1 − 12×1

= 12×1

2-1 learning curves

task: plot learning curve, training set and cross validation set for different training set

12×2 12×1 21×2 21×1 1×1

learning curve(X, y, Xval, Yval, lambda) → error_train

error_val

[ones, X] y [ones, Xval] Yval 0

why doesn't Xval contain x?

12×1

X Xtest Xval

y ytest yval

12×1 21×1 21×1

mod train cross validation

$$J_{train}(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2\right]$$

use train Lmarker

training set used to train parameters

cross validation set used to fine tune and pick equation

test to test final equation

Xval	21×2	X(1:1,:)	1×2	1×3
Yval	21 1	y(1:1)	1×1	1×1
X	12 2	X(1:2,:)	2×2	2×3
y	12 1	y(1:2)	2×1	2×1

X: 3×1 3[ ]

X_poly 3×8 3[8]

10 3

10 1

10 3

10 (