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

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

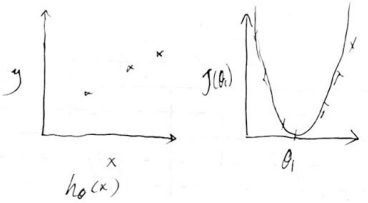
$$\text{cost function} = J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$$

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

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

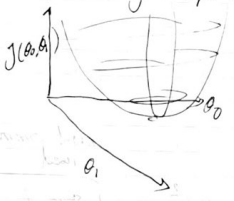
cost function $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

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 {

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

}

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

Partial Derivative Demonstration

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

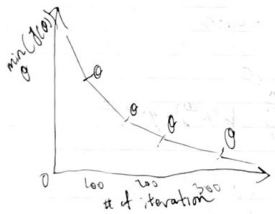
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 x_0 + \theta_1 \cdot x_1 + \dots + \theta_n \cdot x_n) - y^{(i)} \right)^2 \right]$$

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

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

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

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



brah, this shit trash.

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

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

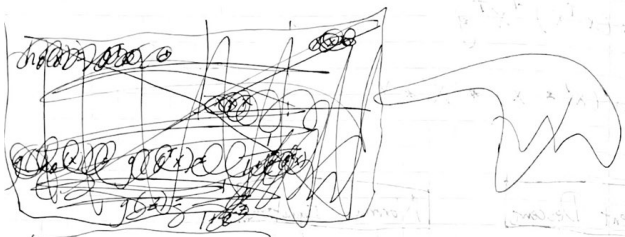
Gradient Descent

Σ, need α,
works well n large.

Normal Equation

matrix, find θ.
no good if n large.

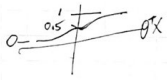
Week 3 Notes



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

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

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



example:

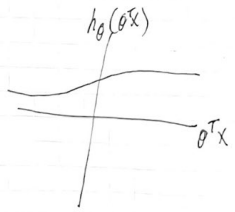
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

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

example:

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

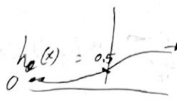
$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$ predict "$y = 1$" if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 \geq 0$



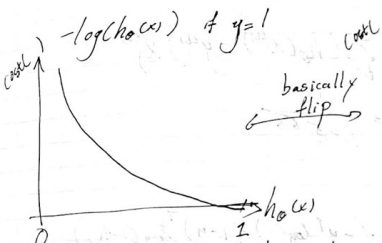
$h_\theta(x) \geq 0.5 \rightarrow y = 1$

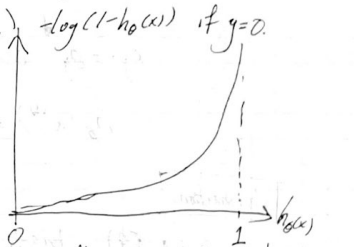
$h_\theta(x) < 0.5 \rightarrow y = 0$

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

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

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 # 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

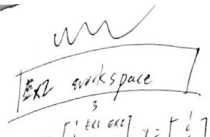
$$J(\theta) = \left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

derivative:

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

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

Ex1 workspace



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

100×3

Ex2.m

$m \times n$

100×2

initial theta: zero (3, 1)

theta (3, 1)

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

$h = sigmoid\begin{bmatrix} \theta_0 + Ex1\cdot\theta_1 + Ex2\cdot\theta_2 \\ \vdots \\ \end{bmatrix} \rightarrow h = \begin{bmatrix} \frac{1}{1+e^{-(\theta_0 + ex_1\cdot\theta_1 + ex_2\cdot\theta_2)}} \\ \vdots \end{bmatrix}$

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

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

Regularization

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

$h = sigmoid(X * theta)$

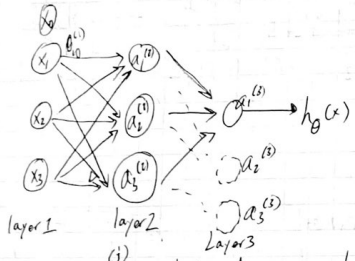
octave code

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

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

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

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 $j+1$.

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

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

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

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

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

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

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

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

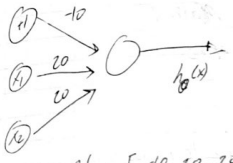
$$z^{(3)} = \Theta^{(2)} \cdot a^{(2)}$$

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

layer $j=2$ node k, variable z. $\quad X = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \dots \\ z_n^{(j)} \end{bmatrix}$

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

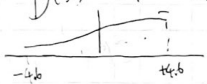
$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$



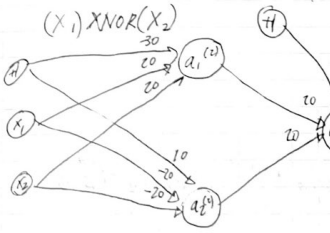
$\Theta^1 = [-10, 20, 20]$

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

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 → X matrix; whether it is → y; 10; 0.1

5000 × 400 → each image 20×20 Grayscale pixel; 5000 × 1

m = 5000

n = 400

all_theta = zeros 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 classifier for each number different

lrCostFunction (theta, X, y, lambda)

@t t 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)

401 0s for each pixel intensity

all_theta (c, :) = ... this means continue.

fmincg (

10 × 401.

return the θ parameters for each of the 10 classes

for peopee = 1: size(X, 1)

X(PeoPee, :)

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

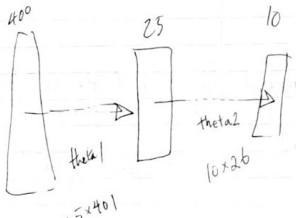
theta1 = 25 x 401

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()]

5000 [401 x 1] → X (bigpenis, :)'

X

theta 1 * 25 x 401.

sig(25 [1]) ⇒ 25 [1]

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

↓

theta 1

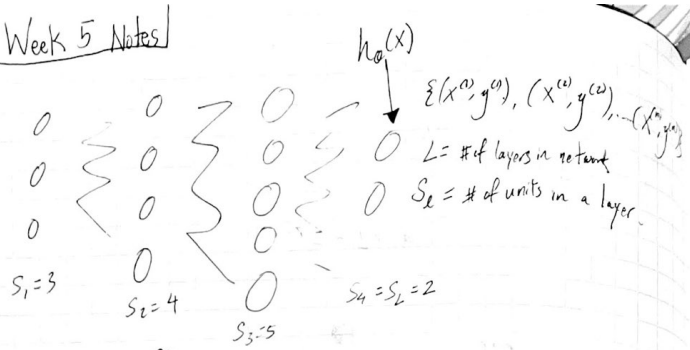
~~TEXT~~ 26 [1] 10 [26]

bigger penis.

theta 2

(Theta2 * 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 $\quad K_k$

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

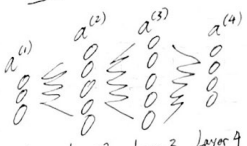
$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta(x^{(i)})\right)_k + (1-y_k^{(i)}) \log\left(1-h_\Theta(x^{(i)})\right)_k\right]$$
$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{S_l}\sum_{j=1}^{S_{l+1}} \left(\Theta_{ji}^{(l)}\right)^2$$

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 → $\delta^{(4)} = a^{(4)} - y$

$\frac{\partial}{\partial z}\left(h_\Theta(x)\right)_j$ ← derivative

$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$ $\quad$ $z^{(3)} = \begin{bmatrix} z_1^{(3)} \\ z_2^{(3)} \\ \vdots \\ z_n^{(3)} \end{bmatrix}$ → $\Theta_{10}^{(1)} a_0^{(2)} + \Theta_{11}^{(1)} a_1^{(2)} \ldots$

$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$ $\quad$ $g'(z^{(3)}) = a^{(3)} .* (1 - a^{(3)})$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$ $\quad$ 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

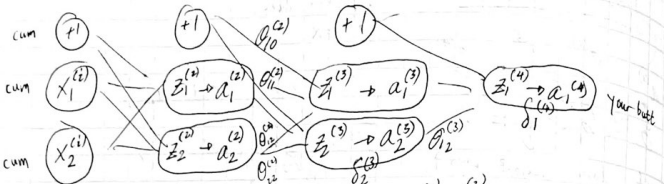
$D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ $\quad$ if $j \neq 0$

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

complex math proof shows that: $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

Forward Propagation

a_i = 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 $\text{cost}(i) = y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log h_\theta(x^{(i)})$

$\delta_1^{(2)} = y^{(i)} - a_1^{(4)}$ ← whatever output you choose

$\delta_2^{(3)} = \theta_{12}^{(3)} \cdot \delta_1^{(4)}$

$\delta_2^{(2)} = \theta_{12}^{(2)} \delta_1^{(3)} + \theta_{22}^{(2)} \cdot \delta_2^{(3)}$

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

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

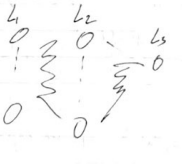
(θ) parameter (D) gradients

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

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

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

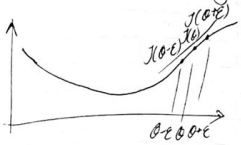
$D^{(1)} \in \mathbb{R}^{10 \times 11} \quad D^{(2)} \in \mathbb{R}^{10 \times 11} \quad D^{(3)} \in \mathbb{R}^{1 \times 11}$




```
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



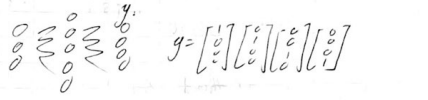
$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon}$

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

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

gradApprox ≈ DVec

Putting it together



ex 4 workspace

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

10285×1 | 400 1× | 25 1×1 | 1×1 10 | 5000×400 | 5000×1 | 1×1

400 25 10

theta1 theta2

~~401×25~~ ~~26×10~~

25×401 10×26

10285

Theta 1 25 × 401

Theta 2 10×26

m = 5000

X = 5000×401

go through X.

1 to 5000

X(1,:) → 1×401 → 1×25

X(1,:)×theta1 = → first → 1×26

1×401 × 401×25

first × Theta 2 = 1×10.

1×26 26×10

yi(1,:) = 1×10

yi(:,1) = 10×1

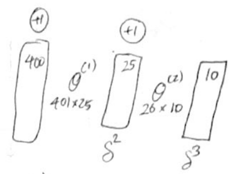
log(second) × −(yi(:,)) — 1×10, 10×1

log ./

log(first

X = 5000 × 401

y = 10×10 identity matrix

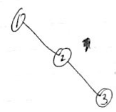


max (null , 2)

1

max(null , null)

1 1



but not on deliverable

$500

create practice for XYZ.

2 3

5000 401

$-y_k^{(i)} \cdot \log(h_\Theta(x))$

25[401]

10[26]

theta1 = 25×401

theta2 = 10×26

X(i, :) = 1×401

25×401 ×

1×401 =

X = 2×3

Y = 1×3

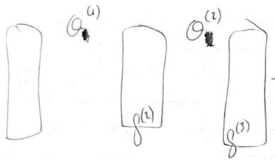
1×10 hΘ×

Y(1,:) · 10×1 = 1×1

25

10×26 × 26×1

10×26 × 26×1 = 10×1



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

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

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

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

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

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

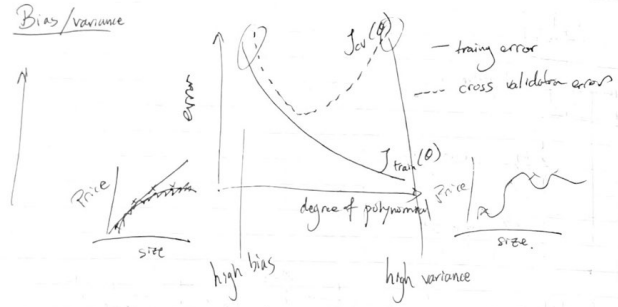
Cross Validation Error

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

Test Error

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

Bias/variance

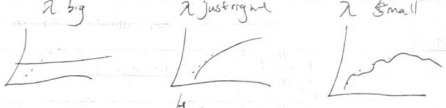


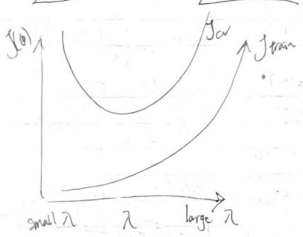
Bias (underfit)
$J_{train}(\theta)$ high
$J_{cv}(\theta) \approx J_{train}(\theta)$

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

Linear Regression with Regularization

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



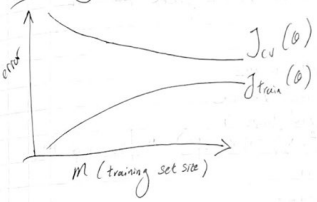


$$J(\theta) = \frac{1}{2m}\cdot\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

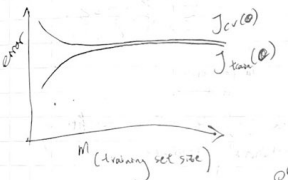
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

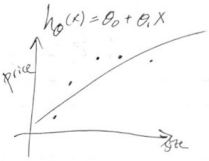
$$J_{cv}(\theta) = \frac{1}{2\cdot m_{cv}}\cdot\sum_{i=1}^{M_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

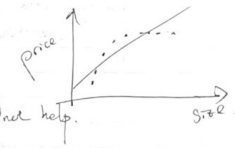
Learning Curves



High Bias

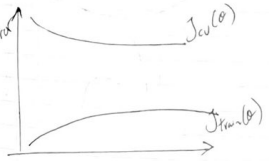






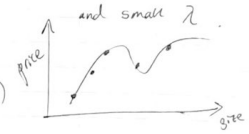
If there is high bias, getting more training data will not help.

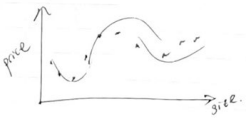
High variance



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.

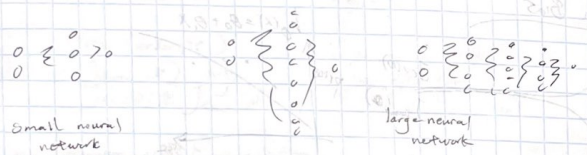




Debugging learning algorithm

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



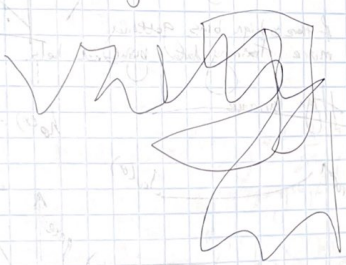
small neural network

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

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

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

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

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

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

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

Yval 21 1

X 12 2

y 12 1

X(1:1,:) 1×2 1×3

y(1:1) 1×1 1×1

X(1:2,:) 2×2 2×3

y(1:2) 2×1 2×1

X: 3×1 3[ ]

X_poly 3×8 3[8]

10 3

10 1

10 3

10 (

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive + false positive}}$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive + false negative}}$$

high precision and high recall = good algorithm.

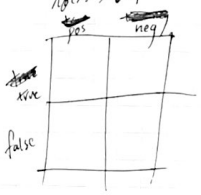
means true positive is high.

evalution for classfication with skew classes

how to choose threshold ? predict 1 if $h_\theta(x) \geq$ threshold

F_1 Score: $2 \cdot \frac{PR}{P+R}$

$h_\theta(x) \geq 0.1$ even low chance, still say cancer.



$$(a + \frac{1}{c}b$$

$$\frac{1}{m}a + \frac{1}{cm}b$$

$\frac{1}{\lambda} \cdot C = \frac{1}{\lambda}$

λ

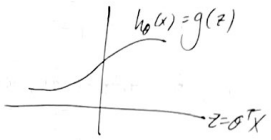
λ

$\lim_{\lambda \to 0}$

Week 7 Notes

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

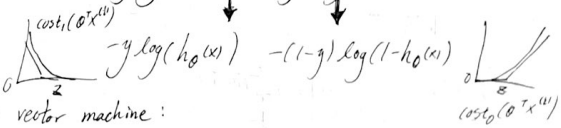
θ k x matrices



if $y=1$, we want $h_\theta(x) \approx 1$ $\theta^T x \geq 0$

if $y=0$, we want $h_\theta(x) \approx 0$, $\theta^T x \leq 0$

cost function: $-[y \log(h_\theta(x)) + (1-y) \log(1-h_\theta(x))]$



Support vector machine:

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

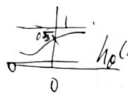
$$\min_\theta \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^T x^{(i)}) + \frac{\lambda}{2m}\sum_{j=0}^{n}\theta_j^2$$

$$\min_\theta C \cdot \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\,\text{cost}_0(\theta^T x^{(i)}) + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

find the θ that will give you the min

hypothesis: $\begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$

$\theta^T x$ is your function



$h_\theta(x) = \frac{1}{1+e^{-(\theta^T x)}}$ makes it between 0 or 1.

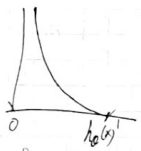
$h_\theta(x) \geq 0.5$ then predicts $y=1$

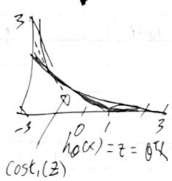
$h_\theta(x) < 0.5$ then predicts $y=0$

$-y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$ cost function for prediction.

↑ the log is ln.

$-\log(h_\theta(x))$ $\qquad$ $-\log\left(\frac{1}{1+e^{-(\theta^T x)}}\right)$

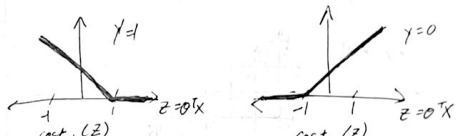




for $y=1$,
if $h_\theta(x)$ predicts 1,
receives 0 penalty

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}\text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)})\text{cost}_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

find the θ for θ^T that will give the lowest sum cost function with regularization.

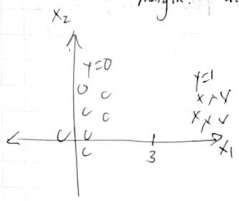


if $y=1$, we want $\theta^T x \geq 1$ (more flexible, $\theta^T x \geq 0$ no need for only)

if $y=0$, we want $\theta^T x \leq -1$ $\qquad$ $\theta^T x \leq 0$

Set C to very large, then if we make prediction mistake, then function will be penalized greatly.

aka.: Large margin classifier



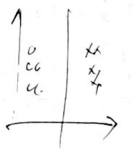
$\theta_3 + \theta_2 x_2 + \theta_1 x_1 \geq 0$

$\theta_3 = -3$

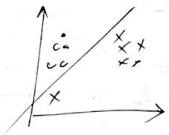
$\theta_2 = 0$

$\theta_1 = 1$

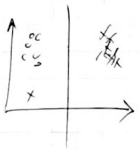
large margin classifier



C very large
no outlier



C very large
outlier

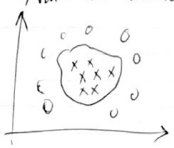


C very small
outlier

$u^T \cdot v = p \cdot \|u\| = u_1 v_1 + u_2 v_2$

p = projection of v onto u

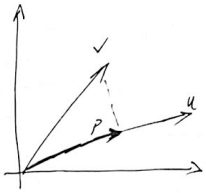
Non linear decision boundary



$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \cdots$

$x_1 = f_1 \quad x_2 = f_2 \quad x_1 x_2 = f_3 \quad \cdots$

is there other/better choices for features $f_1, f_2, f_3 \ldots$?



projection v onto u

$u^T v = p \cdot \|u\| = u_1 \cdot v_1 + u_2 \cdot v_2$

Regularization cost function

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

let say features : $x_1, x_2 \ldots x_{100}$
Parameters : $\theta_0, \theta_1 \ldots \theta_{100}$

Shrinks all parameters θ_1 to θ_{100} skip θ_0

keeping $(h_\theta(x))$ the hypothesis simple to prevent overfitting.

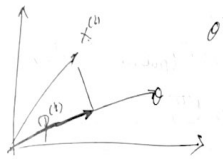
$$\min_\theta \frac{1}{2}\sum_{j=1}^{n} \theta_j^2$$

let say $n=2$

$$= \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2}\|\theta\|^2$$

s.t $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$
$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

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



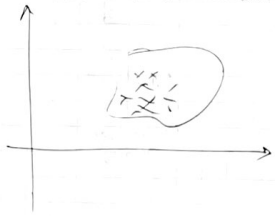
$\theta^T x^{(i)} = p^{(i)} \cdot \|\theta\| = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

$$\min_{\theta} \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}\|\theta\|^2$$

s.t $p^{(i)}\cdot\|\theta\| \geq 1$ if $y^{(i)}=1$

$p^{(i)}\|\theta\| \leq -1$ if $y^{(i)}=1$

Non-linear Decision Boundary



Predict $y=1$ if

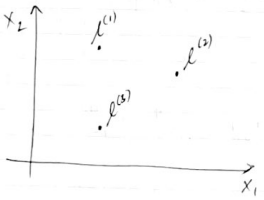
$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$.

$f_1 = x_1,\ f_2 = x_2,\ f_3 = x_1 x_2$

$f_4 = x_1^2,\ f_5 = x_2^2$

better / different choice for $f_1, f_2, f_3, \ldots$?

Kernel



$f_1 = \text{similarity}(x, l^{(1)}) = e^{\left(\frac{-\|x-l^{(1)}\|^2}{2\sigma^2}\right)}$

$f_2 = \text{similarity}(x, l^{(2)}) = e^{\left(\frac{-\|x-l^{(2)}\|^2}{2\sigma^2}\right)}$

↑

Kernel (Gaussian Kernels)

$k(x, l^{(1)})$

$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(\frac{-\|x-l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$

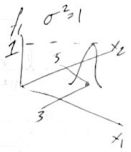
if $x \approx l^{(1)}$

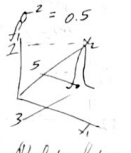
$f_1 \approx \exp\left(\frac{-0^2}{2\sigma^2}\right) \approx 1$

if x is far from $l^{(1)}$

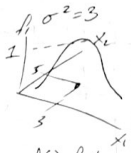
$f_1 \approx \exp\left(\frac{-(\text{large number})^2}{2\sigma^2}\right) \approx 0$.

$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$ $\quad f_1 = e^{\left(\frac{-\|x-l^{(1)}\|^2}{2\sigma^2}\right)}$

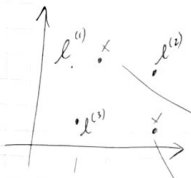




$f^{(1)}$ falls faster when moving away from $l^{(1)}$



$f^{(1)}$ falls slower when moving away from $l^{(1)}$



$x \rightarrow f_1 \approx 1\ f_2 \approx 0\ f_3 \approx 0$

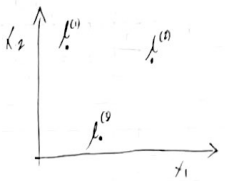
let say $\theta_0 = -0.5\ \theta_1 = 1\ \theta_2 = 1\ \theta_3 = 0$

$f_1 \approx 1\ f_2 \approx 0\ f_3 \approx 0$

$-0.5 + 1\times 1 + 1\times 0 + 1\times 0 = 0.5 \geq 0 \rightarrow y = 1$

$f_1 \approx 0\ f_2 \approx 0\ f_3 \approx 0$

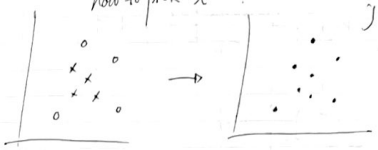
$-0.5 + 1\times 0 + 1\times 0 + 1\times 0 = -0.5 \leq 0 \rightarrow y = 0$



Given x:

$f_i = \text{similarity}(x, l^{(i)}) = e^{\left(\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right)}$

how to pick $l^{(i)}$?



just put them in the same place.

Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$,

choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$

$$f = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_m \end{bmatrix} \quad f_0 = 1$$

$x^{(i)} \rightarrow$

$f_1^{(i)} = \text{sim}(x^{(i)}, l^{(1)})$

$f_2^{(i)} = \text{sim}(x^{(i)}, l^{(2)})$

$\vdots$

$f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$

$x^{(i)}$ & $l^{(i)}$ will be the same

$f_i^{(i)} = \text{sim}(x^{(i)}, l^{(i)}) = e^{\left(\frac{-0}{2\sigma^2}\right)} = 1$

$$f^{(i)} = \begin{bmatrix} f_0^{(i)} = 1 \\ f_1^{(i)} \\ \vdots \\ f_m^{(i)} \end{bmatrix}$$

Training:

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

$\theta^T \theta$

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$

Predict "$y = 1$" if $\theta^T f \geq 0$ $\quad \theta \in \mathbb{R}^{m+1}$

$\theta_0 f_0 + \theta_1 f_1 + \ldots + \theta_m f_m$

$C = \left(\frac{1}{\lambda}\right)$

large C: low bias, high variance (small λ)

lower C: high bias, low variance (big λ)

low bias
low variance

If SVM overfits, decrease C, increase σ^2

Linear kernel vs Gaussian kernel

→ function $f = \text{kernel}(x_1, x_2)$

not all similarities are valid kernels, need to satisfy Mercer's theorem.

Logistic Regression vs SVMs

n = # of features $\quad m$ = # of training sets

if n large relative to m, use logistic regression or svm without kernel. (linear)

if n small, m intermediate ($1 - 1000$) & ($10 - 10{,}000$), use svm with gaussian kernel.

if n is small, m is large, ($1 - 1000$), ($m = 50{,}000$), use logistic / svm without kernel.

- Most cases, neural network better, but slower.

Week 8 Notes

K means algorithm → find lowest distance for cluster centroid

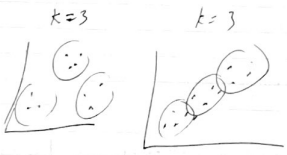
Input
- K (# of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}^n$ drop $x_0 = 1$

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat

{ assign centroid
- for $i = 1$ to m
  - $c^{(i)}$:= $i = 1$ to k $\min_k \|x^{(i)} - \mu_k\|$ (closest to $x^{(i)}$)
  - $c^{(3)} = 5$ ($x^{(3)}$ closest μ_5)

{ move centroid
- for $k = 1$ to K
  - μ_k := average of $x^{(i)}$ assigned to μ_k
  - example: $x^{(1)} = 2$, $x^{(5)} = 2$, $x^{(6)} = 2$, $x^{(10)} = 2$; $\mu_2 = \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(6)} + x^{(10)}]$



$c^{(i)}$ = index of cluster assigned to $x^{(i)}$

μ_k = cluster centroid ← it is a location

$\mu_{c^{(i)}}$ = cluster assigned to $x^{(i)}$

$x^{(i)} \to 5$ $c^{(i)} = 5$ $\mu_{c^{(i)}} = \mu_5$

Optimization Objective:

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

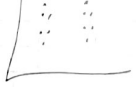
$$\min_{c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$$

How to randomly initialize?

Local optima issues

pick random $x^{(i)}$'s, $k < m$

Choosing k.



2 or 4?

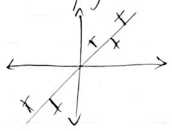
t-shirt sizes

Dimension Compression

3D → 2D graph

PCA Principal Component Analysis

Minimize projection distance



Not linear regression →

$x^{(i)} \in \mathbb{R}^2 \rightarrow x^{(i)} \in \mathbb{R}^1$

Reduce data from n to k dimensions

$$\text{Sigma} = \frac{1}{m}\sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T \quad (n\times 1)(1\times n)$$

$[u, s, v] = \text{svd}(\text{Sigma})$ — Sigma is $n \times n$

u is also $n \times n$.

$$U = \left[\underbrace{u^{(1)}\ u^{(2)}\ u^{(3)} \cdots u^{(k)}}_{k}\ \cdots\ u^{(m)}\right] \in \mathbb{R}^{n\times n}$$

$$z = \left[u^{(1)}\ u^{(2)} \cdots u^{(k)}\right]^T \cdot x \rightarrow$$

$n \times k$

U_{reduce}

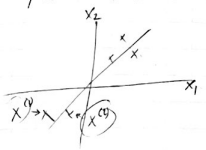
$$X = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$$

sigma = $(\frac{1}{m})$ * X' * X;

$U_{reduce} = U(:, 1:k);$

$z = U_{reduce}' * X;$

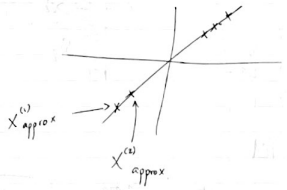
Compression Reconstruction



$z = U_{reduce}^T x$

$x_{approx} = U_{reduce} \cdot z$

$\mathbb{R}^n$; $n \times k$; $k \times 1$; $n \times 1$



$x_{approx}^{(i)}$; $x_{approx}^{(1)}$

K : # of principal components

average squared projection error : $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$

total variation in the data : $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01$$

→ 99% of variance is retained

choose k → and repeat until

Try PCA $k=1$ if fail, $k=2$, if fail, $k=3$ ---

Compute Ureduce, $z^{(1)}, \dots, z^{(m)}, x^{(1)}_{approx}, \dots, x^{(m)}_{approx}$

check if:

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \; ?$$

not efficient

$[U, S, V] = svd(Sigma)$

$$S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & \ddots & \\ & & & S_{nn} \end{bmatrix} \quad n \times n$$

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01$$

Application of PCA

$x^{(1)}, x^{(2)}, \dots, x^{(m)} \in \mathbb{R}^{10,000}$ (100 × 100)

↓ PCA

$z^{(1)}, z^{(2)}, \dots, z^{(m)} \in \mathbb{R}^{1000}$ $\quad h_\theta(z) = \frac{1}{1+e^{-\theta^T z}}$

use PCA for compression and visualization.

But don't use PCA to prevent overfitting.

When designing ML system, try implementing it first before applying PCA. Only use PCA when you have a really good reason.

ex 7 workspace

idx = findClosestCentroids(X , initial_centroids)

(300, 1)

range 1 : K

sizeof(centroids, 1)

300 [2]

3 [2]

initial_centroids(j , :)

$c^{(i)} \rightarrow \in \mathbb{R}^n$

centroids = computeCentroids(X, idx, K)

3 [2]

300 [2]

300 [$c^{(1)}$; $c^{(2)}$; ⋮ ; $c^{(300)}$] 3

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

$2^8 = 256$

24 bit color representation = three 8 bits
0-255 red
0-255 green
0-255 blue

thousands of combination → compress → 16 combo of color

original image 128 [128] $128^2 = 16384$

$\Sigma = \frac{1}{m} X^T X$ covariance matrix

sigma

n by n matrix

Z U

50×1 2×2

$X_{recover}$ = 50×2

Week 9 Notes

Anomaly Detection

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

Model $p(x)$

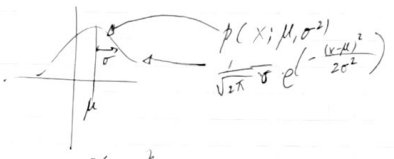
if $p(x) < \varepsilon$ flag for anomaly.

if $p(x) \geq \varepsilon$ flag ok

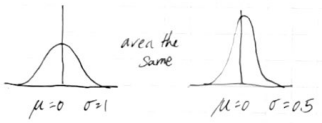
Gaussian (Normal) Distribution

Gaussian Distribution is the same shit as a normal distribution

$X \sim N(\mu, \sigma^2)$

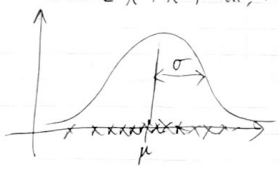


example



Parameter Estimation

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$ $x^{(i)} \sim N(\mu, \sigma^2)$



$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Density Estimation

training set: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$x \in \mathbb{R}^n$

X_i^m ← raw, i ← feature

$$p(x) = p(x_1; \mu_1, \sigma_1^2) \cdot p(x_2; \mu_2, \sigma_2^2) \cdot \dots \cdot p(x_n; \mu_n, \sigma_n^2)$$

$x_1 \sim N(\mu_1, \sigma_1^2)$ $x_2 \sim N(\mu_2, \sigma_2^2)$

$$= \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

Anomaly Detection Algorithm

(1) choose features x_i ← different features

(2) Fit parameter $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \longrightarrow \mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m}\sum_{i=1}^{n} x^{(i)}$$

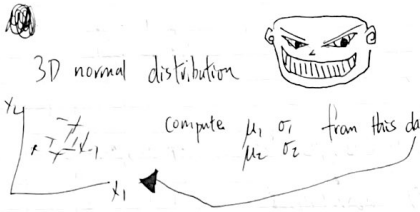
$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

(3) Given new example x, compute $p(x)$:

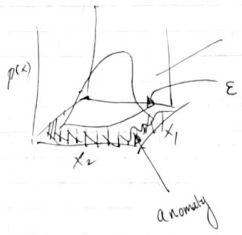
$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\cdot\sigma_j} e^{\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)}$$

Anomaly if $p(x) < \varepsilon$

3D normal distribution



$p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2) = p(x)$



Aircraft engine

Developing and evaluating anomaly detection systems.

10,000 good engines
20 flawed engines

training set : 6000 good engine.
CV set : 2000 good engine ($y=0$), 10 anomalous ($y=1$)
Test set : 2000 good engine ($y=0$), 10 anomalous ($y=1$)

repeated but in practice

	6000	
	4000	10
	4000	10

Algorithm Evaluation

Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$
on cross validation / test example x,
predict:

$y = 1$ if $p(x) < \varepsilon$ (anomaly)
$y = 0$ if $p(x) \geq \varepsilon$ (normal)

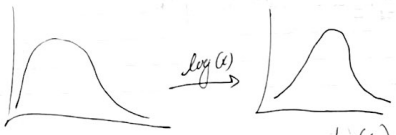
Classification no good, because dataset is skewed because $y=0$ is more common.

Anomaly Detection vs ~~supervised learning~~

(when to use which?)

- Small number (0-20) of positive example ($y=1$)
  Large number of negative number.
- large number of positive and negative number.

choosing what features to use?



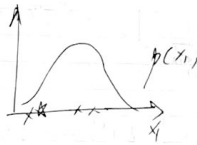
$x_1 \leftarrow \log(x_1)$

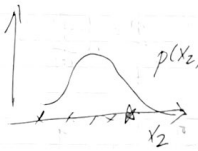
$x_2 \leftarrow \log(x_2 + c)$

coming up with more/new features

Multivariate Gaussian Distribution

example: monitoring machines in a data center.





$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$

$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

$\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} e^{\left[-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right]}$$

Multivariate Gaussian Distribution anomaly detection

Parameters $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right]}$$

Parameter fitting:
Given training set $\{x^{(1)}, \dots, x^{(m)}\}$

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \qquad \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

$$\begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{bmatrix}$$

use multivariate if $m > n$, $m \geq 10n$.

Problem Formulation

Movie	A	B	C	D	X_1 romance	X_2 action

Predicting movie ratings.

n_u = # of users $\quad n_m$ = # of movies

$r(i,j) = 1$ if user j has rated movie i

$y(i,j)$ rating user j gave to movie i.

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vectors for movie i.

rating prediction = $(\theta^{(j)})^T (x^{(i)})$

$m^{(j)}$ = # of movies rated by user j

To learn $\theta^{(j)}$:

$$\min_{\theta^{(j)}} \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} \left[\left(\theta^{(j)} \right)^T (x^{(i)}) - y^{(i,j)} \right]^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

only for single user j.

for all users: to learn $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$

$$\min_{\theta^{(1)}, \dots, \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left[(\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right]^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Gradient Descent:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left[(\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right] \cdot x_k^{(i)} \qquad k = 0.$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} (\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \qquad k \neq 0$$

$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots \theta^{(n_u)})$

Collaborative Filtering

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$ to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$ to learn $x^{(1)}, \dots, x^{(n_m)}$:

$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left[(\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right]^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

$x_0 = 1$

Movie	alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 romance	x_2 action
Love at last	5	5	0	0	? 1	? 0
Romance forever	5	?	?	0	?	?
Cute puppies of love	?	4	0	?	?	?
Nonstop car chases	0	0	5	4	?	?
Swords vs karate	0	0	5	?	?	?

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \quad \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$$

$$(\theta^{(1)})^T \cdot x^{(1)} \approx 5 \qquad (\theta^{(2)})^T x^{(1)} \approx 5$$

$$\begin{bmatrix} 0 & 5 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = 0 + 5 + 0 = 5$$

$$\min_{\substack{x^{(1)}, \dots, x^{(n_m)} \\ \theta^{(1)}, \dots, \theta^{(n_u)}}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left[(\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right]^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$j = 1, 2, \dots, n_u, \quad i = 1, \dots, n_m$

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right) \quad \leftarrow \frac{\partial}{\partial x_k^{(i)}}$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

Vectorization: Low Rank Matrix Factorization

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted Ratings:

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \cdots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & & & \\ \vdots & & & \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

$\theta \rightarrow$ what genre they like $\quad x_1 \rightarrow$ movie genre type

$$X \cdot \theta^T \rightarrow X = \begin{bmatrix} -(x^{(1)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix} \quad \theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ \vdots \\ -(\theta^{(n_u)})^T- \end{bmatrix}$$

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$.
(movie → genre)

movie i related to movie j

if $\|x^{(i)} - x^{(j)}\|$ is small, movie i & j are similar.

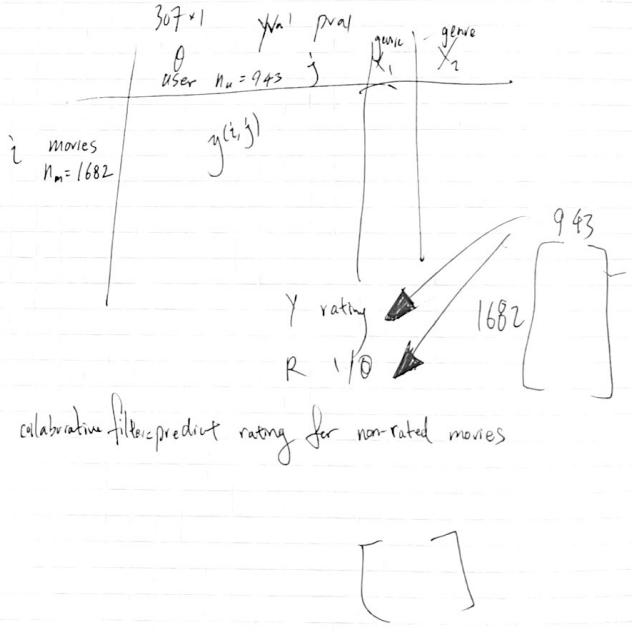
$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \rightarrow \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j, on movie i, predict

$$(\theta^{(j)})^T(x^{(i)}) + \mu_i$$

Ex 8 Work Space

$X = 307 \times 2$ $\quad \mu = 2 \times 1$ $\quad \sigma^2 = 2 \times 1$



collaborative filter: predict rating for non-rated movies

$[\quad]$

$X = 1682 \times 100$ $\quad \Theta = 943 \times 100$

predicts rating for movie i by user j.

$y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$

$X = 5 \times 3$ $\quad$ theta $= 4 \times 3$

$Y = 5 \times 4$ $\quad$ X_grad 5×3

$R = 5 \times 4$ $\quad$ theta_grad 4×3

5×4 $\quad$ theta 4×3 $\quad j \to n_u$: 4

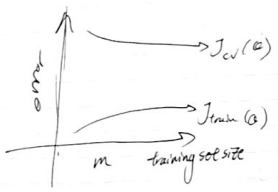
5×4

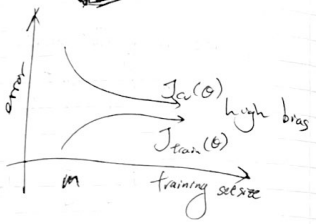
4×5 $\quad 5 \times 3$

Week 10 Notes

learning with large datasets

$m = 100,000,000$? $m = 1,000$?





Stochastic Gradient Descent

batch ← it looks at all the examples

Issue with (normal) gradient descent is if m is large, then computation is expensive. 300,000,000

Batch Gradient Descent

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

Repeat {

$$\theta_j := \theta_j - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}_{\frac{\partial}{\partial \theta_j} J_{train}(\theta)}$$

}

Stochastic Gradient Descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

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

Batch Gradient Descent → m examples each iteration

Stochastic Gradient Descent → 1 example each iteration

Mini Batch Gradient Descent → b examples each iteration

Stochastic Gradient Descent convergence & learning rate α