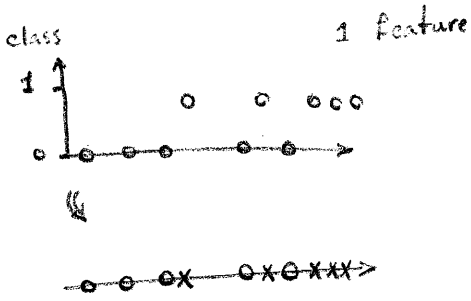
Machine Learning: (Andrew Ng - Stanford)

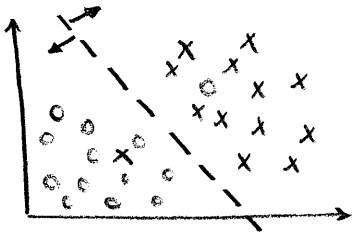
* علل اهمیت ML
  - توانایی‌های جدید برای کامپیوتر
  - آنقدر بزرگ شده است که از AI خارج شده است.

* شاخه‌های کاربردی
  - Database Mining ← Web ① ② داده‌های بزرگ حاصل از اتوماسیون
  - برنامه‌هایی که بهبود می‌توان نوشت ← مهارت، دانش خبره، دانش محتی در کاربرد
    Self-customizing Programs
  - درک یادگیری در مغز

* یادگیری ماشین چیست؟
  - آرتور ساموئل: زمینه مطالعاتی که به کامپیوتر این قابلیت را می‌دهد که کاری را انجام دهد که صراحتاً برای آن برنامه‌ریزی نشده است.
  - تام میچل: یادگیری به صورت یک مسئله خوش تعریف: بالا بردن Performance P براساس تجربه E برای یک وظیفه خاص T

* یادگیری با ناظر (Supervised learning)
  - به مجموعه‌ای از داده‌ها اعمال می‌شود که جواب‌های درست آن موجود است.
  - Regression: پیش‌بینی خروجی‌های دارای مقادیر پیوسته
  - Classification: پیش‌بینی کلاس داده‌ها با خروجی گسسته ☑





* یادگیری بدون ناظر (Unsupervised learning)
  - در مجموعه داده‌ها به دنبال ساختمان می‌گردیم.

social net work anal., organizing large computer clusters, Genomics, news.google.com ☑
astronomical data analysis, market segmentation

  - ساختار داده‌ها را برای دسته‌بندی آنها استفاده می‌کنیم: clustering ← دسته‌ها مهم است
  - ساختار داده‌ها را برای تفکیک آنها استفاده می‌کنیم: PCA, ICA ← داده‌های ورودی اصلی مهم است
    cocktail party problem ☑

Linear Regression *

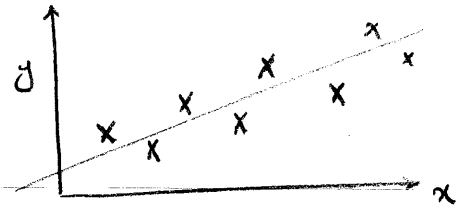
• تطبیق یک خط به داده‌ها با داشتن حداقل داده برای پیش‌بینی خروجی برای مقدار پیوسته

m = # of inputs
x = input vars / features
y = output vars / features

(x, y) : training example
$(x^{(i)}, y^{(i)})$: i^{th} training example

Training Set ⟹ Learning Alg. ⟹ Hypothesis : mapping from x to y

1D feature : $h_\theta(x) = \theta_0 + \theta_1 x$ — Univariate



Find me the θ_0 and θ_1 so that the average of error of my real data to fitted data is minimized.

θ : parameters → goal : find θ's so $h_\theta(x)$ is close to y for training examples (x, y)

$$\min_{\theta_0,\theta_1} (h_\theta(x) - y)^2 \leftarrow$$

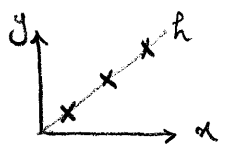
$$\min \sum_{i=1}^{m} \left(h_\theta^{(i)}(x) - y^{(i)}\right)^2$$

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

$J(\theta_0, \theta_1)$: cost function
square error function

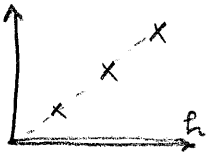
- hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x$
- params : θ_0, θ_1
- cost func: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum (h_\theta(x) - y)^2$
- goal : $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

☑ $\theta_0 = 0$



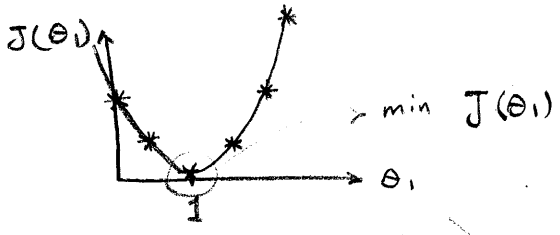
$\theta_1 = 1$
$J(\theta_1) = 0$

$\theta_1 = 0.5$
$J(\theta_1) = 0.58$

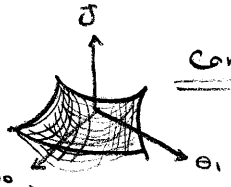


$\theta_1 = 0$
$J(\theta_1) = 2.3$

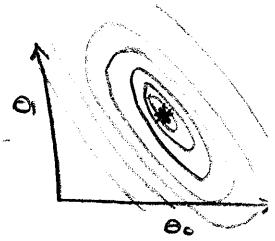
⟹

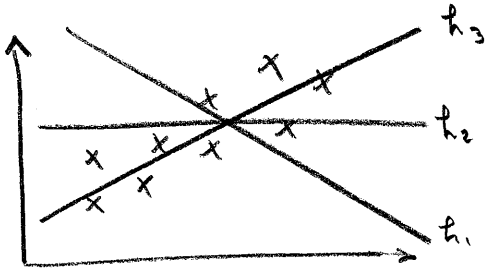


☑ J always have bowl shape (no local minima)

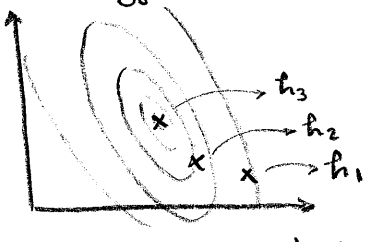


Contour





⟹



• یافتن خودکار θ_0 و θ_1 بهینه
— شروع از یک مقدار θ_0, θ_1
— اندکی θ_0, θ_1 را عوض می‌کنیم تا J کاهش پیدا کند و به کمینه برسد.

- الگوریتم از نقطه پایین میایم، به همه جهت نگاه می‌کنم و به آن سمتی می‌روم که شیب بیشتری دارد. (steepest descend)
- الگوریتم

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

- کنترل همزمانی

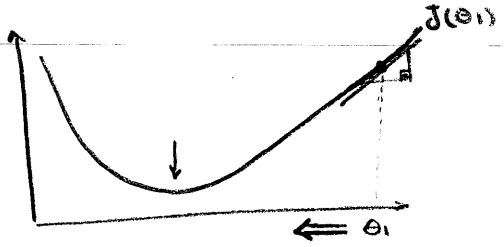
$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := temp0$
$\theta_1 := temp1$

} پارامترها باید همزمان بروزشوند.

$temp0 := \theta_0 - \alpha \dots$
$\theta_0 := temp0$
$temp1 := \theta_1 - \alpha \dots$
$\theta_1 := temp1$

} نادرست ← نتیجه دیگری دارد.

- derivative term



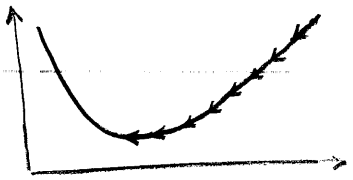
$\theta_1 := \theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\text{slope} > 0}$

$\theta_1 := \theta_1 - \alpha (+)$

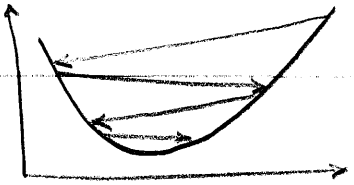
$\theta_1 \downarrow$

، بالعکس

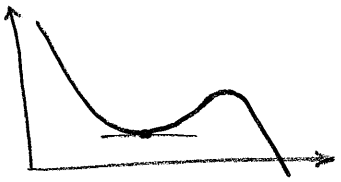
- learning rate



α کوچک ← الگوریتم کند



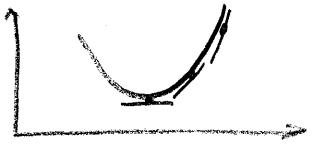
α بزرگ
احتمال عبور از min ← احتمال overshoot
احتمال عدم همگرایی



- local minima

$\theta_1 := \theta_1 - \alpha (0)$

هرچه به کمینه محلی نزدیک می‌شویم به طور خودکار اندازه قدم‌ها کوچک می‌شود (مشتق در کمینه محلی صفر است)



- اعمال GD به LR

- قدم اصلی: محاسبه مشتق

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \sum (h_\theta - y)^2 \right)$$

$$= \frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \sum (\theta_0 + \theta_1 x - y)^2 \right)$$

$$\begin{cases} j=0 : \frac{1}{m} \sum (h_\theta(x) - y) \\ j=1 : \frac{1}{m} \sum (h_\theta(x) - y) \cdot x \end{cases}$$

۳.۴ ترکیب ← جابجایی مشتق و حط همرمانی

از آنجایی که $J(\theta_0, \theta_1)$ همواره convex است الگوریتم همیشه به مینیمم محلی همگرا می‌شود.

- معمولاً $(\theta_0, \theta_1) = (0, 0)$ برای آغاز انتخاب می‌شود.

- این الگوریتم با همه نمونه‌های آموزشی سروکار دارد ($\sum_{i=1}^{m}$) بنابراین Batch نام دارد.

- Extensions
  1. حذف α
  2. تعداد بیشتر feature ها

- Multiple Features:

n = # of features

$x_j^{(i)}$ = value of feature j in i^{th} training example

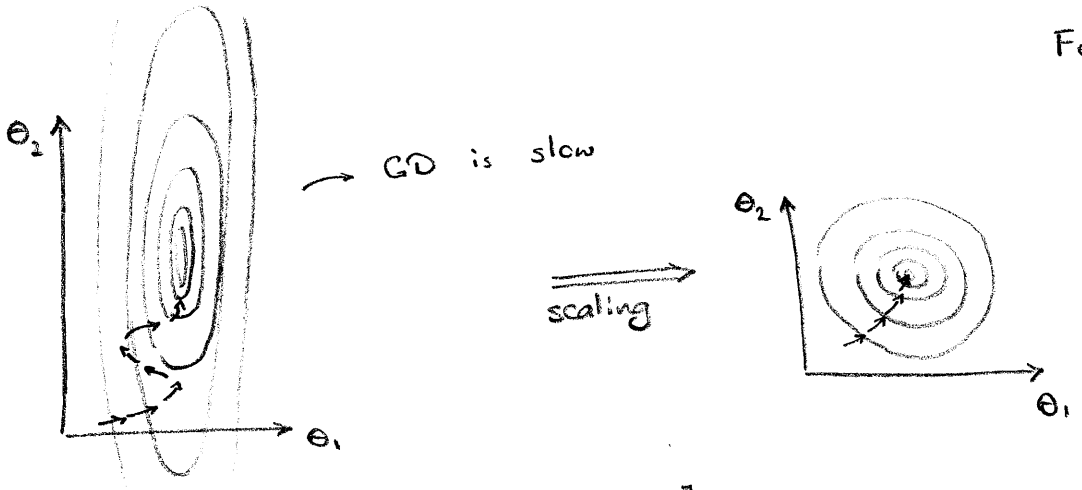
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{i=0}^{n} \theta_i x_i \qquad x_0 = 1 \rightarrow x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}_{(n+1)\times 1}$$

$$= \theta^T x \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

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

Gradient Descend: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad j = 0, \ldots, n$

- Feature Scaling



get every feature approx. into $[-1, +1]$

$0 \leq x_1 \leq 3$ ✓ ☑

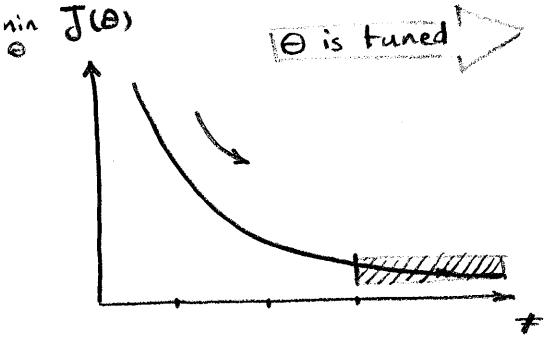
$-2 \leq x_2 \leq 0.5$ ✓

$-100 \leq x_3 \leq 100$ ×

$0.001 \leq x_4 \leq 0.01$ ×

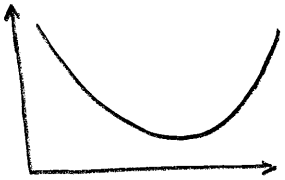
1. Divide By Maximum → x_i / x_{max}
2. Mean Normalization → $(x_i - \bar{x}) / x_{max}$
3. $(x_i - \bar{x}) / \text{range}(x)$
4. $(x_i - \bar{x}) / \sigma_x$ → standard deviation

Choosing learning Rate •

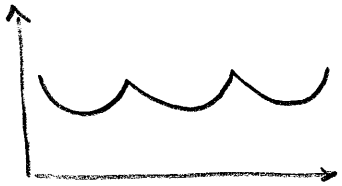


* $J(\theta)$ should decrease after every iter.
* Show the convergance
  - ↳ Automatic Convergance Test
  - ↳ ☑ کمتر از 10^{-3} تغییر $J(\theta)$ بعد از iter

decrease α



decrease α



decrease α

- α too small → slow convergance
- α too big → may not decrease on every iter
  - → may not converge / slow convergance

☑ α : 0.001 , 0.01 , 0.1 , 1 , 10 , ...

α : 0.001 , 0.003 , 0.01 , 0.03 , 0.1 , 0.3 , ...

Complex Features •

☑ x_1 = frontage, x_2 = depth $\Rightarrow z = \text{area} = x_1 \times x_2 \longrightarrow h_\theta(x) = \theta_0 + \theta_1 z_1$

Polynomial Regression •

☑ x_1 = size $\Rightarrow z_1 = x_1 , z_2 = x_1^2 , z_3 = x_1^3$

$h_\theta(z) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3$ (Cubic)

feature scaling become more important

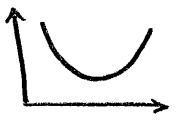
☑ x_1 = size $\Rightarrow z_1 = x_1 , z_2 = \sqrt{x_1}$

$h_\theta(z) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_2}$

Normal Equation •

Method to solve for θ analytically.

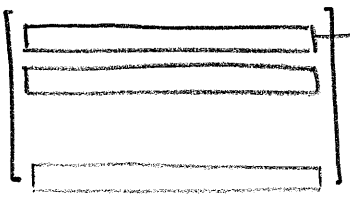
☑ $J(\theta) = a\theta^2 + b\theta + c$



$\frac{d}{d\theta} J(\theta) = 2a\theta + b = 0 \longrightarrow \theta^*$

Generally:

$J(\theta) = \frac{1}{2m} \sum (h_\theta(x) - y)^2 \longrightarrow \frac{\partial}{\partial \theta_j} J(\theta) = 0 \quad \forall j \longrightarrow \theta_0^*, \theta_1^*, \dots, \theta_n^*$

X =  → training example #1

design matrix

$m \times (n+1)$

$$X = \begin{bmatrix} \text{---}\,(x^{(1)})^T\,\text{---} \\ \text{---}\,(x^{(2)})^T\,\text{---} \\ \\ \text{---}\,(x^{(m)})^T\,\text{---} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \cdots \\ 1 & x_2^{(1)} & & \\ & & & \end{bmatrix}$$

$$\boxed{\theta = (X^T X)^{-1} X^T y} \longrightarrow \text{pinv}\,(X' * X) * X' * y$$

does not need feature scaling

Gradient Desc.	Need α	Many Iter	Works with Large n ✓
Normal Eq.	No Need α ✓	No Iteration ✓	Slow with Large n ↳ $(X^T X)^{-1} \rightarrow O(n^3)$

☑ n = 100 → NE ✓
= 1000 → NE ✓
= 10000 → GD ✓
= 10^6 → GD ✓✓

Normal Eq. & Non-invertability.

$X^T X$ → non-invertible / singular / degenerate

Causes:

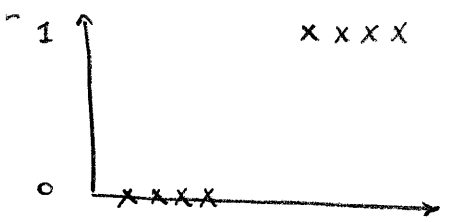
- Redundant features → linearly dependant → check features first
  - ☑ x_1 = size in $ft.^2$, x_2 = size in m^2 → $x_1 = (3.28)^2 x_2$
- Too many features
  - ☑ $m \leq n$
  - → delete some features
  - → use regularization

Classification *

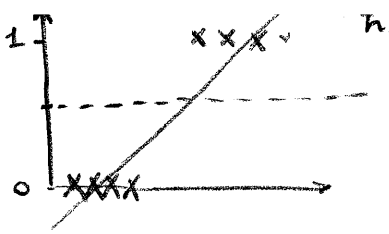
Two Class .

$y \in \{0, 1\}$ 0: Negative Class → Absence
1: Positive Class → Presence

☑ 0: benign tumor , 1: malignant



use method we know

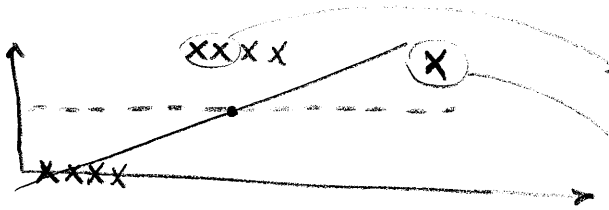


threshold $h_\theta(x) \geq 0.5$

$\downarrow$

1

☑ bad thing



misclassify } reason 1

data sample caused it }

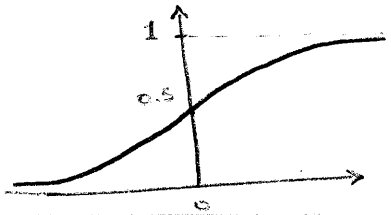
$h_\theta(x)$ can be >1 or <0 } reason 2

برای حل این مشکل می خواهیم کاری کنیم که $0 \leq h_\theta(x) \leq 1$ باشد. پس به سراغ Logistic Regression می رویم.

Logistic Regression

$h_\theta(x) = \theta^T x \xrightarrow{0 \leq h \leq 1} h_\theta(x) = g(\theta^T x)$

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



sigmoid / logistic function

$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ → estimated probability that $y=1$ on input x

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

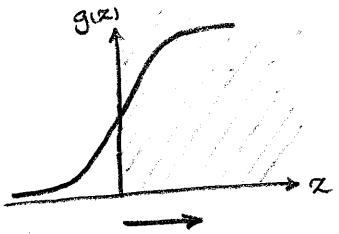
$p(y=0 \mid x; \theta) = 1 - h_\theta(x)$

Decision Boundary

threshold $= 0.5 \longrightarrow h_\theta(x) \geq 0.5 \Rightarrow y=1$

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

$\theta^T x \geq 0$



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

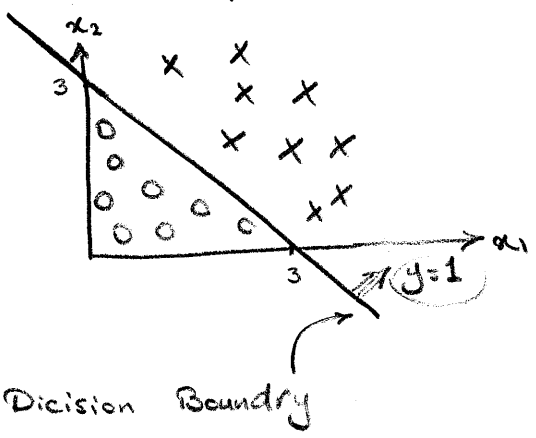
$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

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

↳ بردار

↳ $h_\theta(x) = 0.5$

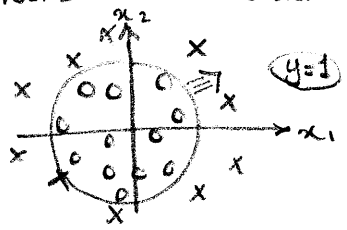
↳ Dicision Boundry

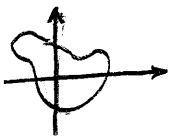


Non_Linear Dicision Boundry

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

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ → predict "y=1" if $x_1^2 + x_2^2 \geq 1$





شکل فرضیه، خاصیت Decision Boundry را تعیین می‌کند، نه داده‌ها

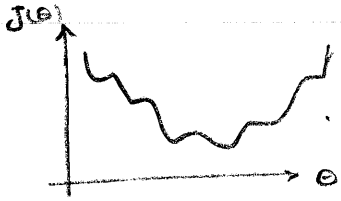
Cost Function

Training Set : $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

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

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

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



تابع هزینه convex است ولی برای تابع سیگموید non convex است.

→ logistic regression cost

دنبال تابع convex برای تابع سیگموید می‌گردیم.

اثبات چگونگی این تابع با تخمین ML است.

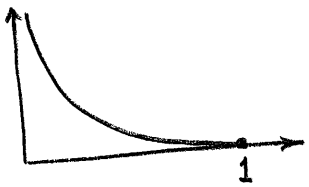
$$\begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases}$$

خواص:

$y=1 \Rightarrow h_\theta(x)=1 \Rightarrow \text{cost}=0$

$h_\theta(x) \to 0 \Rightarrow \text{cost} \to \infty$

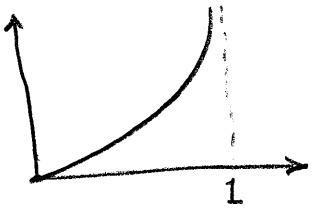
if $h_\theta(x)=0$ → predict $p(y=1|x;\theta)=0$ but $y=1$ → large penalty



$y=0 \Rightarrow h_\theta(x)=0 \Rightarrow \text{cost}=0$

$h_\theta(x) \to 1 \Rightarrow \text{cost} \to \infty$

if $h_\theta(x)=1$ → predict $p(y=0|x;\theta)=0$ but $y=0$ → large penalty



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

$$= -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$ → ! شبیه linear regression ولی $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

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

کنترل α ، feature scaling با این الگوریتم هم اعمال می‌شود.

Advanced Optimization •

cost function $J(\theta)$ → want $\min_\theta J(\theta)$

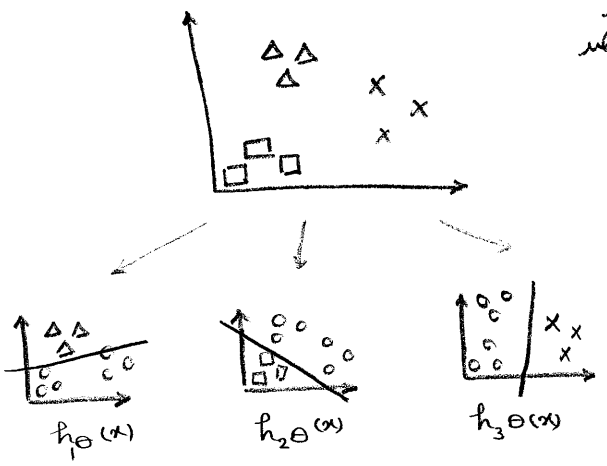
optimization → input $J(\theta)$, $\frac{\partial}{\partial \theta_j} J(\theta)$

- Gradient Descend
- Conjugate Gradient
- BFGS
- L-BFGS

} →
- ☺ No need to pick α
- ☺ Faster than GD
- ☹ More Complex

```
function [jVal, gradient] = costFunction(theta)
options = optimset('GradObj', 'on', 'MaxIter', '100')
initialTheta = ...
[ ] = fminunc(@costFunction, initialTheta, options);
```

Multi Class Classification : one-vs-all •



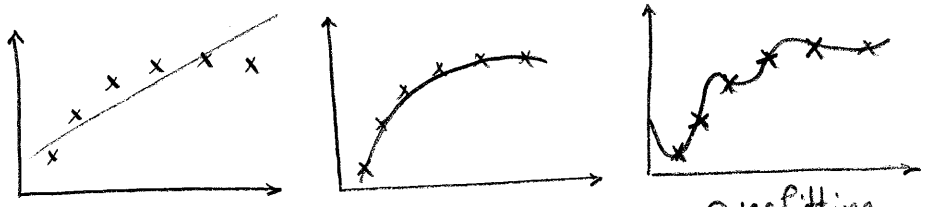
— ساختن training set مصنوعی با قرار دادن target روی کلاس منفی

— آموزش classifier روی هر training set

— برای هر داده ورودی x ، هر سه $h_\theta(x)$ را حساب می‌کنیم که در واقع احتمال عضویت در آن کلاس هستند. پیش‌بینی بدار آن کلاس را مشخص می‌کند.

Regularization *

Overfitting .



Underfit
High Bias

Overfitting
High Variance

↳ not enough data, function goes through all data

Too many features → fit well
↘ fail to generalize (for new data)

Solution .

1. Reduce # of Features:
   - Manually
   - Model Selection Alg.

2. Regularization
   - Keep all features, but reduce magnitude/values of θ → تمام feature ها را نگه دارد، ولی مقدار θ ها را کوچک

Regularization .

مقدار θ ها کوچک ← فرضیه ساده‌تر ← خطر overfitting کمتر

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

↳ λ: regularization parameter → balances fitting & generalization → automation needed

↳ θ_0 is not penalized conventionally.

λ too large → $h_\theta(x) = \theta_0$ → underfitting

Regularized Linear Regression .

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

Gradient Deseend version:

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

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

معمولا کوچکتر از 1 مثلا 0.99

↳ $1 - \alpha \frac{\lambda}{m} < 1 \rightarrow \alpha \frac{\lambda}{m} > 2$

Normal Equation version :

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0 \longrightarrow \theta_j^* \quad \Rightarrow \theta = \left(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix}}_{(n+1)\times(n+1)} \right)^{-1} X^T y$$

☑ $n=2 \longrightarrow \theta = \left(X^T X + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}\right)^{-1} X^T y$

if $m \leqslant n \longrightarrow$ # examples < # features $\longrightarrow$ w/o reg.: singular matrix
↘ w reg. : invertible

• Regularized Logistic Regression

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

Gradient Descend Version : Like Linear Regression with $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

• Advanced Optimization

Cost Function $\longrightarrow$ change $J(\theta)$
↘ change $\frac{\partial}{\partial \theta_j}(\theta)$ $j = 2, \dots, n+1 \longrightarrow \theta_1, \dots, \theta_n$

✶ Neural Networks

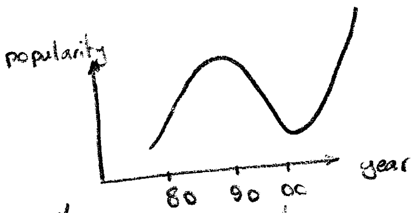
• چرا؟

logistic regression with full set of polynomial features $\longrightarrow$ برای تعداد feature بیشتر تعداد ترکیبهای نهایی آنها زیاد می‌شود.

logistic regression with a subset of polynomial f $\longrightarrow$ $O(n^2)$ کافی نیست / دشواری انتخاب

high number of simple features $\longrightarrow$ ☑ intensity of pixels

• Background



– الگوریتم‌هایی که از مغز تقلید می‌کنند

– "one learning hypothesis" ← بجای سیگنال‌های شنوایی، سیگنال‌های بینایی را به auditory cortex وصل می‌کنند و می‌تواند بخشی از وظایف بینایی را انجام دهد. همین طور برای somatorysensory cortex

– Seeing with tongue

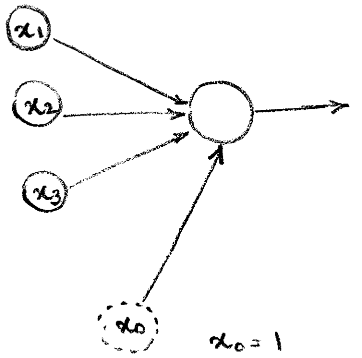
Brain Port

– Human Ecolocation (Sonar)

– Haptic belt : Direction Sense

– Implanting a 3rd Eye

Neuron Model: Logistic



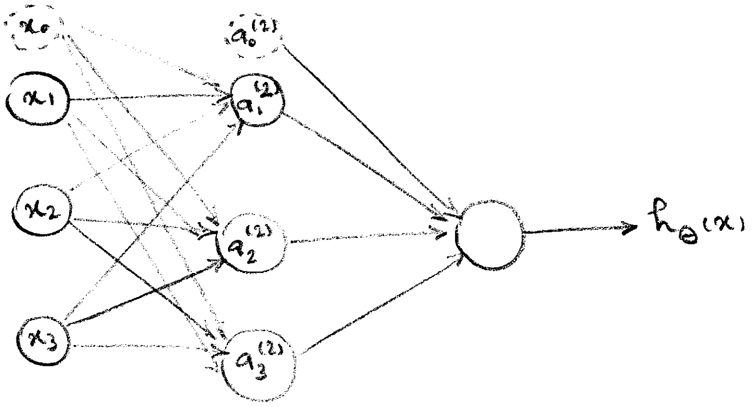
$$h_\Theta(x) = \frac{1}{1+e^{-\Theta^T x}}$$

sigmoid activation function

Θ: parameters / weights

$x_0 = 1$ bias unit

Neural Network



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

$\Theta^{(j)}$: weights between layer j and $j+1$

input layer — hidden layer — output layer

s_1 units — s_2 units

$$\Longrightarrow \Theta^{(j)}_{s_{j+1} \times (s_j + 1)}$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \dots) = g(z_1^{(2)})$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \dots)$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \Longrightarrow z^{(2)} = \Theta \underbrace{x}_{a^{(1)}} \Longrightarrow a^{(2)} = g(z^{(2)}) \xrightarrow{\text{Add } a_0} a = \begin{bmatrix} 1 \\ g(z_1^{(2)}) \\ g(z_2^{(2)}) \\ g(z_3^{(2)}) \end{bmatrix}$$

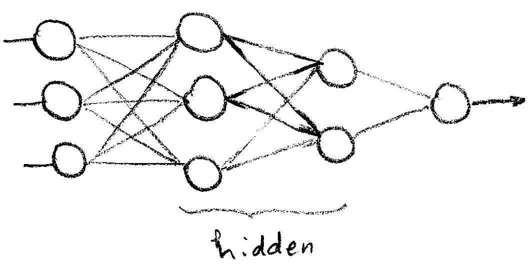
$$\Longrightarrow z^{(3)} = \Theta^{(2)} a^{(2)} \Longrightarrow a^{(3)} = g(z^{(3)}) = h_\Theta(x)$$

محاسبات رو به جلو است و برای همین آن را forward propagation می‌نامند.

در هر لایه یک logistic regres. انجام می‌شود، و داده‌ها به feature های پیچیده‌تر تبدیل می‌شود.

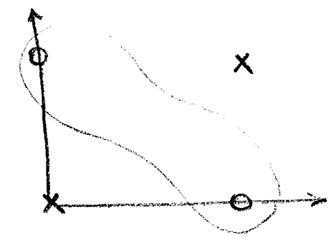
$$\begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \xrightarrow{\Theta^{(1)}} \begin{matrix} a_1 \\ a_2 \\ a_3 \end{matrix} \xrightarrow{\Theta^{(2)}} h_\Theta(x)$$

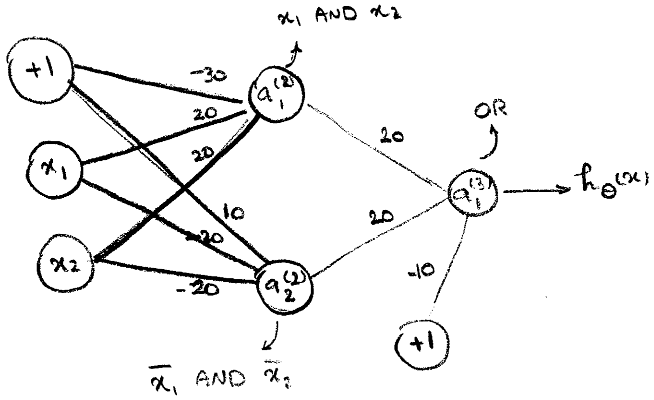
نحوه بهم بندی و ظاهر شبکه را Architecture می‌نامند.



☑ XNOR example

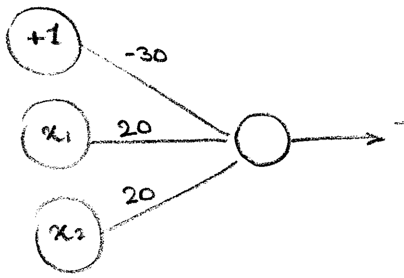
x : y = 1
O : y = 0





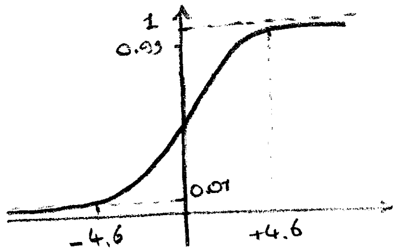
☑ AND example

$y = x_1 \text{ AND } x_2$

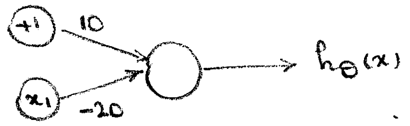


$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$ — $\Theta_{10}^{(1)}$, $\Theta_{11}^{(1)}$, $\Theta_{12}^{(1)}$

x_1	x_2	
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$



☑ NOT example



می‌خواهیم x_1 را NOT کنیم
یعنی جلوی آن عدد منفی بزرگ می‌گذاریم

x_1	$h_\Theta(x) = g(10 - 20x_1)$
0	$g(10) \approx 1$
1	$g(-10) \approx 0$

- One-vs-all ↩ Multiple Output Units

$h_\Theta = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ or $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ or ...

$y^{(i)} \in \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$

- Cost Function

L : # of layers
S_ℓ : # of units in layer ℓ (not counting bias units)

Binary Classification : $y = 0$ or 1
$h_\Theta(x) \in \mathbb{R}$
$S_\ell = 1$
$K = 1$

Multi Class : $y \in \mathbb{R}^K$ $\begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$
$h_\Theta(x) \in \mathbb{R}^K$
$S_\ell = K$

cost function of NN is generalization of Logistic Regression

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

$(h_\Theta(x))_i = i_{th}$ output

Gradient Computation •

$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)} a^{(1)} \Rightarrow a^{(2)} = g(z^{(2)}) \Rightarrow$ add $a_0^{(2)} = 1$ } forward propagation

$z^{(3)} = \dots$

$\delta_j^{(l)}$: error of node j in layer l

$\delta^{(4)} = a^{(4)} - y$

$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* \underbrace{g'(z^{(3)})}_{a^{(3)} .* (1 - a^{(3)})}$

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

No $\delta^{(1)}$ → we can't change inputs } back propagation

if $\lambda = 0 \rightarrow \frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$

Back Propagation Alg. •

Training Set $\{(x^{(1)}, y^{(1)}), \dots (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$

For $i = 1$ to m

- Set $a^{(1)} = x^{(i)}$
- perform forward propagation $\Rightarrow a^{(l)}$ $l = 2, 3, \dots, L$
- $\delta^{(L)} = a^{(L)} - y^{(i)}$
- Calculate $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$
- $\Delta^{(l)} \mathrel{+}= \delta^{(l+1)} (a^{(l)})^T$

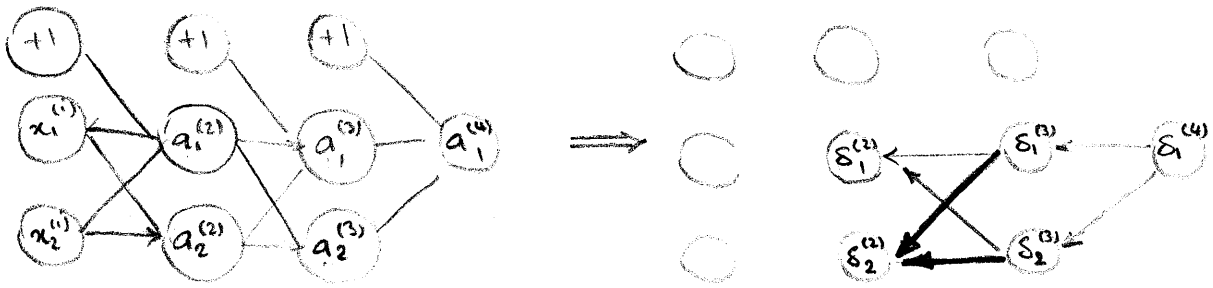
$$D_{ij}^{(l)} = \begin{cases} \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{m} \Delta_{ij}^{(l)} & \text{if } j = 0 \end{cases}$$

$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$

$\delta_2^{(2)} = \Theta_{12}^{(1)} \delta_1^{(3)} + \Theta_{22}^{(2)} \delta_2^{(3)}$

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



Function [jVal, gradient] = costFunction (theta) → should be vector

...

optTheta = fminunc (@costFunction, initialTheta, options)

Implementation Detail.

Unrolling the Matrices: thetaVec = [Theta1(:); Theta2(:); Theta3(:)]

similar for D → gradientVec

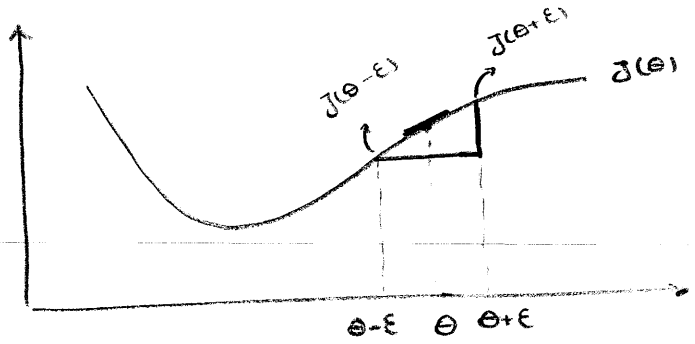
Bringing Back to Matric: reshape (thetaVec (0: m×n), m, n)

reshape (thetaVec (m×n+1 : 2a×a×n), m, n)

...

Gradient Checking.

debugging BP ↵



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

☑ $\varepsilon = 10^{-4}$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta)}{\varepsilon}$$ one sided difference NOT GOOD

$$\frac{\partial}{\partial \Theta_1} J(\theta) \approx \frac{J(\theta_1+\varepsilon, \theta_2, \cdots, \theta_n) - J(\theta_1-\varepsilon, \theta_2, \cdots \theta_n)}{2\varepsilon}$$

...

```
for i = 1 : n
    thetaPlus = theta;
    thetaPlus(i) = thetaPlus(i) + EPSILON;
    thetaMinus = theta;
    thetaMinus(i) = thetaMinus(i) - EPSILON;
    gradApprox(i) = (J(thetaPlus) - J(thetaMinus)) / (2*EPSILON);
```

Random Initialization.

نرون‌های یک لایه را با یک مقدار اولیه

↓

Symmetry

↓

Redundant Representation

$a_1^{(2)} = a_2^{(2)}$

$\delta_1^{(2)} = \delta_2^{(2)}$

$\frac{\partial}{\partial \Theta_{01}^{(1)}} J(\Theta) = \frac{\partial}{\partial \Theta_{02}^{(1)}} J(\Theta)$

$\Theta_{01}^{(1)} = \Theta_{02}^{(1)}$

$\Theta_{ij}^{(l)} = 0$ ← Zero Initialization —

Random Initialization —

$\Theta_{ij}^{(l)} \in [-\varepsilon, +\varepsilon]$ ← Symmetry Breaking

Put all together.

Choose Network Arc —

inputs = # of features

outputs = # of classes

یک لایه مخفی بیشتر نرون‌ها معمولا بهتر است، اگر بیشتر از یک لایه داشتیم تعداد نرون‌های هر لایه یکسان باشد

- Training

1. Randomly initialize weights $\longrightarrow \Theta$
2. Forward Propagation $\longrightarrow h_\Theta(x) \longrightarrow a^{(i)}$
3. Implement Cost Function $\longrightarrow J(\Theta)$
4. Implement Back Propagation $\longrightarrow \frac{\partial}{\partial \Theta} J(\Theta) \longrightarrow \delta^{(i)}$
5. Gradient Checking $\longrightarrow$ BP vs Numerical Estimate
6. Use Gradiend Descend to $\underset{\Theta}{\text{minimize}}$ $J(\Theta) \longrightarrow J(\Theta)$ is non-convex
   ↳ Advanced optimization

* Advice On Applying ML

• Debugging → Large Error ? →
- More Training Examples
- Try smaller sets of features
- Try getting additinal features
- Try adding polynomial features
- Increase / Decrease λ

• Diagnostic → تستهایی که روی ML اعمال می‌شود تا بفهمیم چه خوب کار می‌کند، چه چیز نه → افزایش Performance

• Hypothesis Evaluating

- Split Data → 70 % → Training Set → # m
  → 30 % → Test Set → # m_{test}
  → Randomly Choose → برای نمونه‌برداری از تمام داده

- Procedure

1. Learn Parameters $\Theta \longrightarrow J(\Theta)$ minimized on training
2. Compute Test Error $\longrightarrow J_{test}(\Theta)$ on test set

Misclassification Error_

$$err(h_\Theta(x), y) = \begin{cases} 1 & h_\Theta(x) \geq 0.5 \\ 0 & h_\Theta(x) < 0.5 \end{cases}$$

$$Test\_Err = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(.)$$

• Model Selection

- overfitting → training err < generalization err

☑

d: degree of polynomial

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

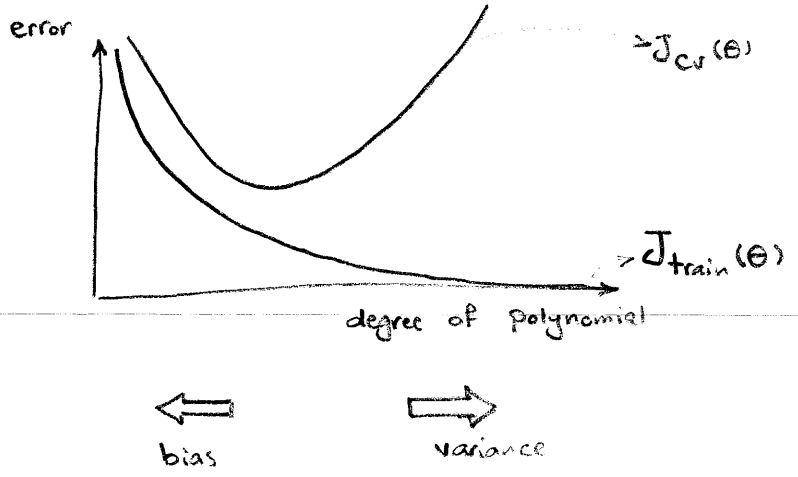
$d=2 \quad h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 \longrightarrow \Theta^{(2)} \longrightarrow J_{test}(\Theta^{(2)})$

$d=3 \quad h_\Theta(x) = \Theta_0 + \Theta_1 x + \Theta_2 x^2 + \Theta_3 x^3 \longrightarrow \Theta^{(3)} \longrightarrow J_{test}(\Theta^{(3)})$ ✓ → new examples ???

$J_{train}(\theta)$
$J_{cv}(\theta)$ $\quad \# m_{cv}$ ← Cross Validation Set 20%
$J_{test}(\theta)$ Test Set 20%

Training Set 60% ← Split Data

☑ $d=1 \rightarrow h_\theta(x) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
$d=2 \rightarrow h_\theta(x) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$ ✓
$d=3 \rightarrow h_\theta(x) \rightarrow \theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$



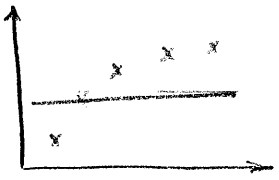
bias ⇐ ⇒ variance

Diagnosing Bias vs. Variance

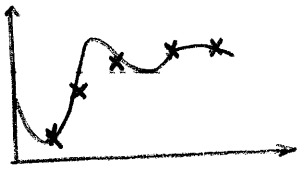
High Bias: $J_{train}(\theta)$ will be high
$J_{cv}(\theta) \approx J_{train}(\theta)$

High Variance: $J_{train}(\theta)$ will be low
$J_{cv}(\theta) \gg J_{train}$

Regularization and Bias / Var



$\lambda = 10000 \quad \theta_1 \approx \theta_2 \approx \dots \approx 0$
$h_\theta(x) \approx \theta_0$
high $\lambda \rightarrow$ underfit



$\lambda = 0.01$
low $\lambda \rightarrow$ overfit

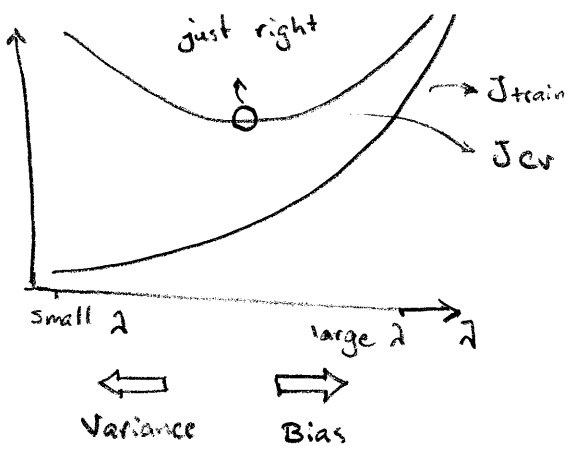
$J(\theta) = \frac{1}{2m}[\text{cost} + \lambda \text{ reg.}]$

$J_{train}(\theta)$
$J_{cv}(\theta)$
$J_{test}(\theta)$
} w/o regularization terms

$\lambda = 0 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
$0.01 \quad \theta^{(2)} \quad J_{cv}(\theta^{(2)})$
0.02
0.02
0.04
$\vdots$
10

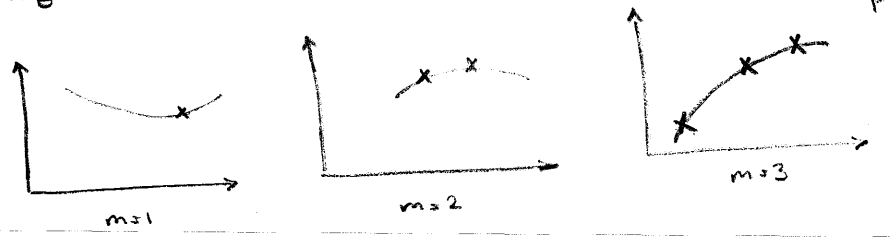
$\rightarrow \min_{\theta^{(5)}} \rightarrow J_{test}(\theta^{(5)})$

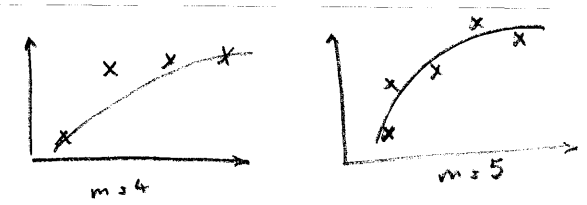
آنقدر λ تست می‌کنیم $\rightarrow \theta$ مربوطه را پیدا می‌کنیم $\rightarrow J_{cv}$ را مقایسه می‌کنیم و بهترین را می‌یابیم $\rightarrow$ پارامتر را می‌یابیم. $\rightarrow$ بررسی در برابر J_{test} می‌کنیم.



learning Curves

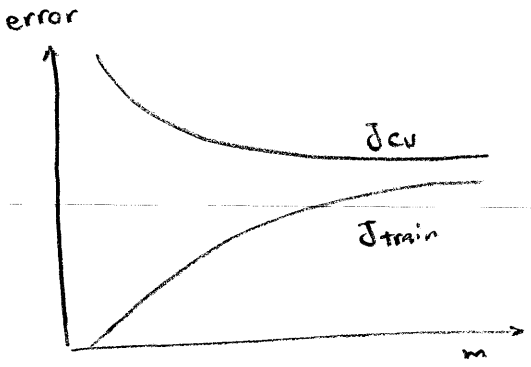
☑ $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



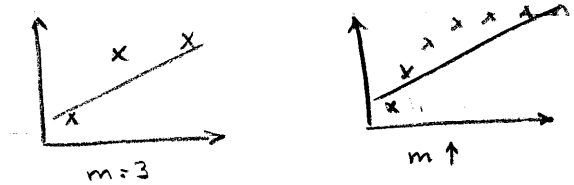


...

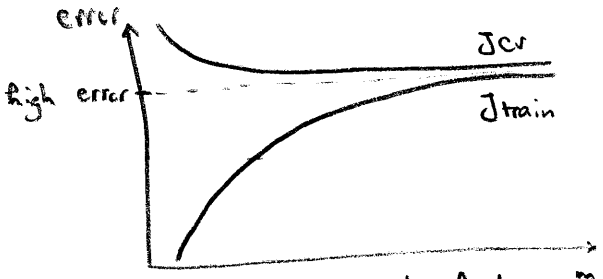
plot error - # of examples -



☑ $h_\theta(x) = \theta_0 + \theta_1 x$

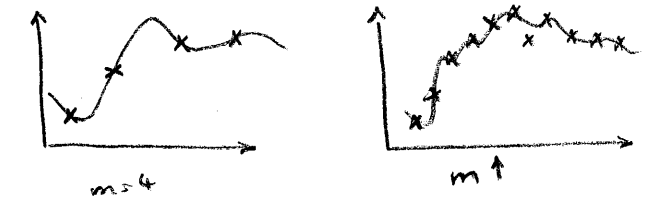


High Bias

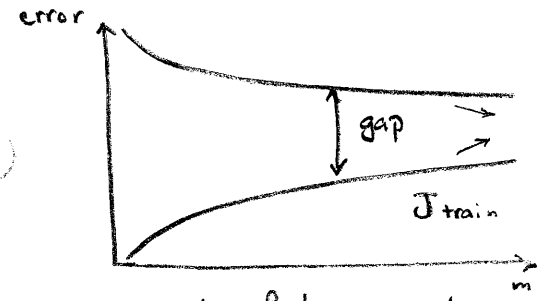


* more data cannot help

☑ $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots \theta_{100} x^{100}$ (and small λ)



High Variance



* more data helps sometimes

what to do next?

- More training example → fix high var
- Smaller sets of features → fix high var
- getting additional features → fix high bias
- adding polynomial features → fix high bias
- decreasing λ → fix high bias
- increasing λ fix high var

Tips for NN

- Small NN → fewer parameters
  - → prone to underfitting
  - → computationally cheaper
- Large NN → more parameters → neurons / hidden layers ↑ → use model selection
  - → prone to overfitting
  - → computationally more complex
  - → use regularization

Designing ML Systems

Example : Spam

- x = features of email
- y = Spam (1) / non Spam (2)
- get 100 words → sort alphabetically → make training vector

{deal, buy, discount, andrew, ... } → $x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$ (andrew, buy, deal, discount)

$x_i = \begin{cases} 1 & \text{if word appears} \\ 0 & \text{otherwise} \end{cases}$

↳ can be chosen by monitoring DB for high frequency words

- Collect lots of data → ☑ honeypots
- Sophisticated Features → ☑ email routing information → email headers
  - → ☑ stemming? Punctuation? → email body
  - → ☑ detect misspelling

Recommended Approach

- Start : Quick Prototype → Test on CV. data
- Plot learning Curves → Decide : more data, more features
- Error Analysis → Manually check False Positives / False Negatives → Trends, Patterns, ...
  - ↳ what type of example it is? → Numeric Evaluation is important !!!
  - ↳ what features are needed to classify them correctly

☑ Pharma : 12
Replica / Fake : 4
Steal Password : 53
Other : 31

Deliberate misspelling : 5
Unusual Email Routing : 16
Unusual Punctuation : 31

↳ Mistakenly Classified By Alg.

- Error Analysis may not be helpful to decide about improving performance.
  ↳ Implement & Decide

☑ Stemming, Case Sensitivity for spam example → with stemming: 3% error
w/o Stemming: 5% error

Error Metrics for Skewed Classes.

☑ Logistic Regression for Cancer
1% error on test set
0.5 % of patients have cancer

اگر همیشه اعلام می‌کردیم که $y=0$ است 0.5 % خطا داشتیم!
حالا اگر دقت از 99.2 % به 99.5 % رسید، آیا الگوریتم بهتر شده است؟

- $y=1$ → presence of rare class that we want to detect

Predicted Class \ Actual Class	1	0
1	True Positive	False Positive
0	False Negative	True Negative

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Pos + False Pos}} \rightarrow \text{Predicted +}$$

چند از پیش‌بینی‌های ما، درباره‌ی کلاس نادر درست بوده است؟

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Pos + False Neg}} \rightarrow \text{Actual +}$$

چند از اعضای کلاس نادر را درست تشخیص داده‌ایم؟

☑ مثال بالا لشن $y=0$ همیشه Recall = 0 است.

- Predict $y=1$ only if very confident

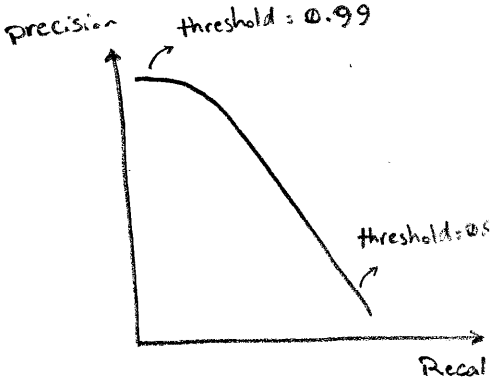
$\begin{cases} h_\theta(x) \geq 0.5 \rightarrow 0.7 \\ h_\theta(x) < 0.5 \rightarrow 0.7 \end{cases}$

⇒ Percision ↑, Recall ↓

- Predict $y=1$ avoiding missing too many cancer

$\begin{cases} h_\theta(x) \geq 0.5 \rightarrow 0.3 \\ h_\theta(x) < 0.5 \rightarrow 0.3 \end{cases}$

⇒ Percision ↓, Recall ↑



- F1 Score

☑

	Percision	Recall	Average	F_1 Score
Alg 1	0.5	0.4	0.45	0.444 ✓
Alg 2	0.7	0.1	0.4	0.175
BAD Alg 3	0.02	1.0	0.51	0.0392
BAD Alg 4	1.0	0.02	0.51	0.0392

×

$$F_1 \text{ Score} = 2\frac{PR}{P+R}$$

$P=0$ or $R=0 \Rightarrow F_1 = 0$

$P=1$ and $R=1 \Rightarrow F_1 = 1$

Data for Machine Learning.

Learning Algorithms with many parameters → low bias algorithms

↘ $J_{train}(\theta)$ small

Use a very large training set → unlikely to verfit

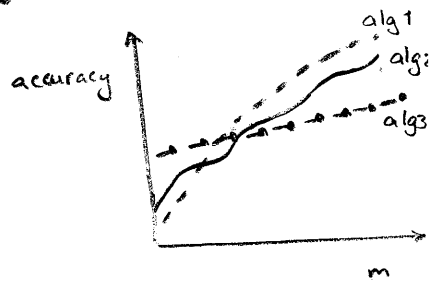
↘ $J_{train}(\theta) \approx J_{test}(\theta)$

+ ⟹ $J_{test}(\theta)$ small

(Large Data Rationale)

test: can a human expert predict results with that info? is data enough?

> $m \uparrow \Rightarrow$ accuracy $\uparrow$ (monotonically)
>
> if alg is low bias



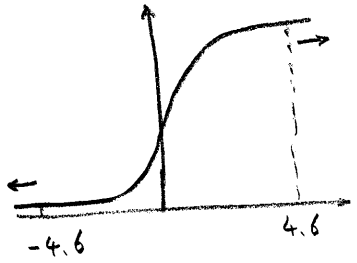
Support Vector Machines

Optimization Objective.

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

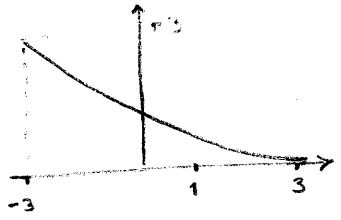
if $y=1$ → we want $h_\theta(x) \approx 1 \rightarrow \theta^T x \gg 0$

if $y=0$ → we want $h_\theta(x) \approx 0 \rightarrow \theta^T x \ll 0$

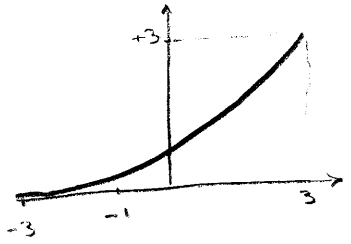


$$\text{Cost} = -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

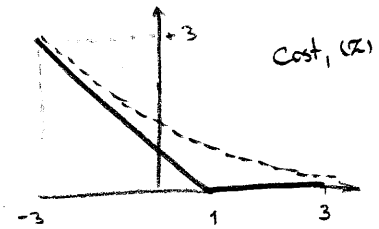
$y=1$

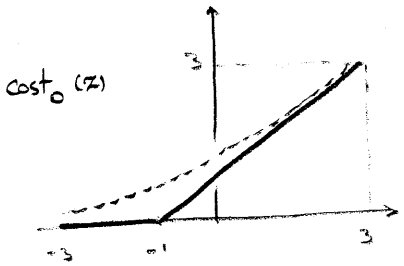


$y=0$



⇓ approx.





cost logistic regression

= A + λ B

err ↙ ↘ reg

⇓

cost SVM

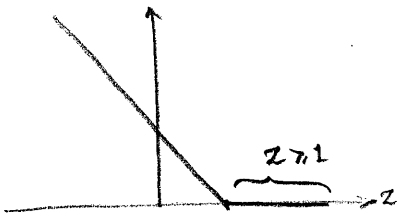
$$\min_\theta \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_2(\theta^T x^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j$$

- * omit $\frac{1}{m}$ in SVM
- * chang from $A + \lambda B$ in Logistic Regres. → $cA + B$ in SVM

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

↳ $$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$ ← احتمال حساب نمی‌شه

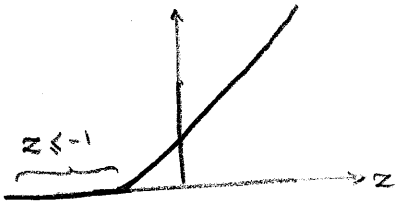
Large Margine Intuition.



$\text{cost}_1(z)$

$y = 1$

want $\theta^T x \geqslant 1$



$\text{cost}_0(z)$

$y = 0$

want $\theta^T x \leqslant 0$

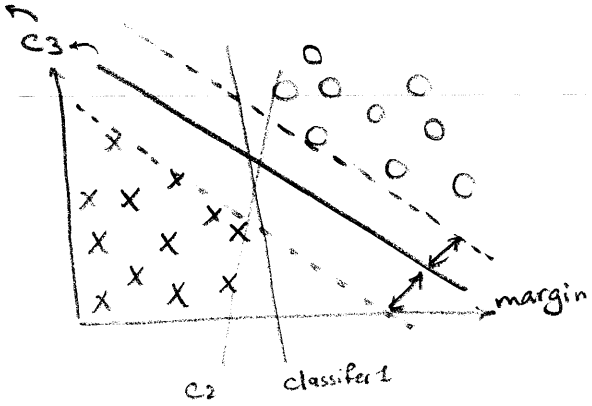
if $C \gg 0$

$$\min \; C \underbrace{\sum \left[y \, \text{cost}_1(\cdot) + (1-y) \, \text{cost}_0(\cdot) \right]}_{0} + \frac{1}{2} \sum \theta_j^2$$

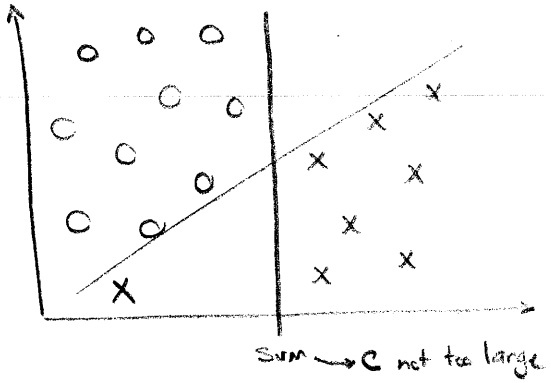
$y^{(i)} = 1 \rightarrow \min \; C \times 0 + \frac{1}{2} \sum \theta_j^2$ s.t. $\theta^T x^{(i)} \geqslant 1$

$y^{(i)} = 0 \rightarrow \theta^T x^{(i)} \leqslant -1$

SVM



Large Margin Classifier



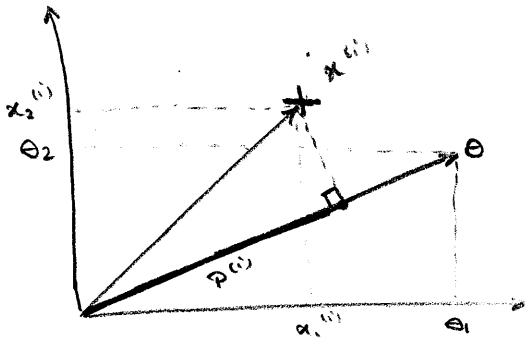
Outliers? Non linear seperable case?

$$\min \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}\left(\sqrt{\theta_1^2 + \theta_2^2}\right)^2 = \frac{1}{2} \|\theta\|^2$$

Simplification: $\theta_0 = 0$, $n = 2$

s.t. $\theta^T x^{(i)} \geqslant 1$ if $y^{(i)} = 1$

$\theta^T x^{(i)} \leqslant -1$ if $y^{(i)} = 0$



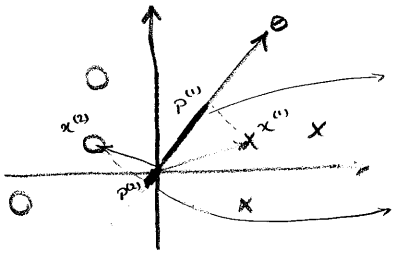
$$\left.\begin{aligned} \theta^T x^{(i)} &= p^{(i)} \|\theta\| \\ \theta^T x^{(i)} &= \theta_1 x_1 + \theta_2 x_2 \end{aligned}\right\}$$

$$\min \frac{1}{2} \sum \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

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

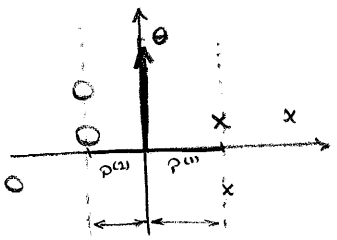
$p^{(i)} \cdot \|\theta\| \leqslant -1$ if $y^{(i)} = 0$

simplification = $\theta_0 = 0$



$p^{(i)} \cdot \|\theta\| \geqslant 1 \rightarrow \|\theta\|$ large

$p^{(i)} \cdot \|\theta\| \leqslant -1 \rightarrow \|\theta\|$ large



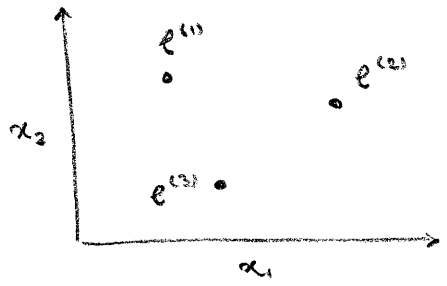
$p^{(i)} \cdot \|\theta\| \geqslant 1 \rightarrow \|\theta\|$ can be small

$p^{(i)} \cdot \|\theta\| \leqslant -1$

Non-Linear Decision Boundary

predict $y=1$ if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \cdots \geq 0$

$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \cdots \geq 0$



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

$f_2 = \text{similarity}(x, l^{(2)})$ and $f_3 = \cdots$

↑ kernel (gaussian kernels)

features are proximity to land marks

$f_1 = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$ → $x \approx l^{(1)} \Rightarrow f_1 \approx 1$

→ x far $l^{(2)} \Rightarrow f_1 \approx 0$

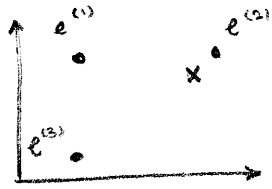
SVM $\begin{cases} h_\theta(x) = 1 & \text{if } \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0 \\ 0 & \text{otherwise} \end{cases}$

☑ $\theta_0 = -0.5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$

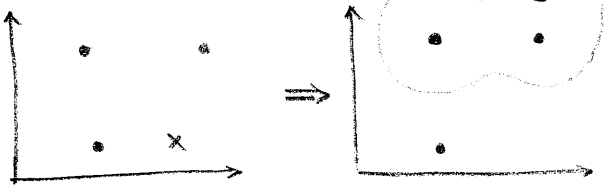
$f_1 \approx 1$ $f_2 \approx 0$ $f_3 \approx 0$

$\theta^T x = 0.5 \geq 0$

predict $y=1$



predict $y=0$



predict $y=0$

kernels

where to get landmarks? → all training examples → $l^{(1)}, \ldots, l^{(m)}$

given x: $f_1 = \text{similarity}(x, l^{(1)})$

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

$\vdots$

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

for training example $x^{(i)}$

$f_i^{(i)} = \text{Sim}(x^{(i)}, l^{(i)}) = 1$

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

predict $y=1$ if $\theta^T f \geq 0$

θ from training: $\min_\theta C \sum \sim + \frac{1}{2}\sum_{j=1}^{m} \theta_j^2$

$m = n$

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

- $\theta^T\theta$ $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$ Regularization اینجا هم نوشته‌ی شود :

- $\theta^T M \theta$ ⟶ scaling for large databases ($m = 10{,}000$)

$C (= \frac{1}{\lambda})$ ⟶ large C : lower bias , high variance
↘ small C : high bias , low variance

• پارامتر C

σ^2 ⟶ large : f_i vary smoothly ⟶ high bias, low var
↘ small : f_i vary less smooth ⟶ low bias, high var

• پارامتر σ^2 توابع گوسی

Using SVM •

- Use Library : liblinear , libsvm
- Choice of C
- Choice of kernel ⟶ linear kernel (no kernel) ⟶ $y=1$ if $\theta^T x \geq 0$
↘ n large , m small ($n \gg m$)
↘ gaussian kernel ⟶ $f_i = \exp\left(-\frac{\|x - \ell^{(i)}\|^2}{2\sigma^2}\right)$, $\ell^{(i)} = x^{(i)}$
↘ use feature scaling
↘ need σ^2
↘ n large , m large
↘ other ⟶ convergance condition : kernel satisfy "Mercer's theorem"
↘ polynomial kernel ⟶ $k(x, \ell) = (x^T \ell)^2$
$= (x^T \ell)^3$
$= (x^T \ell + 1)^3$
$= (x^T \ell + \alpha)^\gamma$
↘ string kernel , chisquare kernel , histogram intersection kernel , ...

- Multi class ⟶ built in in packages
↘ one-vs-all ⟶ pick class i with largest $(\theta^{(i)})^T x$

- n vs m ⟶ $n \gg m$ ⟶ logistic regression , SVM w/o kernel — $n = 10{,}000$, $m = 10 \sim 1000$
↘ n small, m intermediate ⟶ SVM with gaussian kernel — $n = 1 \sim 1000$, $m = 10 \sim 10{,}000$
↘ n small, m large ⟶ add more feature and logistic regression / linear SVM — $n = 1 \sim 1000$, $m = 50{,}000+$

NN works for all but most be slower to train (but local optima is a disadvantage

Unsupervised learning *

Find some structure in data ← unlabled data •

K-means Clustering *

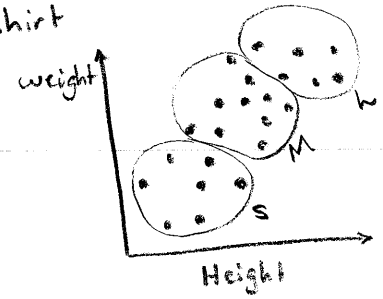
algorithm •

randomly initialize cluster centroids → number = clusters
assignment step → each data is closer to what centroids
move centroids step → move centroids to average of groups

input : K #of clusters , m #of training examples

non-seperated clusters.

☑ T-shirt



K-means can do that!

→ Market Segmentation

Optimization Objective •

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

μ_k = cluster centroid k

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which $x^{(i)}$ is assigned

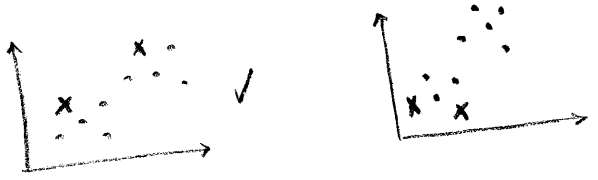
$$J(c^{(1)},\dots,c^{(m)},\mu_1,\dots,\mu_K) = \frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$ → cost function , distortion

cluster assignment : min $J(\cdot)$ w.r.t $c^{(1)}, c^{(2)}, \dots, c^{(m)}$ holding $\mu_1, \dots, \mu_K$

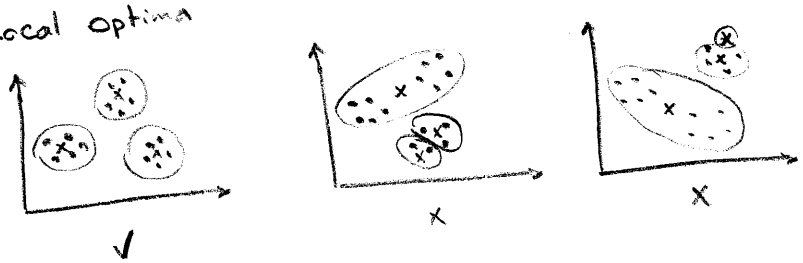
move centroid : min $J(\cdot)$ w.r.t $\mu_1, \mu_2, \dots, \mu_K$ holding $c^{(1)}, \dots, c^{(K)}$

Random Initialization.

- $K < m$
- Randomly pick K examples → set $\mu_1, \dots, \mu_K$ to these examples.



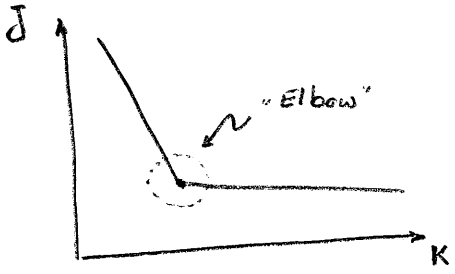
- Local optima

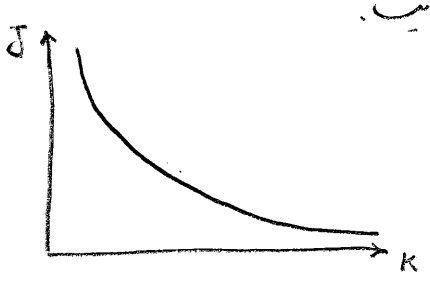


- Run Random Initializations Several times → Pick one giving lowest $J(\cdot)$

• Choosing the number of clusters

تعداد کلاستر‌های بهینه در داده معمولاً واضح نیست.

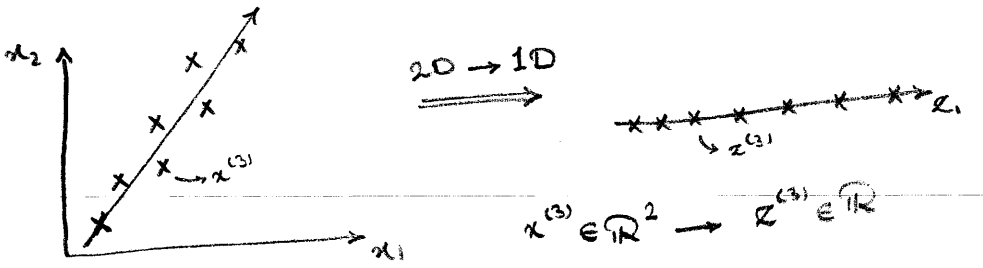




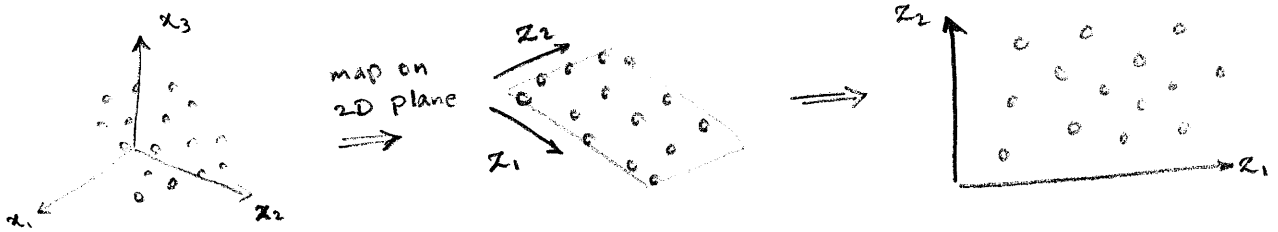
- اگر elbow وجود داشت آن را انتخاب می‌کنیم.

- اگر هدف از clustering خوشه‌بندی بهتری روی داده‌ها انجام می‌دهیم، می‌توان K را با توجه به خروجی آن تغییر داد.

* Dimensionality Reduction



$x^{(3)} \in \mathbb{R}^2 \rightarrow z^{(3)} \in \mathbb{R}$



• کاربرد :

Speed up learning ، Data Visualization ، Data Compression

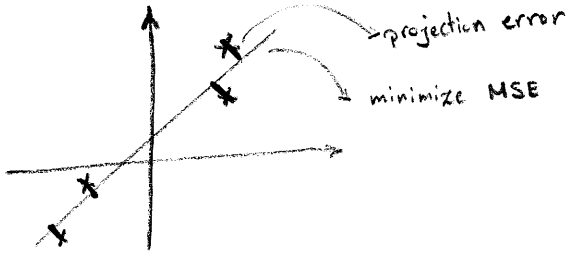
Retain 99% of variance

nD ⇒ 2D

nD ⇒ 3D

nD ⇒ mD

• PCA formulation

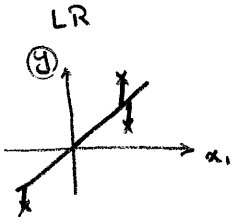


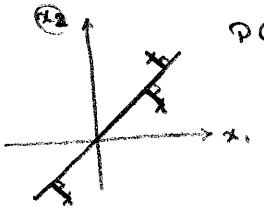
problem : Find a direction onto which to project data so as to minimize the projection error

mD ⇒ kD → find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$

تفاوت با : linear regression

supervised ← LR





PCA → unsupervised

• PCA Algorithm

- Training Set : $x^{(1)}, \ldots, x^{(m)}$

- Preprocessing → feature scaling / mean normalization → Zero Mean is Mandatory

- PCA : nD $\longrightarrow$ kD

Compute covariance matrix $\longrightarrow$ $\Sigma_{n\times n} = \frac{1}{m}\sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$

Compute eigenvectors of Σ $\longrightarrow$ Singular Value Decomposition $\longrightarrow$ svd (matlab)

Take the k first eigen vectors

$$U = \underbrace{\begin{bmatrix} | & | & & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} & \cdots & u^{(n)} \\ | & | & & | & & | \end{bmatrix}}_{k} \in \mathbb{R}^{n\times n} = SVD(\Sigma)$$

$$U_{reduce} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}_{n\times k} \Longrightarrow z_{k\times 1} = U_{reduce}^{T} \times X_{n\times 1}$$

(Matlab)

```
Sigma = X' * X /m ;
[U,S, v]=svd (sigma);
Ured = U (: ,1:k);
z= Ured ' * x ;
```

$$S = \begin{bmatrix} S_{11} & & & 0 \\ & S_{22} & & \\ & & \ddots & \\ 0 & & & S_{nn} \end{bmatrix}$$

Choosing the number of PCs

Average Square Projection Err $= \frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|^2$

Total Variation in data $= \frac{1}{m}\sum_{i=1}^{m} \| x^{(i)} \|^2$

Choose k to be smallest value so that

$$(*)\quad \frac{\sum_{i=1}^{m} \| x^{(i)} - x_{approx}^{(i)} \|}{\sum_{i=1}^{m} \| x^{(i)} \|} \leqslant 0.01 \longrightarrow 1\%$$

"99% of variance is retained"

- start k=1 $\longrightarrow$ check (*) $\longrightarrow$ k++

inefficient

- start k=1 $\longrightarrow$ check (**) $\longrightarrow$ k++

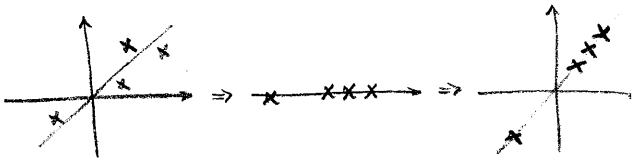
highly efficient

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

$$(**)\quad \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geqslant 0.99$$

$z = U_{reduce}^{T} x \longrightarrow \underbrace{x_{approx}}_{n\times 1} = \underbrace{U_{reduced}}_{n\times k} \cdot \underbrace{z}_{k\times 1}$

Reconstruction



Supervised learning Speed up

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

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

$\{x^{(1)}, \ldots, x^{(m)}\} \xrightarrow[nD]{} PCA \xrightarrow[KD]{} \{z^{(1)}, \ldots, z^{(m)}\} \longrightarrow \{(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})\} \longrightarrow$ training

Cross Validation & Test set should undergo the same PCA, PCA defined on training set.

Not a solution to prevent overfitting •

Use no information about y ↵

Throws away some information ↵

Use regularization instead ⇐

? Part of ML Strategy •

Try your best w/o PCA → improve algorithm for original data

↓

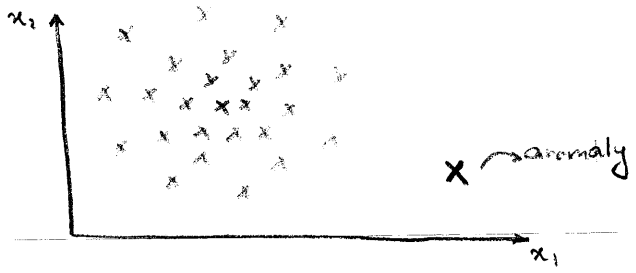
Use PCA ← if it does not what you want

Anomaly Detection * Problem •

☑ Aircraft Engine

x_1 = heat generated

x_2 = vibration intensity



- Problem : Is x_{test} anomalous?

↳ Make Model $p(x)$ → $P(x_{test}) < \varepsilon$ → anomaly

- $P(x_{test}) \geq \varepsilon$ → normal

☑ Fraud Detection

x_1 = Typing Speed of User

x_2 = Login Frequency

x_3 = Transaction Type

⋮

⟹ Identify unusual users

$p(x) < \varepsilon$

☑ Monitoring Computers in data center

x_1 = memory use

x_2 = # disk access / sec

x_3 = CPU load

x_4 = CPU load / Net. traffic

⇒

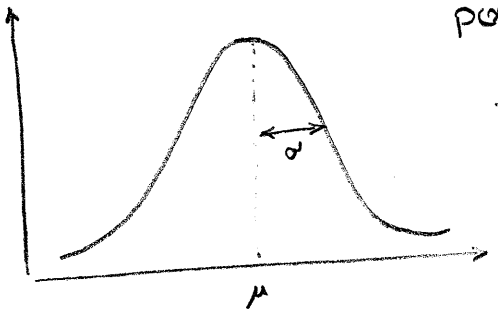
- Gaussian Distribution

mean → μ

variance → σ^2

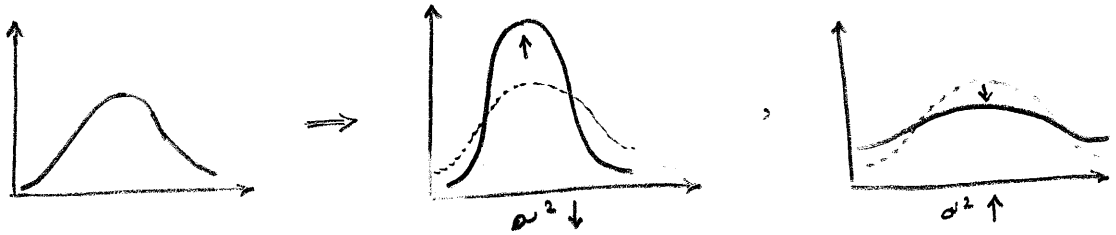
$x \sim \mathcal{N}(\mu, \sigma^2)$

↑ distributed as



Gaussian Distribution •

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



- parameter estimation :

data $\{x^{(1)}, \dots, x^{(m)}\}$

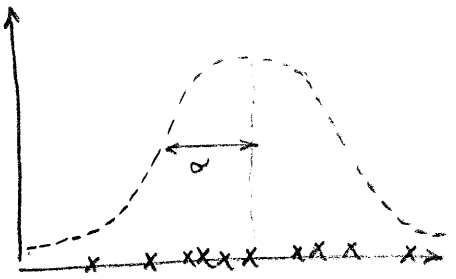
$x^{(i)} \sim \mathcal{N}(\mu, \sigma^2)$

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

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

↳ in ML $\frac{1}{m}$

↳ in Statistics $\frac{1}{m-1}$ → comes from Maximum Likelihood



Anomaly Detection Alg •

- Density Estimation Problem

$x_1 = \mathcal{N}(\mu_1, \sigma_1^2)$

$x_2 = \mathcal{N}(\mu_2, \sigma_2^2)$

$\vdots$

$x_n = \mathcal{N}(\mu_n, \sigma_n^2)$

Independence Assumption between x_i (holds in real world application)

$P(x) = P(x_1; \mu_1, \sigma_1^2) \dots P(x_n; \mu_n, \sigma_n^2)$

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

- Algorithm

1. Choose features x_i might be indicative of anomalous examples
2. Fit parameters μ_i and σ_i^2 → $\mu = \frac{1}{m} \sum x^{(i)}$, $\sigma^2 = \frac{1}{m} \sum (x^{(i)} - \mu)^2$
3. Given new example compute $P(x)$ → $P(x) = \prod \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
4. Anomaly if $p(x) < \varepsilon$

Evaluating Algorithm •

Better Result? ⇐ | New Features —
Threshold Change —

! Labled Data —

$\begin{cases} y = 0 \rightarrow \text{normal} \\ y = 1 \rightarrow \text{anomaly} \end{cases}$

Training Set $x^{(1)}, \dots, x^{(m)}$ → normal examples

CV Set $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m)}, y_{cv}^{(m)})$

Test Set $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m)}, y_{test}^{(m)})$

☑ 10,000 good (normal) engine
20 flawed engine →

training :	6000 good ($y=0$) → μ, σ^2	(60%)
CV :	2000 good, 10 anamalous	(20%)
test :	2000 good, 10 anamalous	(20%)

1. Fit $p(x)$ on training set $x^{(1)}, \ldots, x^{(m)}$
2. On Cross validation / test set

$$y = \begin{cases} 1 & p(x) < \varepsilon \\ 0 & p(x) \geq \varepsilon \end{cases}$$

3. Possible Evaluation Metric (Skew classes): → normal examples are dominating
   - TP, FP, TN, False Negative
   - Percision / Recall
   - F_1 Score
4. Use Cross Validation to Choose ε and new features

Anomaly Det vs. Sup Learning –

anomaly det → very small # of positive examples
- large # of negative examples
- many different types of anomalies → not enough positive examples → hard to learn from positive examples
- future anomalies may look nothing like neg. examples so far ☑

Anomaly → Fraud Detection, Manufacturing, Monitoring
Supervised → Email Spam, Weather Prediction, Cancer Classification

Choosing what Features to Use –

1. Plot histogram of features
   - gaussian ✓
   - non gaussian → algorithm usually handles → do nothing
     - skewed hist → use log transform

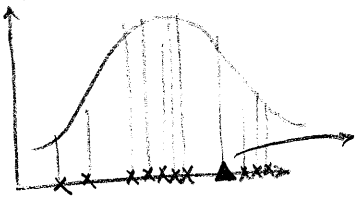
☑ $\log(x_1)$, $\log(x_2+c)$, $\sqrt{x_3}$, $x_4^{\frac{1}{3}}$

☑ hist $(x, 50)$ → hist $(x^{\wedge}0.5, 50)$ → hist $(x^{\wedge}0.4, 50)$
↘ hist $(-\log(x), 50)$

تا زمانی که نمودار نرمال شود.

2. Error Analysis

$p(x)$ → large for normal
↘ small for anomaly

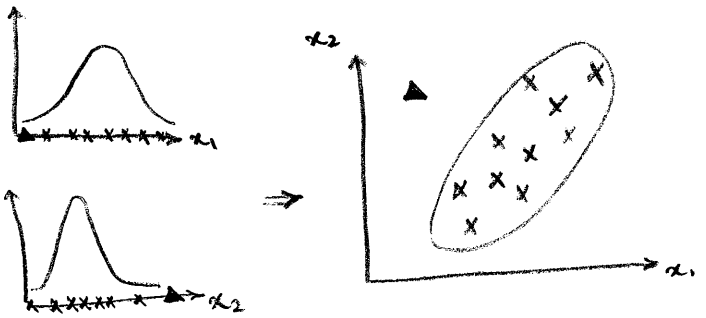


inspect anomalous example for suspecious values

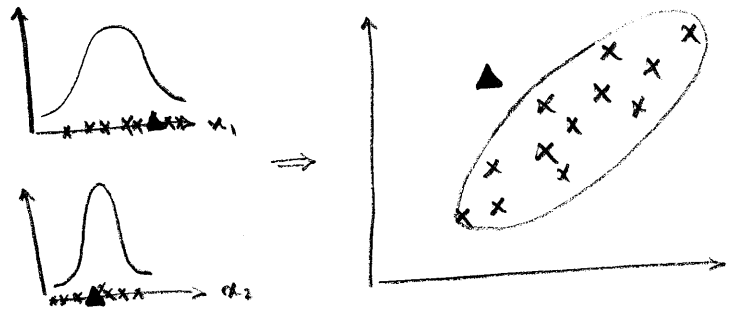
Choose features that might take on unusually large or small values in the event of anomaly

☑ x_3 = CPU load x_4 = Net traffic → x_5 = CPU load / Net traffic → x_6 = (CPU load)^2 / Net traffic

Multi variate Gaussian



OK



Normal Anomaly Detection Fails

↳ Don't Model $p(x_i)$ seperately

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

$\mu \in \mathbb{R}^n$ $\longrightarrow$ $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$

$\Sigma \in \mathbb{R}^{n\times n}$ $\longrightarrow$ $\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$

Algorithm –

1. Fit model $p(x)$ $\longrightarrow$ μ, Σ
2. Calculate $p(x^*)$ for example x^* $\longrightarrow$ $p(x^*)$
3. $p(x^*) < \epsilon$ $\longrightarrow$ Anomaly

Original Model –

$\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ 0 & & \sigma_n^2 \end{bmatrix}$ $\longrightarrow$ no correlation between features

Original Anomaly Detection	Multi variate
* Manually create features to capture when x_1, x_2 take unusual combination	* Automatically captures correlations between features
* Computationaly Cheaper $\longrightarrow$ Scale better	* Computationally More Expensive $\longrightarrow$ Σ^{-1}
* Good for small n	* $m > n$ $\longrightarrow$ Σ is noninvertible for $m < n$ usually used when $m \geq 10\,n$
	* Redundant Features $\longrightarrow$ Rank of $\Sigma \downarrow$ $\longrightarrow$ Σ maybe non-invertible ex $x_1 = x_2$ $x_3 = x_4 + x_5$

Recommender Systems*

☑ Predicting Movie Ratings

	User 1	U2	U3	U4
Romance 1	5	5	0	0
R 2	5	(?)	(?)	0
R 3	(?)	4	0	(?)
Action1	0	0	5	4
Action 2	0	0	5	(?)
	↓			↓
	Romance Lover			Action Lover

n_u = # user

n_m = # movies

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

$y(i,j)$ = rating user j rated movie i

Problem: guess question marks?

↳ recommend to user if they predicted to like a movie

☑ Content-Based Recom.

	U1	U2	U3	U4	x_1	x_2
$x^{(1)}$ → R1	5				0.9	0
$x^{(2)}$ → R2	5				1.0	0.01
$x^{(3)}$ → R3	(?)		•		0.99	0
$x^{(4)}$ → A1	0				0.1	1.0
$x^{(5)}$ → A2	0				0	0.9

$\rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

$\rightarrow x^{(4)} = \begin{bmatrix} 1 \\ 0.1 \\ 1.0 \end{bmatrix}$ ($x_0 = 1$)

$n_u = 4$

$n_m = 5$

$n = 2$ → # features

$m = 5$ → # examples

for each user (j) → learn $\theta^{(j)} \in \mathbb{R}$ → predict user j as rate movie (i) with $(\theta^{(j)})^T x^{(i)}$

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $(\theta^{(1)})^T x^{(3)} = 4.95$

Problem –

$r(i,j)$

$y(i,j)$ = rating of user j for movie i

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

$x^{(i)}$ = features for movie i

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

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

for user j

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

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

$J(\theta^{(1)}, \dots, \theta^{(2)})$

Gradients : video 16.2

Collaborative Filtering •

- Given $y^{(i,j)}$ & $\theta^{(j)}$
- Want x

	U1	U2	U3	U4	x_1 x_2
$x^{(1)}$	5	5	0	0	? ?
$x^{(2)}$	5				
$x^{(3)}$	?				
$x^{(4)}$	0				
$x^{(5)}$	0				

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

$(\theta^{(1)})^T x^{(1)} \approx 5$
≈ 5
≈ 0
≈ 0

$\Rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$

probably a love movie because u1 and u2 that loves romance movie loved it !

learn $x^{(i)}$ — optimization alg —

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

learn x

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

all together —

{ Given x & $y \Rightarrow \theta$
Given $\theta \Rightarrow x$

Guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow \dots$ improve features & parameters by going back & forth

To Minimize Simultaneously :

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

$$\min_{x,\theta} J(\sim)$$

no need to $x_0 = 1$

1. Initialize $x^{(1)} \dots x^{(n_m)}, \theta^{(1)} \dots \theta^{(n_u)}$ to small random values

2. Minimize $J(w)$ using gradient descend

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

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

3. For a user with param θ and movie with (learned features x) predict a rating

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

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix} \quad \begin{matrix} \text{user} \\ \text{movie} \end{matrix}$$

$y^{(i,j)}$

Vectorized Form —

$$(\theta^{(j)})^T x^{(i)} = \begin{bmatrix} (\theta^{(1)})^T x^{(1)} & & \\ & & \\ & & (\theta^{(n_u)})^T x^{(n_m)} \end{bmatrix}$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ \\ — (x^{(n_m)})^T — \end{bmatrix} \qquad \Theta = \begin{bmatrix} (\theta^{(1)})^T \\ \\ (\theta^{(n_u)})^T \end{bmatrix}$$

Low Rank Matrix Factorization

↓

vectorize version of collaborative filtering

How to find movie j related to movie i ?

$\| x^{(i)} - x^{(j)} \|$ small

$\min_j \| x^{(i)} - x^{(j)} \|$ → most similar movie to movie i

Implementation —

☑ new user 5 who haven't rated any movie

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 5 & ? \end{bmatrix} \xrightarrow{n=2} \min J(w) = \dots + \frac{\lambda}{2} \left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2 \right] \rightarrow \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

همه چیز را صفر ارزیابی می‌کند

Mean Normalization

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow \text{new } Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix} \rightarrow \text{train}$$

for user j on movie i predict $(\theta^{(j)})^T x^{(i)} + \mu_i$

☑ us $\rightarrow \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow (\theta^{(5)})^T x^{(i)} + \mu_i = \mu_i$

Mini-Batch Gradient Descent

Batch → Mini Batch → Stochastic

examples in each iter: m b 1

↳ mini-batch size 2~1000

☑ b=10

Get b=10 examples

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)} \quad \forall j = 0, \dots, n$$

- Mini Batch GD → Use Vectorization → Partial Parallelism
  ↘ free parameter b ☹

Convergence

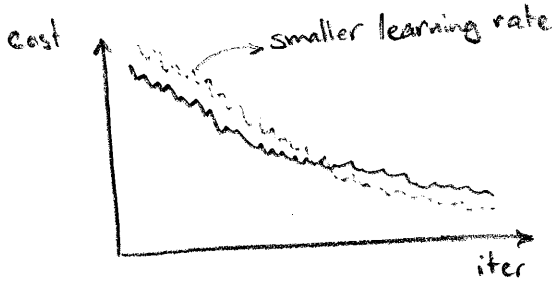
- Batch GD → Plot J_{train} vs iter

- Stochastic GD

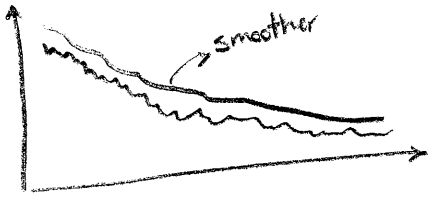
$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

during learning, compute cost before updating θ

every 1000 iter plot averaged 1000 costs → t=1000

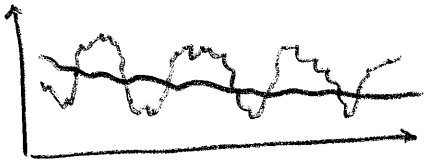


⇒ in SGD the algorith oscillate around min so $\alpha \downarrow$ ⇒ better result

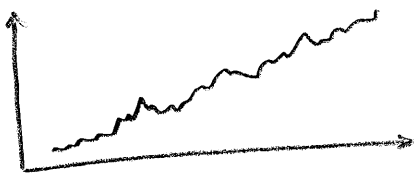


⇒ every 5000 iter

$\underline{t} \uparrow$ → smoother curve



⇒ more appropriate bigger $\underline{t}$
shows trend better



⇒ diverging → use smaller α

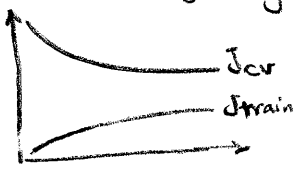
- learning rate → usually constant
  ↘ slowly decrease → $\alpha = \frac{c1}{\text{iter\#} + c2}$ → ☺ convergance
  → ☹ c1 & c2 params

Large Dataset *

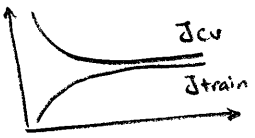
data ↑ ⇒ accuracy ↑ ← how bias ML algorithm.

gradient descend for linear regression → $\sum_{i=1}^{m}$ ... $m = 100,000,000$

learning curve plot ← do sanity check ← why not use $m = 1000$



→ way to go for large m



→ small m is enough

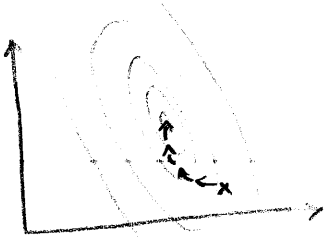
Stochastic Gradient Descent.

here : for linar regression

- $h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$

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

Repeat { $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad \forall j = 0, \dots, n$ }



Batch GD → $m = 300,000,000$ → all in memory

→ $\sum$ over all

- Stochastic GD:

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

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

- Algorithm

1. Randomly Shuffle Dataset
2. Repeat {

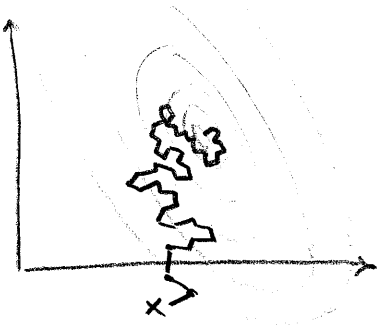
   for $i = 1, \dots, m$ {

   $\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \quad \forall j = 0, \dots, n$

   }

   }

$$\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$$



SGD → every iter is really fast

→ do not converge completely to global minima

→ not neccessarily improves θ in each iter

online learning

Continuous stream of data

☑ Shipping Service price

offer a price
- $y=1$ user accept
- $y=0$ user goes away

$\Longrightarrow$ learn $p(y=1 \mid x;\theta)$ to optimize price

- → use data once then discard it.
- → adapt to change user preferences.

Repeat forever {

Get (x,y) corresponding to user.

Update θ using (x,y)

$$\theta_j := \theta_j - \alpha (h_\theta(x) - y)\, x_j \qquad j=0,\dots,n$$

}

☑ Product Search

user search a query

suggest a number of results
- $y=1$ user clicks
- $y=0$ otherwise

$\Longrightarrow$ learn $p(y=1 \mid x;\theta)$ to predict Click-Through-Rate (CTR)

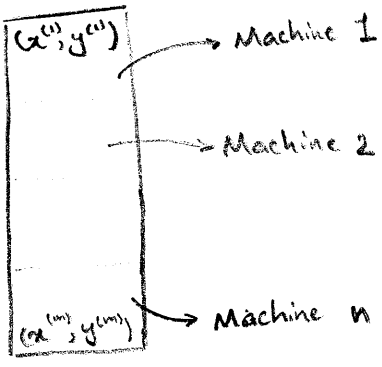
e.g. Suggest 10 phones → 10 training sets (x,y) → 10 iter of learning → discard data

Special offers, news articles, product recommendations, ...

Map-Reduced Data Parallelism

☑ Batch Gradient Descent

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



Use $(x^{(1)}, y^{(1)}) \dots (x^{(100)}, y^{(100)})$

$temp_j^{(1)} = \sum_{i=1}^{100} (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$

Combine:

$\theta_j := \theta_j - \alpha \frac{1}{m} (temp_j^{(1)} + temp_j^{(2)} + \dots)$

- network latency, overhead, combination time → speed is slightly slower than nX
- works for functions expressed as computing sum of function over training set.
- ☑ advanced optimization with logistic regression
- work for multi-core processing → no net. latency