Experiment : (Classification / Regression)

Date

Page No.

✓ Supervised Learning
- right ans. given

Regression
- predict continuous valued o/p

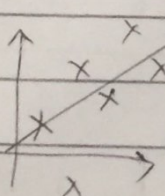
(Find structure in data)
✓ Unsupervised Learning
↳ Clustering ~ Social n/w analysis

* SVD

✓ linear Regression

h - hypothesis
(maps from x's to y's)

Linear Reg (One var.)
$h_\theta(x) = \theta_0 + \theta_1 x$



Cost f^n: Idea is to choose $\theta_0, \theta_1$ so that $h_\theta(x)$ is close to y for our training eg (x, y)

$m \rightarrow$ # training ex.

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

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

Sq error f^n

Gradient Descent
To minimize $J(\theta_0, \theta_1)$

Have: some f^n $J(\theta_0, \theta_1)$
Want: $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:
- Start with some $\theta_0, \theta_1$
- Keep changing $\theta_0, \theta_1$ to reduce $J(\theta_0, \theta_1)$ until we end up @ min

Gradient Descent Algo:

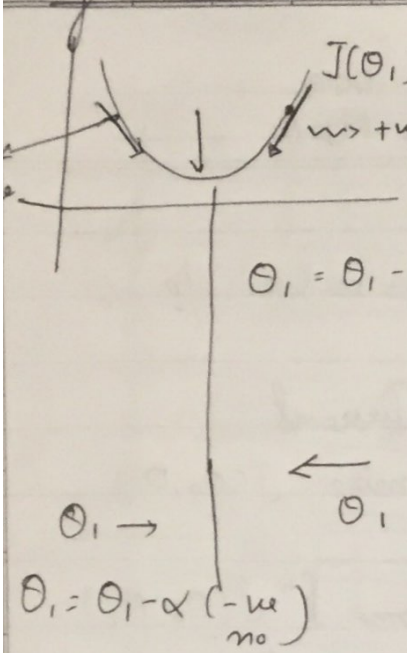
repeat until (convergence)

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0, 1)$$

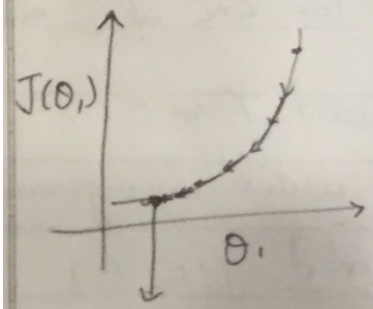
$:=$ Assignment operator

$\alpha$ (learning rate)
$\alpha \downarrow$ - baby steps
$\alpha \uparrow$ - giant steps

## Gradient Descent Intuition




As we approach local min., gradient descent will automatically take smaller steps. So, no need to decrease α over time




Gradient Descent Algo

repeat until convergence {

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

(for $j=1$ & $j=0$)

}

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

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

repeat until convergence {

$\theta_0$ $j=0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

$\theta_1$ $j=1$: $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$

}

"Batch" Gradient Descent
→ Each step uses all train eg

Linear Regression Model

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

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

# Matrix

Experiment : $m \times n$

$A \cdot I = I \cdot A = A$ ($I_{n \times n}$)

$m \times n$ → $m \times n$, $m \times m$, $m \times n$

Note:

$AB \neq BA$

$AI = IA$

$A(A^{-1}) = I$

## Matrix Transpose

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$$

## Linear Regression with multiple variables

$n$ = no. of features

$x^{(i)}$ = i/p features of $i^{th}$ training eg

$x_j^{(i)}$ = value of feature $j$ in $i^{th}$ " (i → training eg no., j → feature no.)

## New hypothesis

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \ldots + \theta_n x_n$ ($x_0 = 1$ for convenience)

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

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

$[\theta_0 \theta_1 \theta_2 \ldots \theta_n]$ ($\theta^T$, $(n+1) \times 1$ matrix → Parameter vector) $\times$ $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ → feature vector

## Gradient Descent

Previously ($n=1$)

Repeat {

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

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

(Simultaneously update $\theta_0 \theta_1$)

New algo:

Repeat {

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

(Simultaneous update $\theta_j$ for $j = 0, \ldots, n$)

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

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

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_2^{(i)}$$

## Feature scaling

Get every feature into $-1 \le x_i \le 1$ range.

### Mean normalization

$x_i \longrightarrow x_i - \mu_i$ → to make zero mean

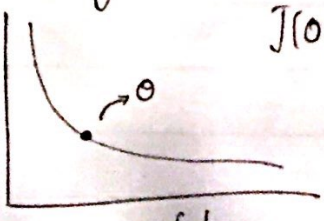
$\mu_i$ → avg. value of $x_i$ in train set

$$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$$

$s_1$ → range (max − min)

Make sure gradient descent works

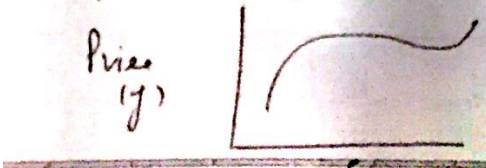
$\min_\theta J(\theta)$




$J(\theta)$ should ↘ with every iter

## Polynomial Regression

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$



## Normal eqn

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

What if $X^T X$ is non-inv? (singular/degenerate)

→ Redundant features
$x_1$ = size in $ft^2$
$x_2$ = " " $m^2$

→ Too many features ($m \leq n$)
- Delete some features / use regularization

'm' training eg, n features

| Gradient Descent | Normal eqn |
|---|---|
| • Need to choose $\alpha$ | • No need |
| • Needs many iteration | • Don't need to iterate |
| • Works well for $n \nearrow$ | • Slow for $n \nearrow$ |

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

probability that $y=1$, given $x$, parameterized by $\theta$

$h_\theta(x) = 0.7$
probability that patient has tumor

$P(y=0 \mid x;\theta) + P(y=1 \mid x;\theta) = 1$

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

## Logistic Regression

Classification

Email: Spam / Not Spam?
Online Transaction: Fraudulent (Y/N)?
Tumor: Malignant / Benign?

$0 \leq h_\theta(x) \leq 1$

$y \in \{0, 1\}$
0 → −ve class
1 → +ve class

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

Want $0 \leq h_\theta(x) \leq 1$

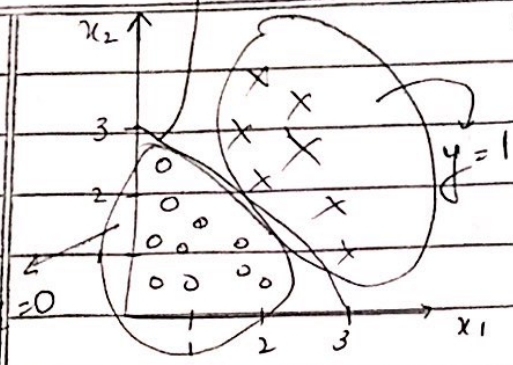
$h_\theta(x) = g(\theta^T x) \rightarrow g(z) = \frac{1}{1+e^{-z}}$

# Decision Boundary

Experiment

Date

Page No.




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

$\hookrightarrow x_1+x_2 \geq 3$

$x_1+x_2=3$

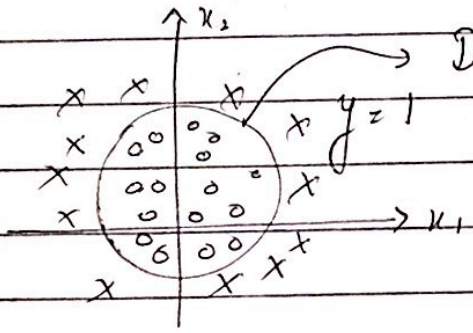
$h_\theta(x)=0.5$

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

$\theta_0=-3,\ \theta_1=1,\ \theta_2=1$

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

## Non-linear decision boundary




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

$$\begin{bmatrix}-1\\0\\0\\1\\1\end{bmatrix}$$

Predict $y=1$ if $-1+x_1^2+x_2^2 \geq 0$

$x_1^2+x_2^2 \geq 1$

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

Cost fn:

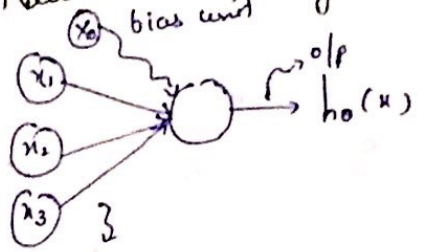
$$\text{Cost}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x)-y)^2$$

$h_\theta(x) \to 0$

Cost $\to \infty$

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

Neuron model: Logistic unit

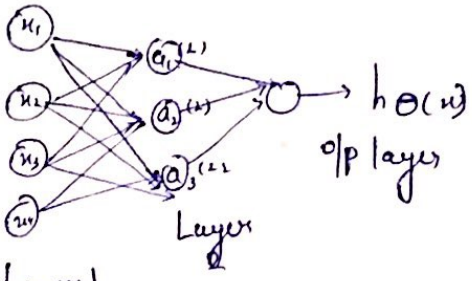



$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Sigmoid (logistic) activation fn

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$\theta$ → "weights"




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

$\Theta^{(j)}$ - matrix of wts controlling function mapping from layer j to j+1

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

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

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

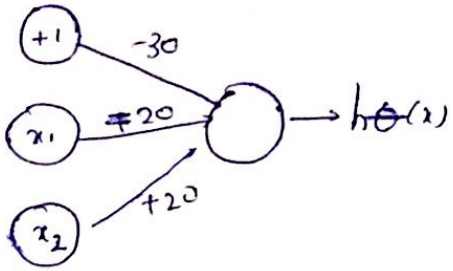
if n/w has $s_j$ units in layer j, $s_{j+1}$ units in layer j+1, then

$\Theta^{(j)}$ - dim - $s_{j+1} \times (s_j + 1)$

## AND

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

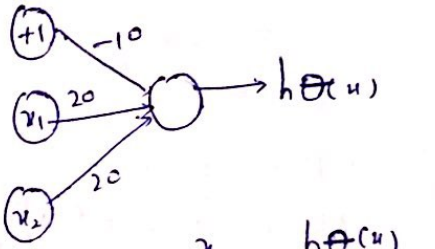
$y = x_1$ AND $x_2$




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

| $x_1$ | $x_2$ | $h_\theta(x)$ |
|---|---|---|
| 0 | 0 | $g(-30) \approx 0$ |
| 0 | 1 | $g(-10) \approx 0$ |
| 1 | 0 | $g(-10) \approx 0$ |
| 1 | 1 | $g(10) \approx 1$ |

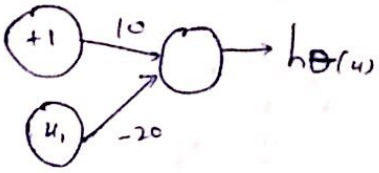
## OR




| $x_1$ | $x_2$ | $h_\theta(x)$ |
|---|---|---|
| 0 | 0 | $g(-10) \approx 0$ |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$g(-10 + 20x_1 + 20x_2)$

## Negation




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

| $x_1$ | $h_\theta(x)$ |
|---|---|
| 0 | $g(10) \approx 1$ |
| 1 | $g(-10) \approx 0$ |

## Multiple o/p units

$h_\theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ - Pedestrian

$h_\theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ - Car

$h_\theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ - Motorcycle

Cost fn for Logistic Reg Vs Neural n/w

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

G. Normal

Logistic regression cost f^n.

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

$Cost(h_\theta(x), y)$

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

Put $y = 1, y = 0$ to get respective cost

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

Want: minimize $J(\theta)$

Repeat {

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

} (simultaneous update $\theta_j$)

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

Optimization algo

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Adv. – No need to pick $\alpha$ manually
– Often faster than GD

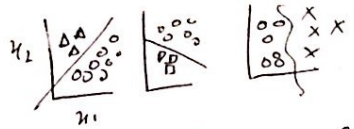
Disadv
– Complex

Multiclass Classification

Email: Work, Friends, Fam, Hobby

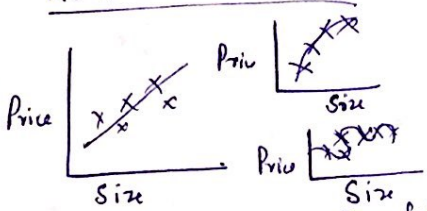
Weather: Sunny, Cloudy, Rainy

One Vs all




Train logistic classifier $h_\theta^{(i)}(x)$ for each class i to predict probability that $y = i$.

REGULARIZATION




Overfitting

⇩

High variance

Too many features

→ fails to generalize new eg

Soln: 1. Reduce features
- Manually select specific ones
- Model selection algo.

2. Regularization
- Reduce mag. of param $\theta_j$
(Keep all features, $\theta_1 \times \theta_2 \approx 0$ → if all $\theta_s$ '0'

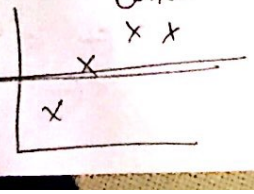
$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$ → compressed param.

$\theta_1, \theta_2, \theta_3 \ldots$

→ cost f^n after regular

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$\theta_0 = h_\theta(x)$




Gradient Descent – linear Reg.

Repeat {

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

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

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

→ 0.99

Logistic Reg.

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

State Of Art

## Precision/Recall

| Predicted Class \ Actual Class | 1 | 0 |
|---|---|---|
| 1 | True +ve | False +ve |
| 0 | False -ve | True -ve |

Precision = $\frac{\text{True +ve}}{\text{True +ve + False +ve}}$

Recall = $\frac{\text{True +ve}}{\text{True +ve + False -ve}}$

→ Of all, what fraction actually has cancer.

→ Of all having, how many correctly detected

| Algo | (P) | (R) | Avg. |
|---|---|---|---|
| 1 | 0.5 | 0.4 | 0.45 |
| 2 | 0.7 | 0.1 | 0.40 |
| 3 | 0.02 | 1.0 | 0.51 |

Avg = $\frac{P+R}{2}$

## F1 Score

$2\frac{PR}{P+R}$

$0 \le F \le 1$ → gives comparision thing

More features / Many training set

Test - better

unlikely to overfit

low bias

Less variance

Sliding window detection.
Increase data set
Distortions introduced!

## SVM

$\text{Cost}_1(z)$

$-\log \frac{1}{1+e^{-z}}$

$\text{Cost}_0(z)$

$-\log\left(1-\frac{1}{1-e^{z}}\right)$

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

SUPERVISED

- Linear Reg
- Logistic Reg.
- NN
- SVMs

## Logistic Reg.

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

UNSUPERVISED

- K-means
- PCA
- Anomaly detection

| F1 Score |
|---|
| 0.444 |
| 0.175 |
| 0.039 |

Recomm.
Large Scale ML

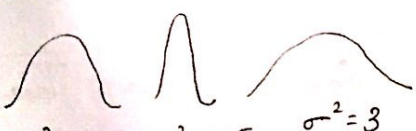
So, SVMs

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

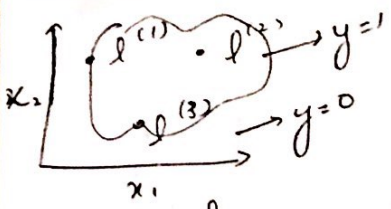
If $y=1$, $\theta^T x \ge 1$ (not just $\ge 0$)

If $y=0$, $\theta^T x \le -1$ (not just $\le 0$)

## Kernels 'σ'



$\sigma^2=1$ $\sigma^2=0.5$ $\sigma^2=3$




Predict when

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

$\theta_0 = -0.5$

$\theta_1 = 1$

$\theta_2 = 1$

$\theta_3 = 0$

$\rightarrow \theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$

$= -0.5 + 1 = 0.5 \geq 0$

---

$f_1, f_2, f_3 \approx 0$

$\theta_0 + \theta_1 f_1 + \ldots = 0.5 \leq 0$

---

Where to get $l^{(1)}, l^{(2)}, l^{(3)}, \ldots$ ?

Given $x$:

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

$= \exp\left(\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right)$

---

Perform feature scaling b4 Gaussian kernel

---

Satisfy Mercer's Theo to make sure SVM pkgs optimize better.

---

## SVM with kernels

Given: $(x^{(1)}, y^{(1)})$

$(x^{(2)}, y^{(2)})$,

$\ldots$

$(x^{(m)}, y^{(m)})$

Choose: $l^{(1)} = x^{(1)}$

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

$l^{(m)} = x^{(m)}$

Given eg $x$:

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

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

$\vdots$

$f_m^{(i)}$

---

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

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

---

Given $x$:

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

$= \exp\left(\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right)$

$\sigma^2 \uparrow$, bias $\uparrow$, variance $\downarrow$

$\sigma^2 \downarrow$, bias $\downarrow$, var $\uparrow$

$C\left(\frac{1}{\lambda}\right)$ — Large $C$ ↑

→ Small $C$

---

Linear kernel = No kernel

Gaussian kernel

$f_i = \exp\left(\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right)$

where $l^{(i)} = x^{(i)}$

---

B4 PCA, data preprocessing

Off-the-shelf Kernels -

1.) Polynomial kernel

2.) More esoteric:
String kernel
Chi-sq. kernel
Histogram intersection kernel

---

★ if $n \geq m$ → train eg. (10K ≥ 1-1K) features
use Logistic Reg or SVM w/o kernel

★ if $n$ is small (1-1000), $m$ intermediate (10-10000)
— Use SVM with Gaussian kernel (50K)

★ If $n$ is small, $m$ is large (1-1K):
Create/add more features, then use logistic reg / linear SVM

★ NN will work 4 all, slow to train though.

## UNSUPERVISED LEARNING

Find patterns in data

### K-means algo

i/p :
- K (no. of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

[unlabeled]

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

Repeat {

for i = 1 to m

$c^{(i)}$ := index (from 1 to k) of cluster centroid closest to $x^{(i)}$

} Cluster assignment step

for K = 1 to k

$\mu_k$ := avg (mean) of pts assigned to cluster k

} move centroid

}

Move $J(\dots)$ wrt $\mu_1 \dots \mu_K$ (cost fn)

Minimize $J(\dots)$ wrt $c^{(1)}, c^{(2)} \dots c^{(n)}$ (holding $\mu_1, \dots \mu_K$ fixed)

Choose value of k :- Elbow method

See the purpose

### Batch GD vs Stochastic GD

### DIMENSIONALITY REDUCTION

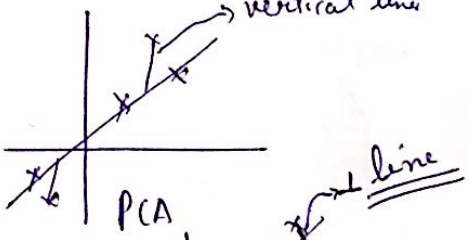
2D → 1D

3D → 1D

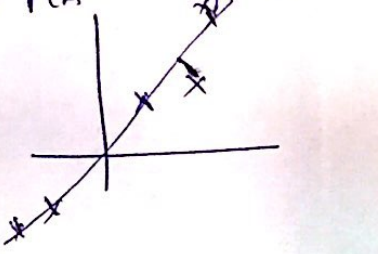
1000 D → 100 D

PCA ≠ Linear Reg.

Linear reg




PCA




### Random initialization

For i = 1 to 100 { → 50 - 1000

Randomly initialize k-means.

Run K-means. Get $c^{(1)} \dots c^{(m)}$, $\mu_1, \dots \mu_K$

Compute cost fn (distortion) $J(c^{(1)}, \dots c^{(m)}, \mu_1, \dots \mu_K)$

}

Pick clustering that gave lowest cost fn $J(c^{(1)} \dots \mu_1, \dots \mu_K)$

---

Bf PCA, data preprocessing is must!

- Reduce data from n-dim to k-dim.

Compute "covariance matrix"

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

($\Sigma$: (n, n) mat.) ($x^{(i)}$: (n×1), $(x^{(i)})^T$: (1×n), n×n)

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

Preprocessing (feature scaling / mean normalization)

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

Replace each $x_j^{(i)}$ with $x_j - \mu_j$

If diff. features on diff scales, scale features to have comparable range of values.

→ Compute eigen vectors of mat $\Sigma$:

[U, S, V] = svd(Sig

n×n matrix

## Choosing k in PCA (no. of principal components)

Avg. squared proj. error:

Total variation in data:

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

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

Choose k such that –

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

99% of variance retained

1 5% "

$\leq 0.05$

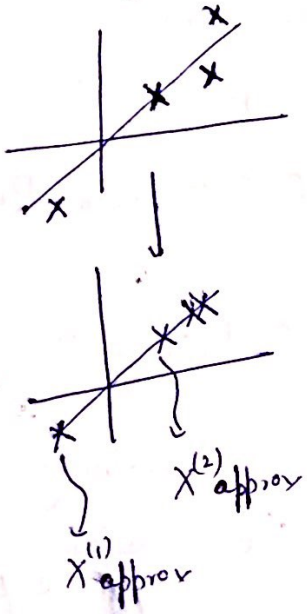
## Reconstruction from compressed rep.

$\mathbb{R}^n$

$x_{approx}^{(i)} = U_{reduce} \cdot z^{(i)}$

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

$(n \times 1)$



## Algo:

Try PCA with k = 1, 3, 4, ... 17

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x_{approx}^{(1)}, \ldots x_{approx}^{(m)}$

Check if –

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

$[U, S, V] = \text{svd}(\text{Sigma})$

$S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & S_{33} & \\ & & & \ddots \\ & & & & S_{nn} \end{bmatrix}$ ($k = 3$)

For given k,

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

or

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

## Use of PCA

Use $U_{reduce}$ Mapping $x^{(i)} \rightarrow z^{(i)}$ can't be applied in cv & test sets, only on training set

Bad use of PCA

– To prevent overfitting

Use $z^{(i)}$ replacing $x^{(i)}$ to reduce no. of features $k < n$,

fewer features, less likely to overfit,

↓

Use regularization instead

## ANOMALY DETECTION:-

- Fraud Detection
  - → $x^{(i)}$ – features of users i's activities
  - → Model $p(x)$ from data
  - → Identify unusual cases by checking $p(x) < \varepsilon$

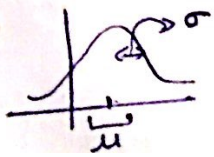
## GAUSSIAN DIST.

If x is a dist Gaussian with mean $\mu$, variance $\sigma^2$

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

↓

dist. as



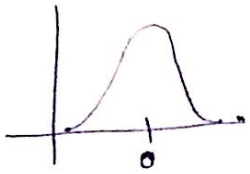
## Gaussian Dist

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

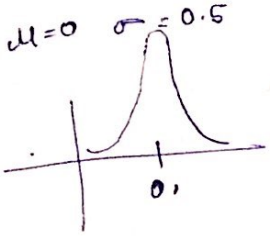
$$p(x;\mu,\sigma^2)$$

$\sigma$ – SD.

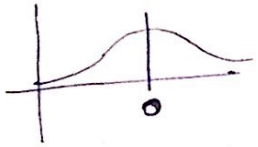
$\mu=0, \sigma=1$



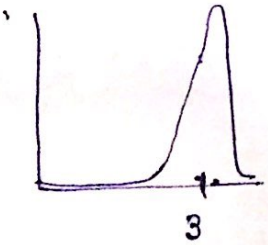

$\mu=0$ $\sigma=0.5$




$\mu=0, \sigma=2$




$\mu=3, \sigma=0.5$




$$\mu = \frac{1}{m}$$

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

↓ avg

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

↓

SD

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

## Density estimation

Training Set:

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

$p(x)$ ↳ features

$= p(x_1;\mu_1,\sigma_1^2)$

$p(x_2;\mu_2,\sigma_2^2)$

$p(x_3;\mu_3,\sigma_3^2)$

$\vdots$

$p(x_n;\mu_n,\sigma_n^2)$

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

(Prod.)

Anamoly detection algo:

1. Choose features $x_i$, indicating anomalous eg
2. Fit param $\mu_1, \mu_2 \dots \mu_n, \sigma_1^2, \sigma_2^2 \dots \sigma_n^2$

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

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

3. Give new eg $x$, compute $p(x)$:

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

$$= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j}\exp\left(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2}\right)$$

Anamoly if $p(x) < \varepsilon$

## Algo evaluation:

→ Fit model $p(x)$ on training set $\{x^{(1)}, \dots, x^{(m)}\}$

→ On a cv/test eg $x$, predict

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

anomaly / normal

- Possible evaluation metrices:
  - True +ve, False +ve, False -ve, True -ve
  - Precision / Recall
  - $F_1$-score

## Anomaly Det. Vs Supervised Learning

Anomaly Det.:
- → Very small no. of +ve eg ($y=1$)
- → Large no. of -ve ($y=0$) eg.
- ⇒ Fraud detection
- ⇒ Mfg
- ⇒ Monitor machines in data center

Supervised Learning:
- Large no. of +ve & -ve eg.
- ⇒ Email spam classification
- ⇒ Weather pred
- ⇒ Cancer classification

## Multivariate Gaussian Dist

Vary $\mu, \Sigma$, as per your case

Parameters $\mu, \Sigma$

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

Fit param:

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

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

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

(cov mat)

Algo –

1. Fit model $p(x)$ by setting $\mu, \Sigma$ as above
2. Give a new eg $x$, compute

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

Flag anamoly if $p(x) < \varepsilon$

## Original Model Vs Multivariate Gaussian

$p(x_1; \mu_1, \sigma_1^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$

Manually create features to capture anamolies

Computationally cheaper

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

Automatically does so

Computationally expensive

---

## Recommender Sys

- Content based
- Collaborative