Machine learning

By - Andrew Ng

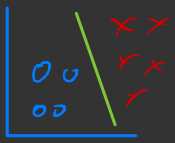
Thu 28 Jul

week 1

- Unsupervised learning:
  - Clustering
  - Anomaly detection
- Recommender systems
- Reinforcement learning

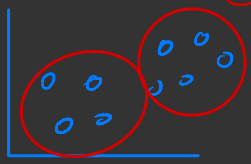
Clustering :

Supervised :



Training set : $(x^{(i)}, y^{(i)})$

Unsupervised :



Training set : $(x^{(i)})$

Application of clustering :

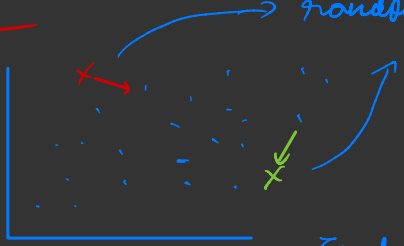
Eg. Grouping similar news

DNA analysis

Astronomical data analysis.

K - Means clustering

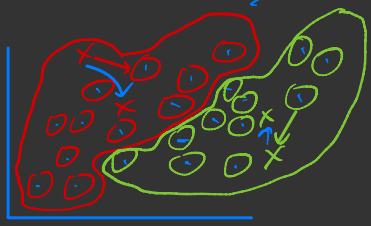
$\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$



Step 1 : Assign each pt. to its closest centroid.

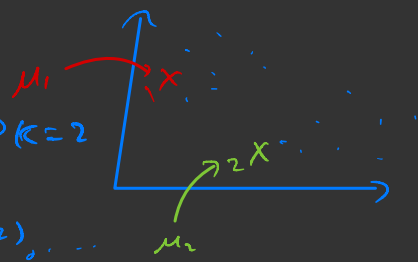
Step 2 : Recompute the centroids

repeat



K-means Algorithm

randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K$ → $K = 2$



μ_1, μ_2 — are vectors of dimension same as $x^{(1)}, x^{(2)}, \ldots$

repeat {

Assign pt.s to cluster centroids

for $i = 1$ to m { m training ex. }

$c^{(i)}$ = index (1 to K) of cluster centroid closest to $x^{(i)}$. $\min_K \|x^{(i)} - \mu_K\|^2$

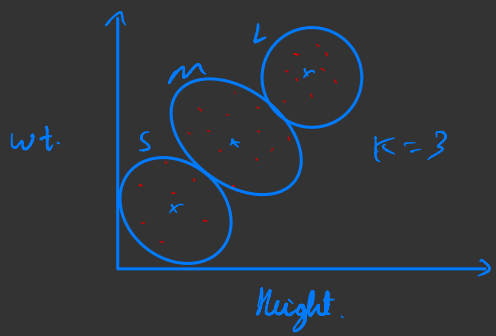
Move cluster centroids

for $k = 1$ to K

μ_K = avg. of pt.s assigned to cluster K.

eg: $\mu_1 = \frac{1}{4}[x^{(1)} + x^{(5)} + x^{(2)} + x^{(3)}]$

If no pt.s get assigned to a cluster then it's better to remove that cluster centroid ~~K~~ $K-1$



T-shirt sizing S M L

Optimization objective:

K means cost fun^n → $c^{(i)}$ = index of cluster (1 to K) to which $x^{(i)}$ is currently assigned.

μ_K = cluster centroid K

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

$x^{(10)}$ c^{10} $\mu_{c^{(10)}}$

Cost fun^n:

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

Distortion

Cost function for K-means

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

Repeat {

 # Assign points to cluster centroids

for i = 1 to m

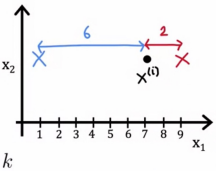
$c^{(i)}$:= index of cluster centroid closest to $x^{(i)}$

 # Move cluster centroids

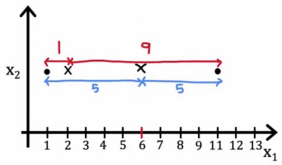
for k = 1 to K

μ_k := average of points in cluster k

}



Moving the centroid



$\frac{1}{2}(1^2 + 9^2) = \frac{1}{2}(1+81) = 41$

$\frac{1}{2}(1 + 11) = 6$

$\frac{1}{2}(5^2 + 5^2) = 25$

Initializing K-means : random location

Step 0: Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots \mu_K$

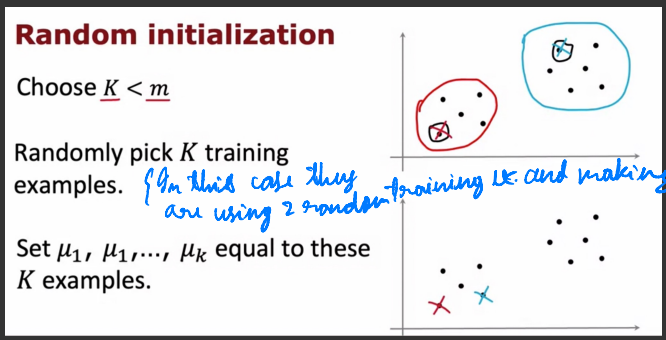
Repeat {

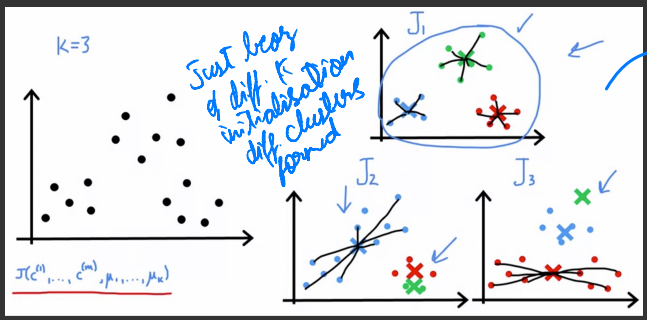
Step 1: Assign pt.s to cluster centroids

Step 2: Move cluster centroids

}

$K < m$





Algo:

For $i = 1$ to (100) { (50-100) ← (random examples.)

Randomly initialize K-means.

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

compute cost fun^n (distortion)

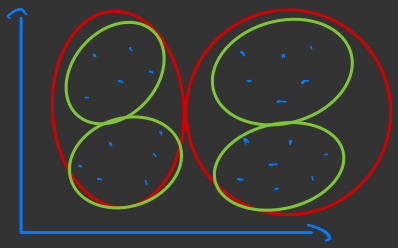
$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots \mu_K)$ ← J

}

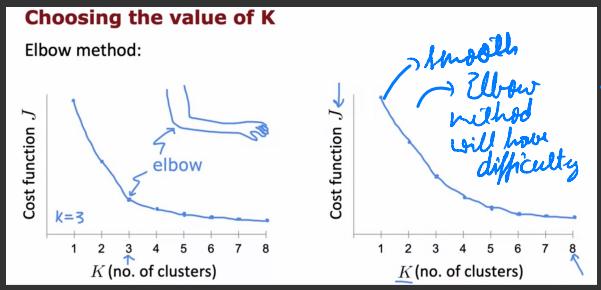
Pick set of clusters that gave the lowest cost fun^n.

Choosing the no. of clusters (K)

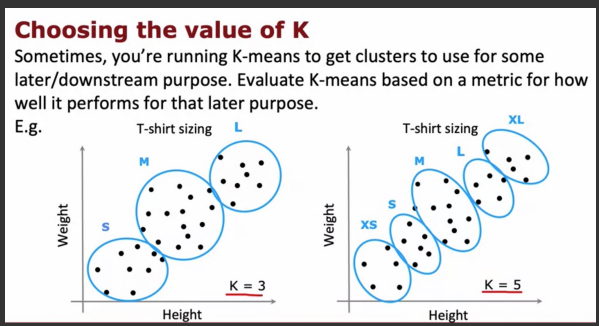
right value of K = ?



Elbow method ? { not preferred }



→ Don't choose K by seeing where the cost fun" is lowest

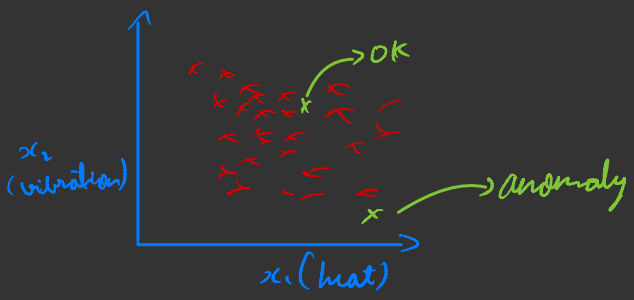


Anomaly Detection

Eg: Aircraft engine features:
x_1 = heat generated
x_2 = vibration intensity

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

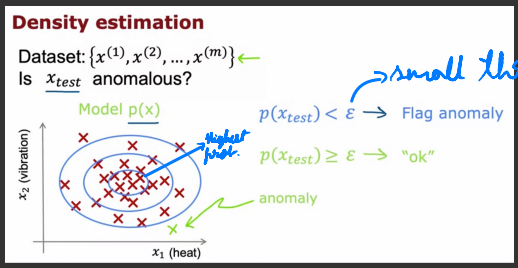
New engine = x_{test}



Is x_{test} anomalous?

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

→ first makes a model called $p(x)$, which tells which x_1 & x_2 are more probable.



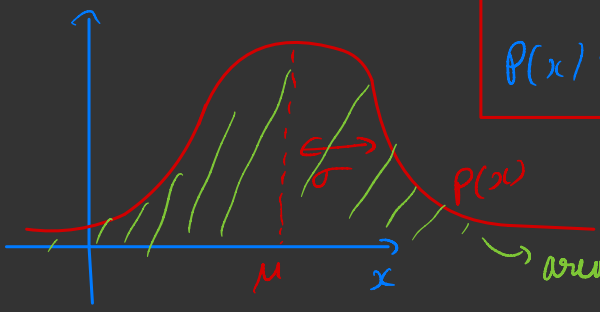
Eg: Fraud detection:
$x^{(i)}$ = features of user i's activities
Model $p(x)$ from data
Identify unusual users by checking which have $p(x) < \varepsilon$

Eg: Monitoring computers in data center.
$x^{(i)}$ = features of machine i like CPU load, memory use etc.

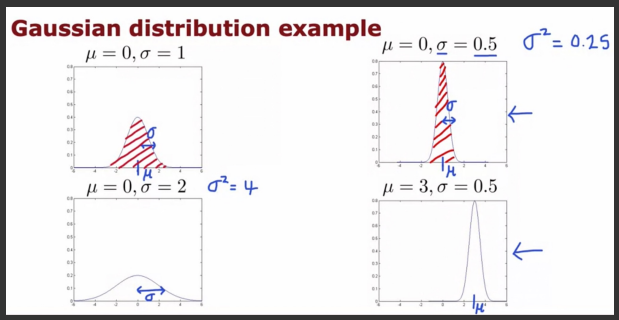
Gaussian Distribution (Normal distribution) for P(x)

Say x is a no. If x is a Distributed Gaussian with mean μ, σ^2 variance

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$



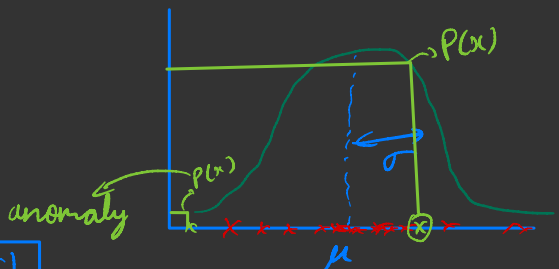
area = 1 {always}



How P(x) or normal distribution varies with μ & σ ↓ Standard deviation.

Parameter estimation :

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



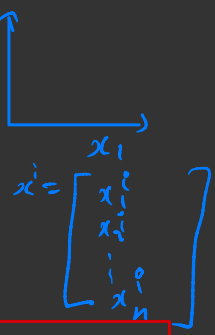
$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

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

What if n no. of features? Then how would you implement gaussian Distribution?

Training set: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

Each example $x^{(i)}$ has n features



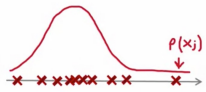
$$P(x) = P(x_1; \mu_1, \sigma_1^2) * P(x_2; \mu_2, \sigma_2^2) \ldots P(x_n; \mu_n \sigma_n^2)$$

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

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

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

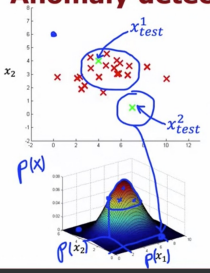


3. Given new example x, compute $p(x)$:

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

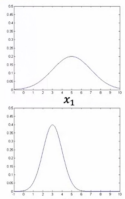
Anomaly if $p(x) < \varepsilon$

Anomaly detection example



$\mu_1 = 5, \sigma_1 = 2$

$\mu_2 = 3, \sigma_2 = 1$



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

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

$\varepsilon = 0.02$

$p(x_{test}^{(1)}) = 0.0426 \rightarrow$ "ok"

$p(x_{test}^{(2)}) = 0.0021 \rightarrow$ "anomaly"

How to choose ε?

The importance of real-number evaluation

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

Assume we have some labeled data, of anomalous and non-anomalous examples. ($y = 0$ if normal, $y = 1$ if anomalous).

Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous) ← $y=0$

Cross validation set: $\left(x_{cv}^{(1)}, y_{cv}^{(1)}\right), \ldots, \left(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})}\right)$

Test set: $\left(x_{test}^{(1)}, y_{test}^{(1)}\right), \ldots, \left(x_{test}^{(m_{test})}, y_{test}^{(m_{test})}\right)$

} Include a few anomalous examples $y=1$

Aircraft engines monitoring example

10000 good (normal) engines ← $y=0$

20 flawed engines (anomalous) $y=1$

Training set: 6000 good engines ← Train algorithm

CV: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$) ← ε, x_j

Test: 2000 good engines ($y = 0$), 10 anomalous ($y = 1$) ←

Alternative:

Training set: 6000 good engines

CV: 4000 good engines ($y = 0$), 20 anomalous ($y = 1$) ← ε, x_j

No test set ←

If less dataset like if anomaly is only 2.

Algorithm evaluation:

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

On a cross validation/test example x, predict

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

10

2000

Algorithm evaluation

Fit model $p(x)$ on training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

On a cross validation/test example x, predict

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

10

2000

Possible evaluation metrics:

- True positive, false positive, false negative, true negative
- Precision/Recall
- F_1-score

Can also use cross validation set to choose parameter ε

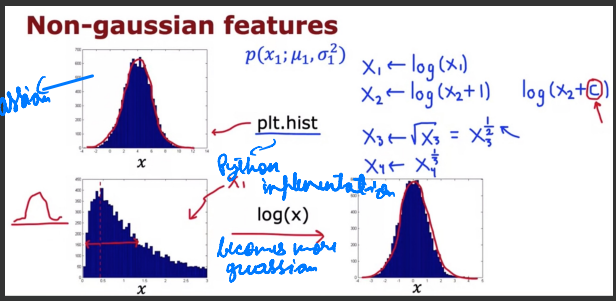
When to use anomaly detection vs when to use supervised learning?

Anomaly detection	vs.	Supervised learning
Very small number of positive examples ($y = 1$). (0-20 is common). Large number of negative ($y = 0$) examples. $p(x)$		Large number of positive and negative examples. 20 positive examples
Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.
Fraud → always new types of frauds		Spam → very similar

Anomaly detection	vs.	Supervised learning
→ Fraud detection		→ Email spam classification
→ Manufacturing - Finding new previously unseen defects in manufacturing.(e.g. aircraft engines)		→ Manufacturing - Finding known, previously seen defects scratches $y=1$
→ Monitoring machines in a data center		→ Weather prediction (sunny/rainy/etc.)
⋮		→ Diseases classification
		⋮

Choosing what features to use :

Non-gaussian features →



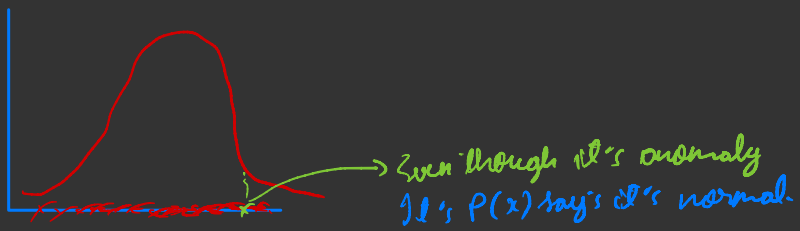
Error analysis for anomaly detection :

want $p(x) \geq \varepsilon$ large for normal examples x.

$p(x) < \varepsilon$ small for anomalous examples x.

most common problem :

$p(x)$ is comparable (say, both large) for normal & anomalous examples.

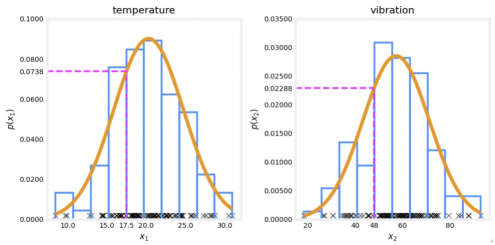


Eg: Monitoring computers in a data center :

x_1 = memory use of computer
x_2 = no. of disk accesses/sec.
x_3 = CPU load
x_4 = network traffic
$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$
$x_6 = \frac{(\text{CPU load})^2}{\text{Network traffic}}$

5. You are monitoring the temperature and vibration intensity on newly manufactured aircraft engines. You have measured 100 engines and fit the Gaussian model described in the video lectures to the data. The 100 examples and the resulting distributions are shown in the figure below.

The measurements on the latest engine you are testing have a temperature of 17.5 and a vibration intensity of 48. These are shown in magenta on the figure below. What is the probability of an engine having these two measurements?



- ○ 17.5 * 48 = 840
- ◉ 0.0738 * 0.02288 = 0.00169