

Unsupervised Learning, Recommender Systems, Reinforcement Learning

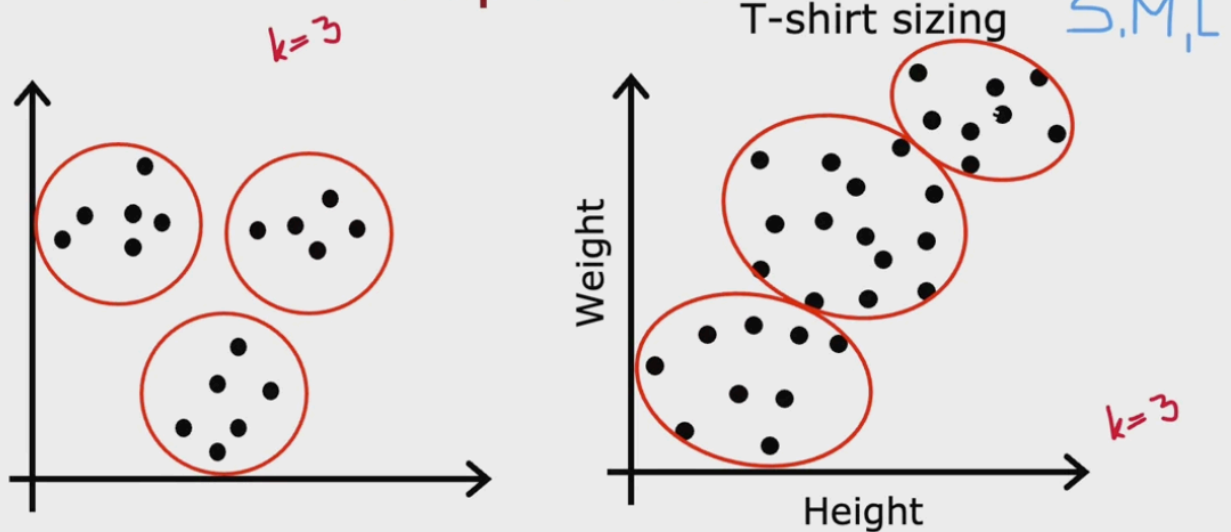
Supervised Learning

Clustering

First - takes random guess, looks how much close these random *cluster centroids* are.

K-means Algorithm

K-means for clusters that are not well separated



Optimizing a specific cost function.

K-means optimization objective

$c^{(i)}$ = index of cluster $(1, 2, \dots, K)$ to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k

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

Cost function

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

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

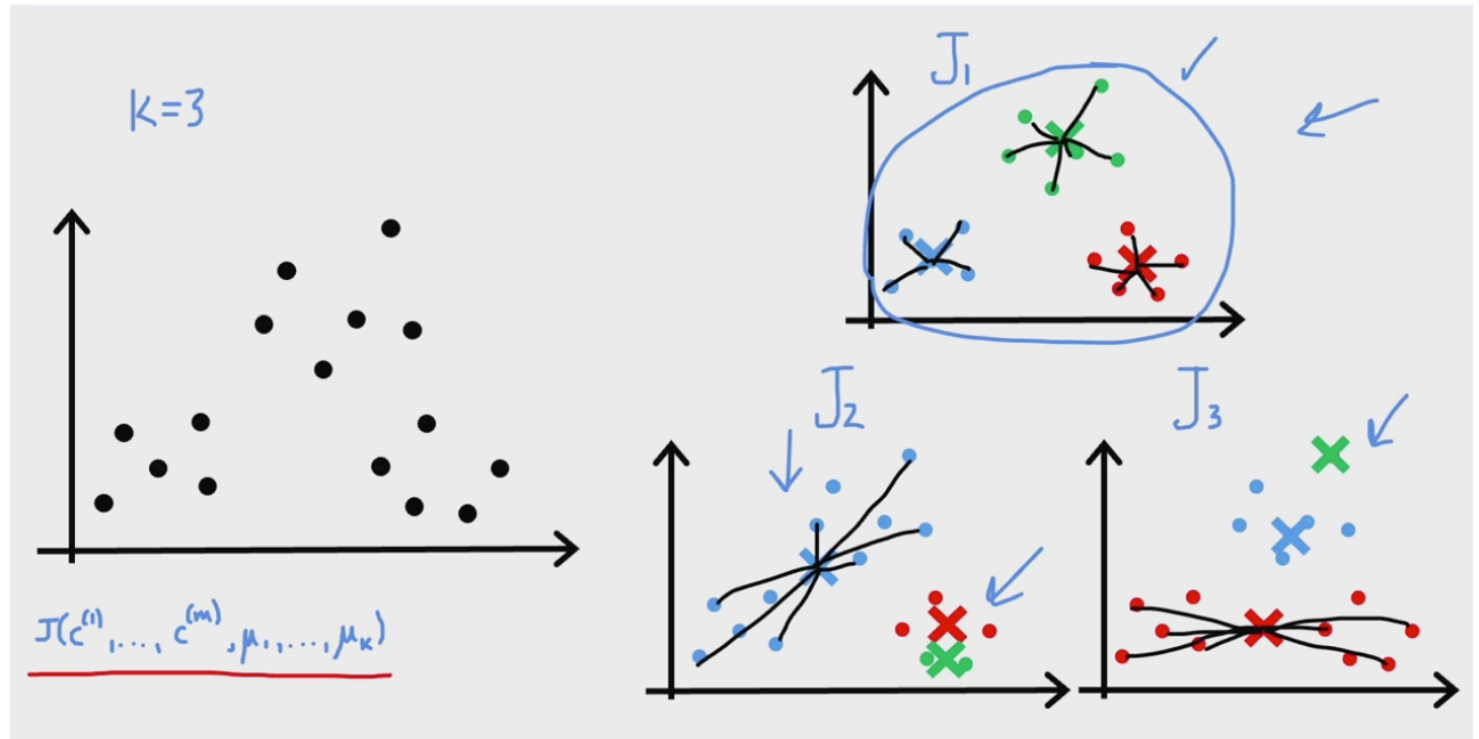
Distortion

Handwritten red notes: $x^{(i)}$, $c^{(i)}$, μ_c

Random initialization of K

$$K < m$$

Local Optima's can be a problem.



Randomly initialize K -means \rightarrow 50-1000 different initializations

Pick set of clusters that gave lowest cost J

Right Value of K

Elbow method - not practical

Anomaly Detection

feature vector

Density Estimation

Probability of x being seen in dataset. areas with different probabilities.

epsilon as the limit. if less then epsilon \Rightarrow anomaly.

where there are many features. (# of pressed keys by user in minute, transaction #, clicks, visits, CPU load, CPU load / network activities)

Gaussian - Normal - Bell Shape Distribution

std dev, (std dev)² = variance

Gaussian (Normal) distribution

σ standard deviation

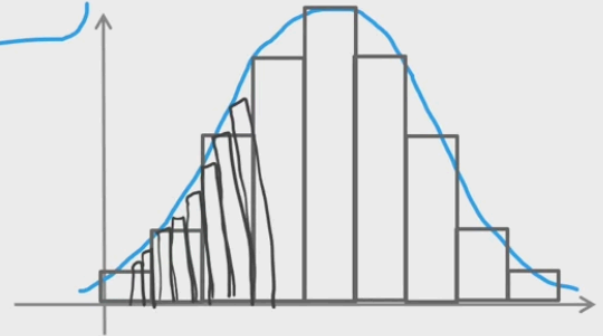
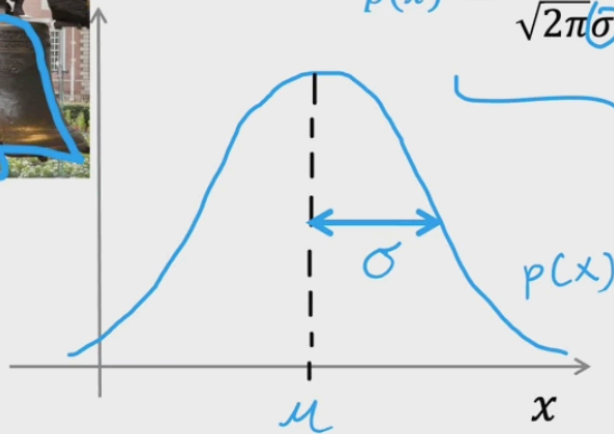
σ^2 variance

Say x is a number.

Probability of x is determined by a Gaussian with mean μ , variance σ^2 .

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

$$\pi = 3.14$$



Anomaly Detection:

Anomaly detection algorithm

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

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

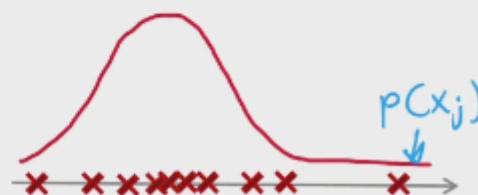
Vectorized formula

$$\bar{\mu} = \frac{1}{m} \sum_{i=1}^m \bar{x}^{(i)} \quad \bar{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_n \end{bmatrix}$$

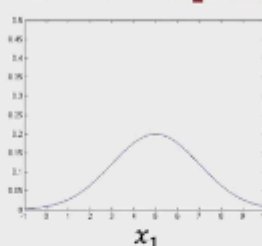
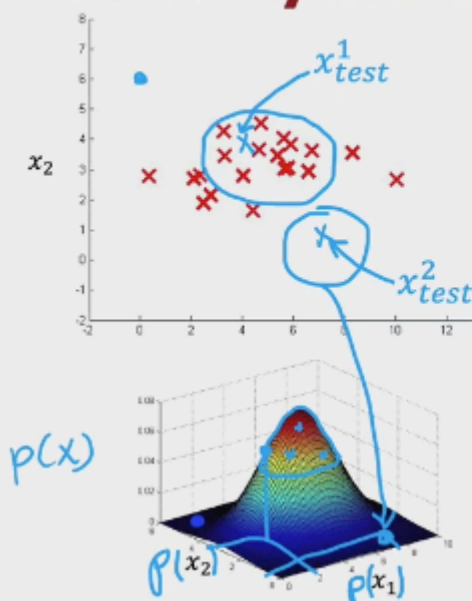
3. Given new example x , compute $p(x)$:

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

Anomaly if $p(x) < \varepsilon$

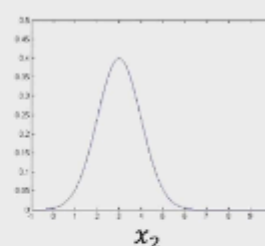


Anomaly detection example



$$\mu_1 = 5, \sigma_1 = 2$$

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



$$\mu_2 = 3, \sigma_2 = 1$$

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

$$\varepsilon = 0.02$$

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

$$p(x_{test}^{(2)}) = 0.0021 \longrightarrow \text{anomaly}$$

How to evaluate anomaly detection systems?

Real-number Evaluation:

Making decisions is much easier if we have a way of evaluating learning algorithm.

Some labeled data, anomalous and non-anomalous.

Aircraft engines monitoring example

10000 good (normal) engines
~~20~~ flawed engines (anomalous)

2 to 50

$y=1$

$y=0$
Training set: 6000 good engines

train algorithm on training set

CV: 2000 good engines ($y=0$) 10 anomalous ($y=1$)

use cross validation set

tune ϵ

tune x_j

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

Alternative: No test set Use if very few labeled anomalous examples

Training set: 6000 good engines 2 higher risk of overfitting

CV: 4000 good engines ($y=0$), ~~20~~ anomalous ($y=1$)

tune ϵ tune x_j

Algorithm evaluation

course 2 week 3

skewed datasets

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

On a cross validation/test example x , predict

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

10

2000

Possible evaluation metrics:

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

Use cross validation set to choose parameter ϵ

Anomaly Detection vs Supervised Learning:

Number of examples is the key factor in between.

Anomaly Detection: Learns what to accept as normal to some extent.

Supervised Learning: Learns what looks like an acceptable or unacceptable example.

Anomaly detection vs. Supervised learning

Very small number of positive examples ($y = 1$). (0-20 is common).
Large number of negative examples ($y = 0$)
examples. $p(x)$

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.

Fraud

Large number of positive and negative examples.

20 positive examples

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.

Spam

Spam -> probably you seen similar before

Fraud -> oncoming examples may be completely different from what we have seen.

Anomaly detection vs. Supervised learning

→ Fraud detection

→ Manufacturing - Finding new previously unseen defects in manufacturing. (e.g. aircraft engines)

→ Monitoring machines in a data center

⋮

→ Email spam classification

→ Manufacturing - Finding known, previously seen defects scratches $y = 1$

→ Weather prediction (sunny/rainy/etc.)

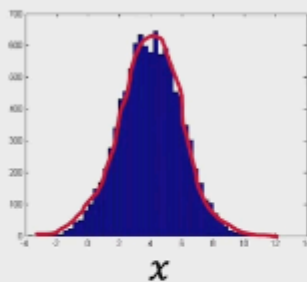
→ Diseases classification

⋮

when sufficient number of both normal and anomaly examples, use supervised learning (1:1).

Choosing/Manipulating Features For User:

Non-gaussian features



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

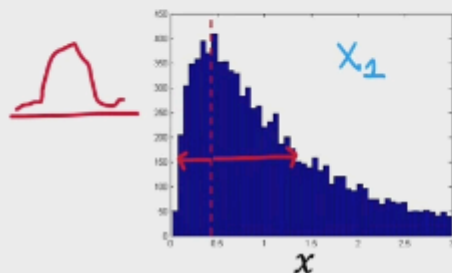
`plt.hist(x)`

$$x_1 \leftarrow \log(x_1)$$

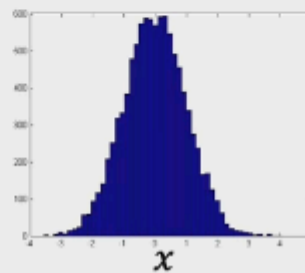
$$x_2 \leftarrow \log(x_2 + 1) \quad \log(x_2 + c)$$

$$x_3 \leftarrow \sqrt{x_3} = x_3^{1/2}$$

$$x_4 \leftarrow x_4^{1/3}$$



`np.log(x)`



Play with features/values.

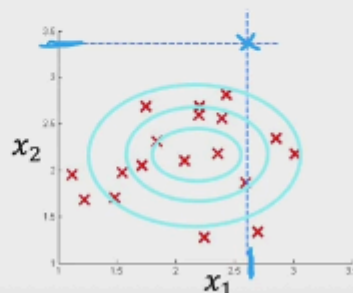
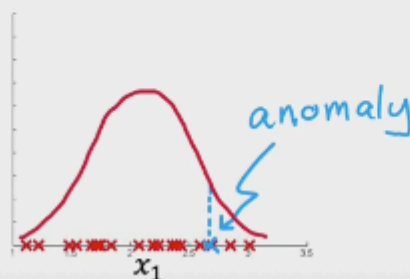
Error Analysis for Anomaly Detection

Error analysis for anomaly detection

Want $p(x) \geq \epsilon$ large for normal examples x .
 $p(x) < \epsilon$ small for anomalous examples x .

Most common problem:

$p(x)$ is comparable for normal and anomalous examples.
 ($p(x)$ is large for both)



x_1 numtransactions x_2 typing speed

Find additional features that when combined gives reasonable outs, when some features alone fail to explain/detect an anomaly.