

Unsupervised Learning, Recommenders, Reinforcement Learning

K-means algorithm

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

Repeat {

Assign points to cluster centroids

for $i = 1$ to m

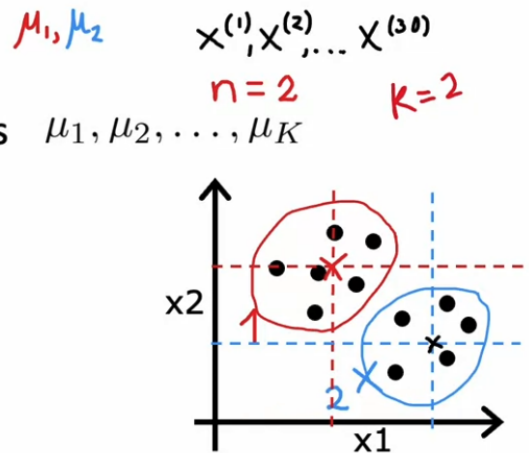
$c^{(i)} :=$ index (from 1 to K) of cluster
centroid closest to $x^{(i)}$

Move cluster centroids

for $k = 1$ to K

$\mu_k :=$ average (mean) of points assigned to cluster k

}



K-means optimization objective

$c^{(i)}$ = index of cluster (1, 2, ..., 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

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

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

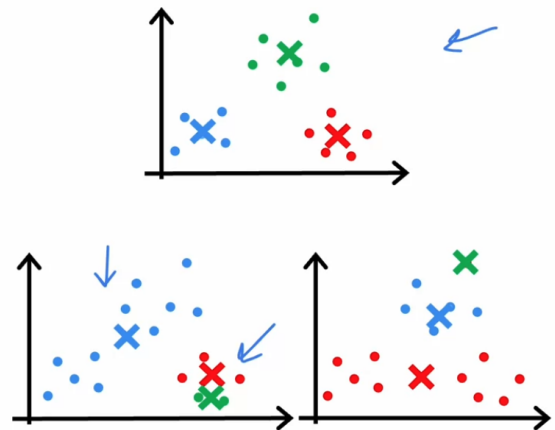
Distortion

Random initialization

Choose $K < m$

Randomly pick K training examples.

Set $\mu_1, \mu_1, \dots, \mu_k$ equal to these K examples.



With random initialization using some of the data points themselves, depending on which points you choose, we can end up with different clusters. Local minima of the cost function may be found. We can run it with multiple random initializations and then pick the one with lowest final cost function.

Random initialization

For $i = 1$ to 100 { 50-1000

Randomly initialize K-means. ← k random examples

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

Computer cost function (distortion)

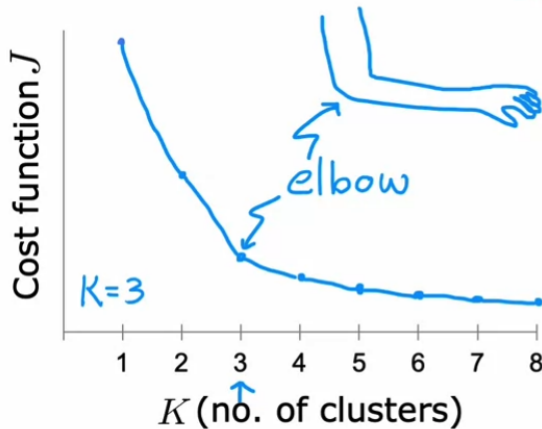
$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \mu_1, \dots, \mu_k)$ ←

}

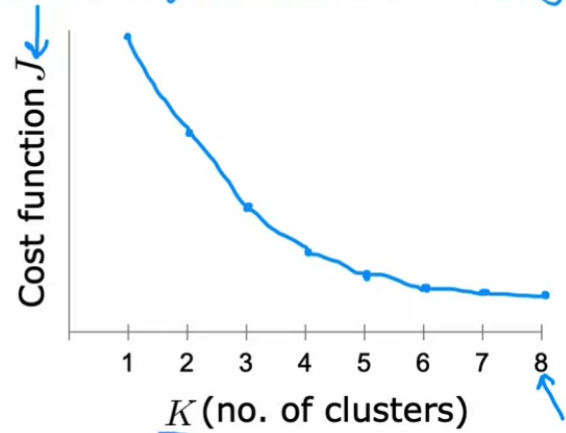
Pick set of clusters that gave lowest cost J

Choosing the value of K

Elbow method

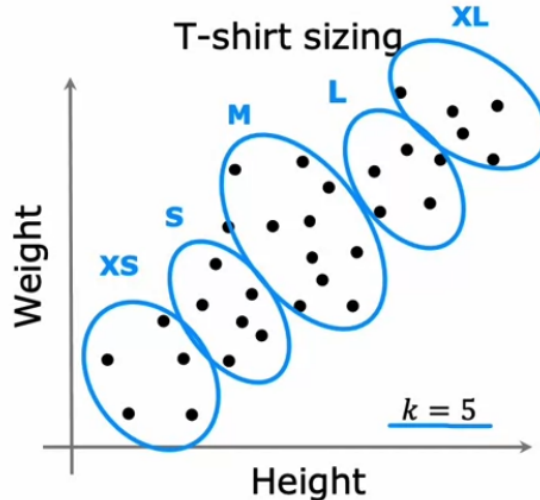
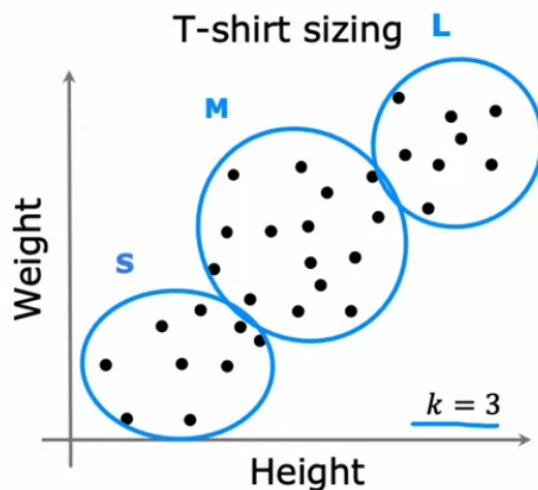


*the right "K" is often ambiguous
Don't choose K just to minimize cost J*



Choosing the value of K

Often, you want to get clusters for some later (downstream) purpose.
Evaluate K-means based on how well it performs on that later purpose.



Anomaly Detection

Majority of training examples are normal. Does the test example look like them?

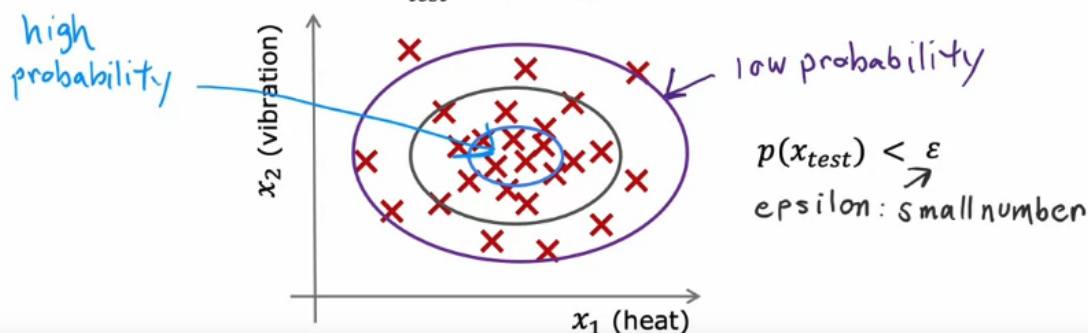
Uses: fraud (e.g. financial), manufacturing defects, computer monitoring in data centers

Density estimation

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ *probability of x being seen in dataset*

Model $p(x)$

Is x_{test} anomalous?



Density estimation

Training set: $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots, \vec{x}^{(m)}\}$
Each example $\vec{x}^{(i)}$ has n features

$$\begin{matrix} x_2 \\ \uparrow \\ \text{---} \end{matrix} \quad \begin{matrix} \text{---} \\ \uparrow \\ x_1 \end{matrix} \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$p(\vec{x}) = p(x_1; \mu_1, \sigma_1^2) * p(x_2; \mu_2, \sigma_2^2) * p(x_3; \mu_3, \sigma_3^2) * \dots * p(x_n; \mu_n, \sigma_n^2)$$

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

"add" "multiply"

$$p(x_1 = \text{high temp}) = 1/10$$

$$p(x_2 = \text{high vibra}) = 1/20$$

$$p(x_1, x_2) = p(x_1) * p(x_2)$$

$$= \frac{1}{10} * \frac{1}{20} = \frac{1}{200}$$

We fit a Gaussian distribution for each feature that could be indicative of anomaly. The intuition is that if any feature is too small or large, then it's likely an anomaly...

Anomaly detection algorithm

1. Choose n features x_i that you think might be indicative of anomalous examples.

2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

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

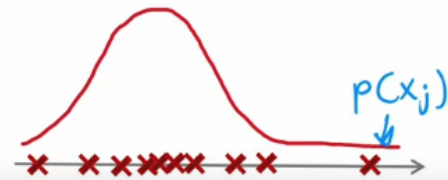
Vectorized formula

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

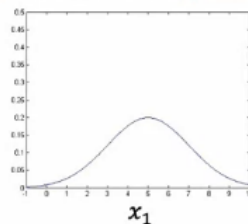
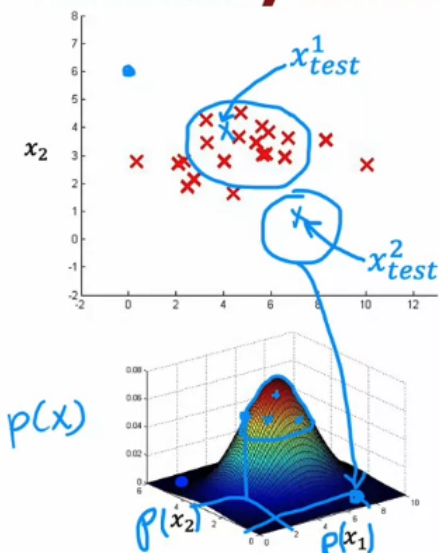
3. Given new example x , compute $p(x)$:

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

Anomaly if $p(x) < \epsilon$

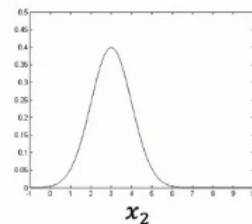


Anomaly detection example



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

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



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

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

$$\epsilon = 0.02$$

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

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

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.

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

Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

} include a few anomalous examples $y=1$ mostly normal examples $y=0$

Aircraft engines monitoring example

10000 good (normal) engines $y=0$ 2 to 50
20 flawed engines (anomalous) $y=1$
2

Training set: 6000 good engines $y=0$ 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

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

We use the cross validation to tune parameters of the model and the test set to see how well it's doing. If we have very few anomalous examples, one alternative is to only have a cross validation set, although the chances of overfitting are higher. The algorithm is still unsupervised since the training examples are unlabeled data. We use labels only on the cross validation to evaluate the model that was already fit. We can then get metrics on the cross validation and test sets. We can calculate TP, FP, FN, TN, precision, recall, f1 score, specially if the dataset is really skewed.

Anomaly detection vs. Supervised learning

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

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

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

⋮

Carefully choosing the right features is more important for anomaly detection than supervised learning.

- Pick Gaussian features or adjust non-gaussian features to gaussian (e.g. take the log or the sqrt)
- Error analysis (which anomalies is the algorithm not detecting?) to try and identify a new feature that would help it label the anomaly correctly