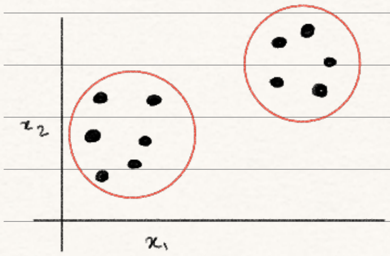
Subject:

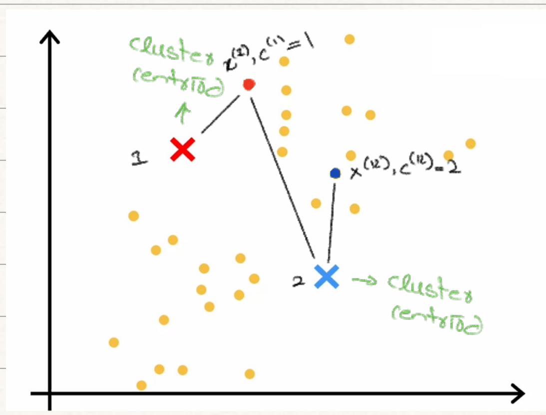
Clustering:



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

K-means clustering algorithm:

- Finds centres of clusters randomly



- Goes to each point see which point is closest to which centroid & the assign the point to that centroid.
- Take avg of red points & move red centroid to that avg.
  Does the same for blue cross.

k = cluster index

Algorithm:

- Randomly initialize k cluster centroids $\mu_1, \mu_2, \dots, \mu_k$

Repeat {

    # Assign points to cluster centroids

μ has same dimensions as x, training examples.

Subject:

for $i = 1$ to m:

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

$\min_k \|x^{(i)} - \mu_k\|^2$ ↳ L2 norm

\# Move cluster centroids

for $k = 1$ to k

μ_k = avg of points assigned to cluster k

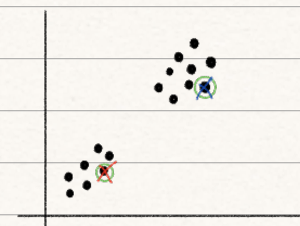
{

Optimization objective:

Cost function:

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

Initialization k-means:

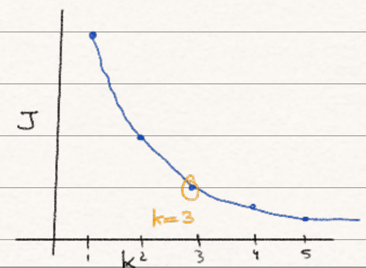


- Choose $k < m$
- Randomly pick k training examples
- Set $\mu_1, \mu_2, \dots, \mu_k$ equal to these k examples
- Run multiple times & choose best one by computing J of all & choose the set of clusters with lowest J.

50–1000 times

Choosing k:

Elbow method:



Subject: / /

- Don't choose k to minimize J cuz largest k would give smallest J.

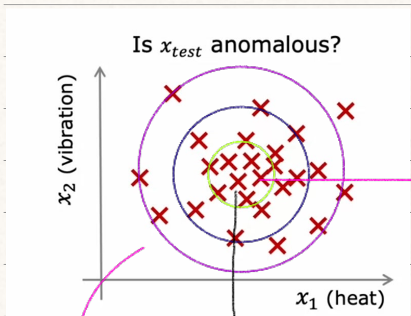
How to choose?

- Evaluate k-means based on how well it performs

Anomaly detection:

- Learns normal conditions & thereby detects anomalies

Density estimation:



$p(x_{test}) \geq \epsilon$

Ok!

High probability

lower as move to bigger rings

$p(x_{test}) < \epsilon$

↳ small number

flag as anomaly

Normal distribution:

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

Subject:

Anomaly Detection algorithm:

Density estimation:

$X : \{ \vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(m)} \}$
has n features

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

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

*

1) Choose n features x_i
2) Fit parameters

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

3) Given new x, compute $p(x)$

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

4) Anomaly if $p(x) < \epsilon$

Real number Evaluation:

- Assume we have some labeled data, of anomalous & non-anomalous examples.
  ($y = 1$) ($y = 0$)
- Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
- CV set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$ → tune ϵ, x_j
- Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$ → some $y=1$ but mostly $y=0$

NOTE:
if anomalous data is very less, use only CV and NO Test set

Subject:

Anomaly detection VS Supervised learning:

Anomaly detection	Supervised learning
• Very small (0–20) $y=1$ examples • Large $y=0$ examples • Many anomalies of different types	• Large no. of + & − examples • Future tev examples are similar to future ones

Choosing features:

• Use gaussian features

• Transform non-gaussian x to be gaussian

$x = \log(x)$

$x = \log(x + c)$ ↳ large c will not transform

$x = \sqrt{x}$

$x = \sqrt[3]{x}$

Error analysis:

• $p(x)$ is large for both normal & anomalous examples & thus model will fail to flag it.

• Use new feature that will help distinguish

```python
# Step 1: Estimate Gaussian parameters
mu, var = estimate_gaussian(X)

# Step 2: Calculate probability densities for the dataset
p = multivariate_gaussian(X, mu, var)

# Step 3: Determine the best threshold for anomaly detection
epsilon, F1 = select_threshold(y_val, p)

# Step 4: Identify anomalies in the dataset
anomalies = X[p < epsilon]
```