Overview

- Supervised vs Unsupervised Learning
- Clustering: $k$-means algorithm
• Training data: \( \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\} \)
• Training data is labelled i.e. we know \( y^{(1)}, y^{(2)} \) etc
Unsupervised Learning

- Training data: \( \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \)
- Training data is unlabelled i.e. we do not know \( y^{(1)}, y^{(2)} \) etc
- We need algorithms that try to cluster the training data ...
Applications

- Google News
Applications

- Fraud detection - try to cluster into normal and anomalous activity based on observed features
- Market segmentation e.g. try to detect customers about to leave a service
- Social network analysis e.g. try to detect communities/groupings
$k$-means algorithm
**k-means algorithm**

Input:

- $k$, number of clusters
- Training data: \( \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \)
- We’ll drop the \(x_0 = 1\) convention and use \(x_1, \ldots, x_n\) as elements of \(x\).

Randomly initialise \(k\) cluster centres \(\mu^{(1)}, \ldots, \mu^{(k)}\). e.g. choose \(k\) points from training set and use these (need \(k < m\)).

- Repeat:
  - **cluster assignment:**
    for \(i = 1\) to \(m\),
    \[ c^{(i)} := \text{index of cluster centres closest to } x^{(i)} \]
  - **update centres:**
    for \(j = 1\) to \(k\)
    \[ \mu^{(j)} := \text{average (mean) of points assigned to cluster } j \]
- Stop when assignments no longer change
**k-means algorithm: optimisation objective**

$c^{(i)} = \text{index of cluster to which example } x^{(i)} \text{ is assigned}$

$\mu_j = \text{centre of cluster } j$

$\mu_{c^{(i)}} = \text{cluster centre to which example } x^{(i)} \text{ is assigned}$

$\|x - c\|^2 = \sum_{j=1}^{n}(x_j - c_j)^2 \text{ (Euclidean distance)}$

**Goal:** minimise

$J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu^{(c^{(i)})}\|^2$
**k-means algorithm: optimisation objective**

Goal: minimise
\[
J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu^{(c^{(i)})}\|^2
\]

- Repeat:
  - **cluster assignment:**
    for \(i = 1\) to \(m\),
    \[c^{(i)} := \text{index of cluster centres closest to } x^{(i)}\]
    i.e. select \(c^{(1)}, \ldots, c^{(m)}\) to minimise \(J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)})\)
  - **update centres:**
    for \(j = 1\) to \(k\)
    \[
    \mu_j := \text{average (mean) of points assigned to cluster } j
    \]
    \[
    = \frac{1}{|C_j|} \sum_{k \in C_j} x^k
    \]
    i.e. select \(\mu^{(1)}, \ldots, \mu^{(k)}\) to minimise \(J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)})\) (a least squares task)

- Stop when assignments no longer change
**k-means algorithm: local optima**

- *k*-means algorithm can converge to a local optimum, rather than a global optimum. e.g.
**$k$-means algorithm: local optima**

Use random initialisation and multiple runs of algorithm:

for $i = 1$ to 100
  randomly initialise the $k$ centres $\mu^{(1)}, \ldots, \mu^{(k)}$
  run $k$-means algorithm
  compute cost function $J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)})$

Pick clustering that gives lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(jk)})$
**k-means algorithm**: choosing the number of clusters

**Elbow method:**
- Vary $k$ and pick value at “elbow”
- Problem: there might not be an elbow, or at least not a clear one
Cross-validation:

- Randomly select a subset of training data
- Run $k$ means algorithm
- Calculate cost $J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)})$ for the test data not used for training
- Repeat for multiple random subsets and several values of $k$. 