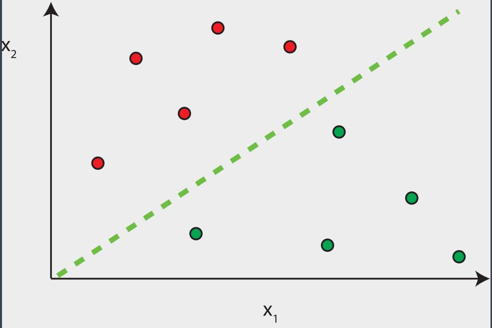
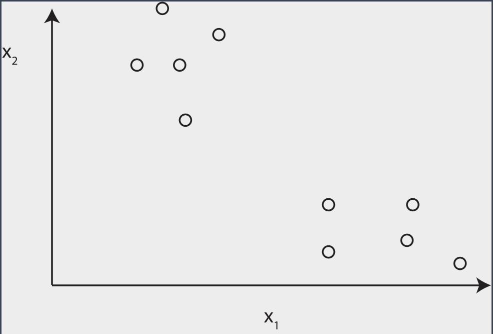
» Supervised Learning



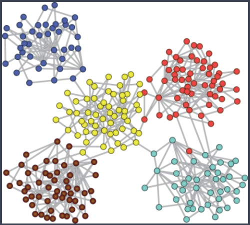
- Training data: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
- Training data is labelled i.e. we know $y^{(1)}$, $y^{(2)}$ etc

» Unsupervised Learning



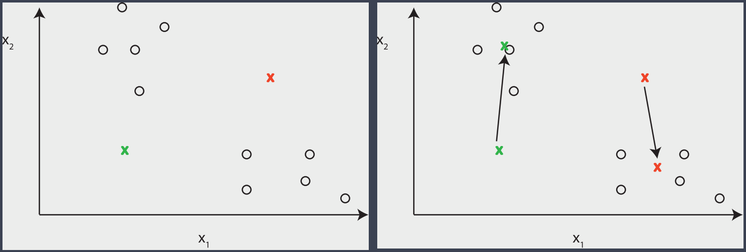
* Training data: $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
* Training data is unlabelled i.e. we do not know $y^{(1)}$, $y^{(2)}$ etc
* We need algorithms that try to cluster the training data ...

» Applications



* Social network analysis e.g. try to detect communities/groupings based on social graph
* News, Music e.g. try to cluster related news articles or songs
* Market segmentation e.g. try to cluster customers for targetted advertising

» *k*-means algorithm



» *k*-means algorithm: example

* Consider a 1D dataset with four examples: -3,-1,2,4. Suppose the initial cluster centres are -4 and 0 ...
* Round 1:
    * point -3 is assigned to centre -4, points -1, 2 and 4 are assigned to centre 0
    * update the centres to be the average of the assigned points, so the first centre becomes -3/1=-3 and the second centre (-1+2+4)/3=1.66
* Round 2:
    * points -3 and -1 are assigned to centre -3 and points 2 and 4 to centre 1.66
    * update centres to (-3-1)/2=-2 and (2+4)/2=3
* Round 3: points -3 and -2 are assigned to centre -2 and points 2 and 4 to centre 3. stop

» k-means algorithm

Input:

* k, number of clusters
* Training data: $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
* 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,
  $c^{(i)}$:= index of cluster centres closest to $x^{(i)}$
  update centres:
  for $j = 1$ to k
  $\mu^{(j)}$:= average (mean) of points assigned to cluster j
* Stop when assignments no longer change

» k-means algorithm: optimisation objective

$c^{(i)}$ = index of cluster to which example $x^{(i)}$ is assigned
μ_j = centre of cluster j
$\mu_{c^{(i)}}$ = cluster centre to which example $x^{(i)}$ is assigned
$\|x - c\|^2 = \sum_{j=1}^{n} (x_j - c_j)^2$ (Euclidean distance)

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

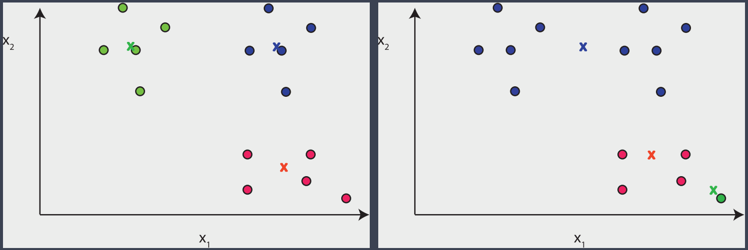
» k-means algorithm: optimisation objective

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

* Repeat:
  cluster assignment:
  for $i = 1$ to m,
    $c^{(i)}$:= index of cluster centres closest to $x^{(i)}$
    i.e. select $c^{(1)},\dots,c^{(m)}$ to minimise
  $J(c^{(1)},\dots,c^{(m)},\mu^{(1)},\dots,\mu^{(k)})$
  update centres:
  for $j = 1$ to k
    μ_j := average (mean) of points assigned to cluster j
      $=\frac{1}{|C_K|}\sum_{k\in C_j} x^k$ where $C_j = \{i : c^{(i)} = j\}$
    i.e. select $\mu^{(1)},\dots,\mu^{(k)}$ to minimise
  $J(c^{(1)},\dots,c^{(m)},\mu^{(1)},\dots,\mu^{(k)})$ (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.



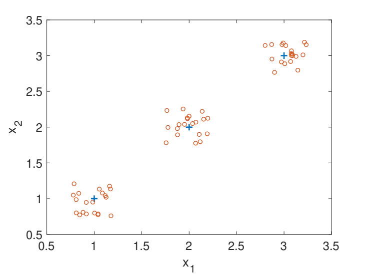
» *k*-means algorithm: local optima

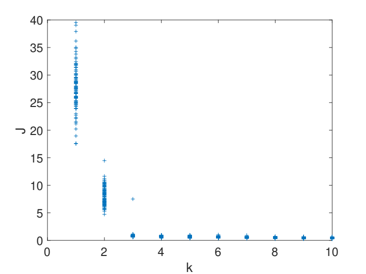
Use random initialisation and multiple runs of algorithm:

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





Cross-validation:

- Split data into test and training data (random splits of k-fold)
- run k means algorithm on training data
- Calculate cost $J(c^{(1)}, \ldots, c^{(m)}, \mu^{(1)}, \ldots, \mu^{(k)})$ for the test data not used for training
- Repeat for multiple splits and several values of k.

» Example: Clustering News Articles

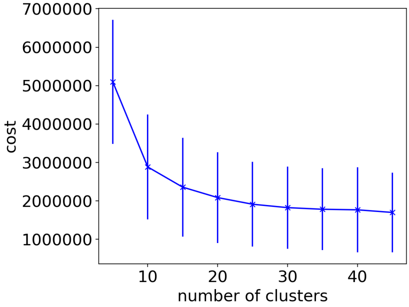
- Dataset: https://www.kaggle.com/snapcrack/all-the-news/home
- Fields: id, title, publication name, author, date, year, month, url, content. E.g. :
  - Rift Between Officers and Residents as Killings Persist in South Bronx - The New York Times
  - Tyrus Wong, 'Bambi' Artist Thwarted by Racial Bias, Dies at 106 - The New York Times
  - Among Deaths in 2016, a Heavy Toll in Pop Music - The New York Times
  - Kim Jong-un Says North Korea Is Preparing to Test Long-Range Missile - The New York Times
- We'll use 50,000 articles from articles1.csv
- Remove stop words, use steming, use bag of words model to map text to feature vector

» Example: Clustering News Articles

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold
import math

text = pd.read_csv('articles1_1000.csv')
text_content = text['content']
tfidf = TfidfVectorizer(stop_words = 'english', max_features=500, max_df=0.5).fit_transform(text_content).toarray()
K = range(5, 50, 5)
SSE_mean = []; SSE_std=[]
for k in K:
        gmm = KMeans(n_clusters=k)
        kf = KFold(n_splits=5)
        m=0; v=0
        for train, test in kf.split(tfidf):
                gmm.fit(train.reshape(-1, 1))
                cost=-gmm.score(test.reshape(-1, 1))
                m=m+cost; v=v+cost*cost
        SSE_mean.append(m/5); SSE_std.append(math.sqrt(v/5-(m/5)*(m/5)))
plt.errorbar(K, SSE_mean, yerr=SSE_std, xerr=None, fmt='bx-')
plt.ylabel('cost'); plt.xlabel('number of clusters'); plt.show()

k = 15
gmm = KMeans(n_clusters=k).fit(tfidf)
centers = gmm.cluster_centers_.argsort()[:,::-1]; terms = vector.get_feature_names()
for i in range(0,k):
    word_list=[
    for j in centers[i,:25]:
        word_list.append(terms[j])
    print("cluster%d:"% i); print(word_list)

labels = gmm.predict(tfidf); count = 0
print('\nsimiliar articles:')
for j in range(0,labels.shape[0]):
        if labels[j]==0:
                print('\n'+text['title'].iloc[j])
                count = count+1
                if (count>=5):
                        break
```

» Example: Clustering News Articles

Choose $k = 15$:



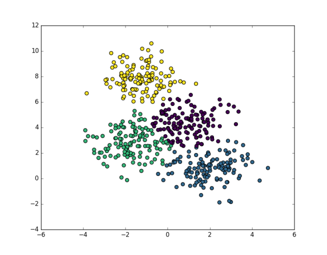
» Example: Clustering News Articles

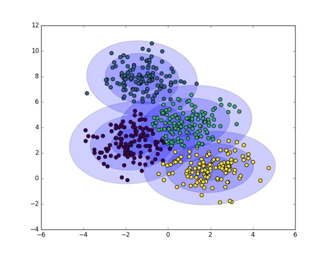
Typical output:

```
cluster 0:
['russia', 'russian', 'intelligence', 'news', 'election', 'united', 'agencies', 'officials', 'information', 'report', 'american', 'campaign', 'clinton', 'government', 'committee', 'obama', 'friday', 'media', 'political', 'turkey', 'agency', 'democratic', 'national', 'democrats', 'evidence']
cluster 1:
['ms', 'family', 'mother', 'husband', 'school', 'children', 'york', 'life', 'times', 'film', 'daughter', 'news', 'home', 'told', 'father', 'women', 'work', 'house', 'food', 'later', 'day', 'love', 'public', 'education', 'job']
cluster 2:
['judge', 'court', 'supreme', 'justice', 'law', 'order', 'case', 'death', 'legal', 'federal', 'administration', 'lawyers', 'washington', 'democrats', 'senate', 'republicans', 'united', 'wrote', 'right', 'state', 'executive', 'government', 'ban', 'cases', 'white']
...
similiar articles:
After The Biggest Loser, Their Bodies Fought to Regain Weight - The New York Times
Work. Walk 5 Minutes. Work. - The New York Times
Is Your Workout Not Working? Maybe You're a Non-Responder - The New York Times
Scientists Say the Clock of Aging May Be Reversible - The New York Times
Light Pillars, a Million-Mirror Optical Illusion on Winter Nights - The New York Times

similiar articles:
N.F.L. Playoffs: Schedule, Matchups and Odds - The New York Times
A 'World Unto Itself' in New York Area Yeshivas: Floor Hockey - The New York Times
5 Must-See Shows if You're in New York This Month - The New York Times
Broadway Breaks Multiple Records Through New Year's Weekend - The New York Times
Brock Osweiler and Texans Knock the Battered Raiders Out of the Playoffs - The New York Times
```

» Summary

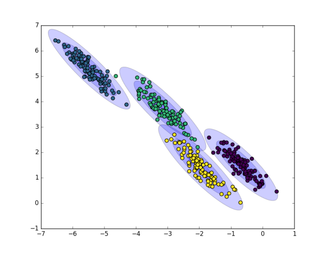
- k-means algorithm is straightforward, popular, often works pretty well
- Two situations where it performs less well:
  - Clusters overlap i.e. an item can be a member of more than one cluster simultaneously → we've looked at "hard" k-means but easy extension to soft k-means

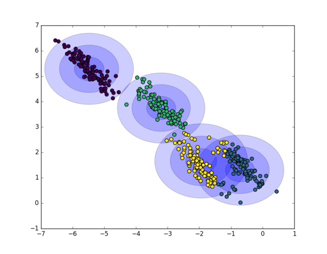




» Summary

* Some situations where it performs less well:
  * Clusters are not roughly spherical e.g. if long and narrow → extend *k*-means to estimate shape of cluster





» More on ML

* Get a decent GPU and try to implement deep learning object detection etc - you have all the tools you need for this already
* Deep Learning for text analytics/natural language processing e.g. BERT
* Time series, incl. recurrent neural nets
* Recurrent neural nets for session-aware recommenders (special case of time-series)
* Explainable models incl. visualising deep learning models
* Ensemble methods: bagging and boosting
* Online learning: exploration/exploitation, multi-arm bandits, reinforcement learning
* Privacy-aware machine learning: k-anonymity, differential privacy
* Scalable, fast optimisation methods. GPU programming.