UNSUPERVISED LEARNING

35.1
→ What is clustering?

classification & Regression.

$\mathcal{D} = \{x_i, y_i\}$ ← Predict y_i given x_i

$y = f(x)$

$y_i \in \{0, 1\}$ → 2-class classification.

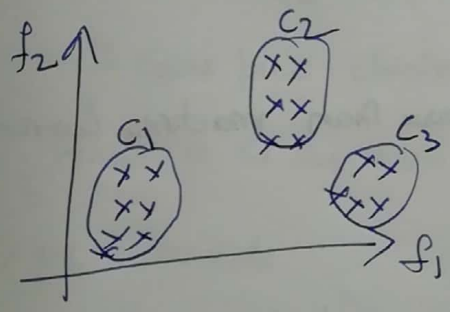
$y_i \in \mathbb{R}$ → regression.

In clustering

$\mathcal{D} = \{x_i\}$ → no y_i's

Task: group/cluster similar data points

↳ points are close together & far from other points away from other points



(a) Point in a cluster are close together

(b) Points in diff clusters are far away

Similar: is problem specific

$\mathcal{D} = \{x_i\}$ no y_i's

Q) How do you measure how well your algo. performed?

↳ well will know later

Clustering: K-means, Hierarchical clustering, DBSCan.

35.2 Unsupervised learning

→ Clustering is always referred to as unsupervised learning

→ In case of classification & regression

$\{x_i, y_i\}$ we will a function $f(x) = y$

Here y_i is supervising / helping to find function $f(x)$

→ Semi-supervised learning : $D = D_1 \cup D_2$

$D_1 \rightarrow \{x_i, y_i\}$ $D_2 \rightarrow \{x_i\}$

Length of $|D_1| << |D_2|$

35.3 Applications

→ Deeply studied in Data mining course than machine learning

→ Tons of application

https://en.wikipedia.org/wiki/Cluster_analysis#Applications

Ex : (1) eCommerce : Amazon.com, Alibaba, Ebay, Flipkart

Task : group "similar" customers based on their purchasing behaviour

C1, (C2), C3, (C4), C5

we can offer a discount (10%, 20%)

we can offer a deal

{money, credit card, product, geographical of customer.} {estimate income per year}

Eg (2) Image segmentation (computer vision & image processing)

↳ grouping / clustering similar pixels into regions / segments

↳ Typical apply ML algo to begin obj detection

eg:3 Amazon Food Reviews → class label.

review & text	Polarity (+ve, -ve)
r_1	+ve
r_2	-ve
r_3	+ve
⋮	⋮

(Manually reviewed & labelled)
↳ Time Consuming
↳ Costly / Expensive

Imagine we have 1M reviews (r_i's)

① cluster them into 10K groups using word-similarity (BoW, tfidf, w2v)
↓
$c_1, c_2, c_3 \ldots c_{10K}$

② review & label each cluster.

I will pick cluster c_i → pick one pt → true true . All the points in c_i as +ve
↓
In real world (365 points)

③ ML- models

35.4 Metrics for clustering

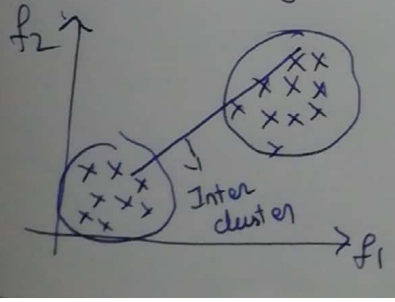
$D = \{x_i\}$; no y_is

what are the performance measures ?

[classn & regression → AUC, log-loss, Pre, Recall

y_i's ← class label / regression label { ground truth }]

geom: what is a clustering result :



$D = \{x_i\}$
$x_i \in \mathbb{R}^2$

→ 2 clusters.
→ Intra cluster : within a cluster
→ Inter cluster : b/w clusters

In a ideal world

↳ Inter-cluster dist → V- high

↳ Intra-cluster dist → V. low

Dunn Index:

$D = \uparrow \max_{i,j} d(i,j)$ → dist c_i, c_j

K - clusters.

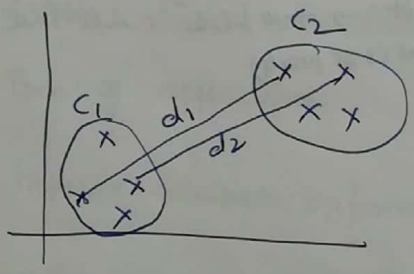
$\{c_1, c_2 \cdots c_i, c_j, c_k\}$

$\downarrow \max_k d'(k)$ ↑ Intra cluster distance

↳ maximal inter cluster distance

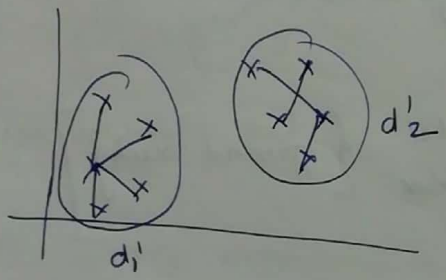
D is high ⇒ good clustering

$d(i,j)$ = dist b/w c_i & c_j farthest point.



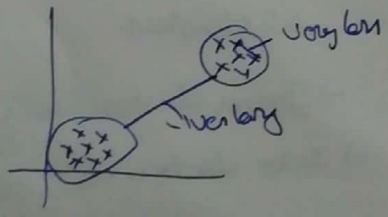
for every pair : $d_1, d_2, d_3 \cdots d_k$

Pick the max distance point



$\max(d_1', d_2')$

Ideal dust:



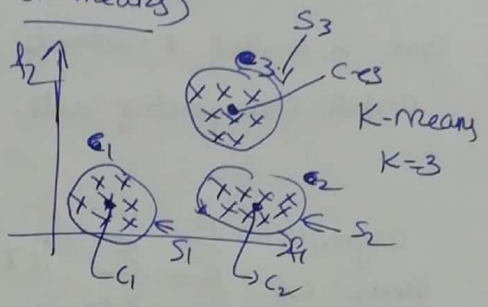
→ many other measures / metrics for measuring clustering

35.5 Centroids, Geometric Intuition (K-means)

→ popular, simple

→ Basic K-means.

→ Variants of K means.



- K: # of clusters.
- hyper parameter → (CV) ideas (ik)

c_1, c_2, c_3 : centroids

S_1, S_2, S_3 ← sets

$S_1 \cap S_2 = \phi$ (null set)

$S_1 \cap S_3 = \phi$

$S_2 \cap S_3 = \phi$

→ when we say we want k-clusters we will get k-centroids & k-sets

K-clusters : 1 2 3 - - - K

K-centroids = $c_1, c_2, c_3 \dots c_K$

K-sets : $S_1, S_2, S_3 \dots S_K$

$$c_i = \frac{1}{n} \sum_{x_j \in S_i} x_j \quad \leftarrow \text{mean point of } S_i$$

K-means : centroid based clustering scheme

Big challenge : How to find k-centroids, once we get centroids we can easily find the sets by using nearest centroid idea.

35.6 Mathematical Formulation : objective function (K means)

$\mathcal{D}$: $\{x_1, x_2 \cdots x_n\}$

Task is to find K-Centroids : $c_1, c_2 \cdots c_k$ $\quad \forall\, x_i \in S_j$

and its corresponding sets : $S_1, S_2 \cdots S_k$ $\quad \forall_{i,j}\ S_i \cap S_j = \phi$

$$\underset{c_1, c_2 \cdots c_k}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2 \quad \text{s.t} \quad x \in S_i, \ \forall_{ij}\ S_i \cap S_j = \phi$$

(Intra cluster distance minimized; $\|x - c_i\|^2$: Sq. dist of x from c_i)

$\{c_1, c_2 \cdots c_k$ → using proximity idea

$S_1\ S_2 \cdots S_k$

→ In the above function there is no Intra cluster distance

→ V.V. hard to solve the above Equation.

↳ NP-hard

↳ Exponential time Complexity

In Computer Science : If we have a hard problem

↓

approximation algo → using some hacks

→ Lloyd's algo

35.7 K-means Algorithm

Lloyd's algo

Step 1 : Initialization : (random initialization)

→ randomly pick k pts from $\mathcal{D}$ and call them.

$c_1, c_2, \cdots c_k$

Step 2 : Assignment

For each pt x_i in $\mathcal{D}$

→ Select the nearest c_j

→ Compute the dist (x_i, c_j) $\forall\ j = 1, 2, \cdots k$

→ add x_i to set S_j corresponding to centroid c_j

→ we will have a point x_i assigned to a set S_j $\quad j = 1, 2, \ldots k$ $\quad i = 1$ to n.

Step3 : Recompute centroids

→ recalculate/update c_j's as follows

$$c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

$|S_j|$ → # of points in set S_j

$\sum_{x_i \in S_j} x_i$ → mean-pt

Step4 : Repeat Step 2 & Step 3 until convergence

Step 2 ↓ assignment

Step 3 ↓ Recalculate

What is Convergence ?

Centroids don't change much

Old Centroid = $\{c_1, c_2 \ldots c_k\}$

new centroid = $\{c_1', c_2' \ldots c_k'\}$

$c_1 - c_1'$, $c_2 - c_1'$, $c_k - c_k'$

distance b/w new-centroid & old centroid is small

35.8: How to Initialize K-means++

Lloyd's algo + Initialization stage

↓

random ← random: pick K-pts randomly from D

Initialization ↓

$c_1, c_2, c_3 \cdots c_k$

⇒ K-means has a problem called initialization sensitivity

→ final clusters + centroids are very much dependent on initialization fo

How to deal with this problem

① Repeat K-means multiple times with different initializations

↳ Pick the best clustering based on

- smaller intra cluster distance
- large inter cluster distance

② K-mean++

↳ Instead of random-init → it uses smart initialization

Initialization in K-means: (Task) pick $c_1, c_2, \cdots c_k$

① Pick the first centroid randomly → c_1 from D

② $\forall x_i \in D$ create a distribution as follows

dist

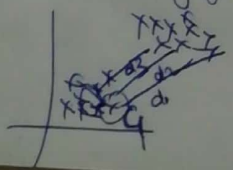
$x_i \longrightarrow$ dist (xi, nearest centroid) $\Rightarrow \|(x_i - c_1)\|^2$

x_1	d_1
x_2	d_2
x_3	d_3
⋮	⋮
x_n	d_n

pick a pt from $D - \{c_1\}$ with a prob. prop to d_i

→ This is called a probabilistic approach.

→ Pick a point with large distance, because it is farther away from the centroid c_1



(Q) Why do this probabilistically?

K-mean :- does get effected with outliers

35:9 Failure cases / limitations

→ K-means has problems when clusters are of diffent
* Sizes
* Densities
* Non-globular shapes (Non-Convex) 

→ k-means has problems when dataset has outliers.

Solution for all above problems is to increase the "k"

35:10 K-Medoids

Problem K-means :- → Centroids may not be interpretable

$C_1, C_2, \cdots C_k$

$\mathcal{D} : \{x_1, x_2 \cdots x_n\}$

→ We will get data point instead of centroids

Partitioning around ~~methods~~ medoids (PAM) : K-medoids.

① Initialization :- K-means++ → Probabilistic method pick k pts from $\mathcal{D}$

② Assignment :- closest medoid → same as in k mean.

$\{ x_i \in S_j$ if medoid is the closest medoid to x_i

③ Update / recompute

k mean :- $c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$

K-Medoids (PAM) :- (a) swap each medoid with a non medoid point
(b) If loss decrease keep the swap & undo the swap

loss in k-means

J_1

(min) $$\sum_{i=1}^{k} \sum_{x \in S_j} \|x - m_j\|^2$$

m_j → medoid j

m_1 x_1, x_2, x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} (m_2)

(a) loss value

$x_1 = m_1$; $x_6 = m_2$ → loss → l_1

(b) swap

$M_1 = x_2$

$m_2 = x_6$

↳ Compute loss value

$m_1 = x_2$

$m_2 = x_6$ } → loss – l_2

If $l_2 < l_1$

$m_1 = x_2$

$m_2 = x_6$

Else

$m_1 = x_1$

$m_2 = x_6$

K-modoids

↳ Interpretability

↳ kernalization ; sim; dist

35.11 Determining the right 'k'

↑ hyper-parameter

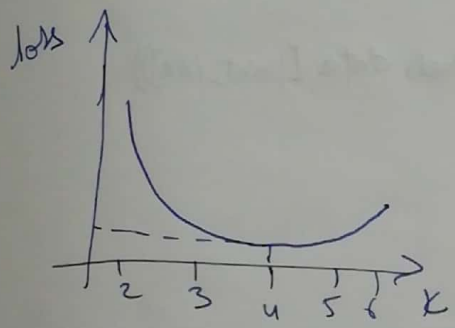
① domain - knowledge : Food reviews

↳ +ve & -ve

(2 clusters)

Elbow method (or) knee method

$$\text{loss} = \sum_{i=1}^{k} \sum_{x \in S_i} \|x_i - c_i\|^2 \rightarrow \text{minimize.}$$



best k is k=4

35.12 Code Samples

Sklearn.cluster. KMeans (n_cluster= 8, init = 'k-means++', n_init= 10

max_iter =300, tol = $1e^{-4}$,)

K-medoids

```
from sklearn.metrics.pairwise import pairwise_distances
import numpy as np
import kmedoids
data =
D= pairwise_distances(data, metric = 'euclidean')
# Split into 2 clusters
M, C = kmedoids.KMedoids(D, 2)
```

```
print ('medoids')
for point_idx in M:
    print (data[point_idx])
print ('')
print ('clustering result:')
for label in C:
    for pointidx in C[label]:
        print ('label {0}: {1}'.format (label, data [point_idx]))
```

35:13 Time & Space Complexity

K_means : $O(n \times k \times d_i \times \#\text{iterations})$

n → #pts, k → #clusters, d → dim

Typically $k \leq 10$

$i \leq 100$

$\approx O(nd)$ → linear time complexity

Space : $O(nd + kd)$

$\approx O(nd)$

↑ Linear

* k-means is quite fast