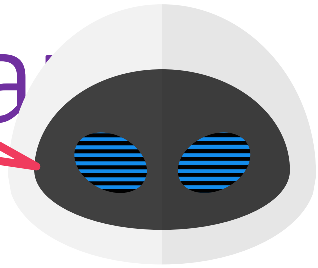
Clustering

CS771: Introduction to Machine Learning

Purushottam Kar

Ho xa

Grouping similar objects together is the task of clustering



gradescope

Q1. $\int x = ?$

5 marks

$\frac{x^2}{2} + b$

$x^2/2$

$x^2 + c$

5/5

$x^2 + d$

$x^2/2$

4/5

$\frac{x^2}{2} + c$

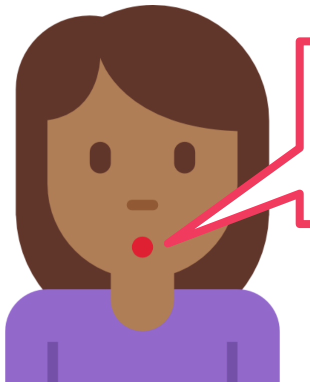
$x^2 + a$

$x^2 + e$

2/5

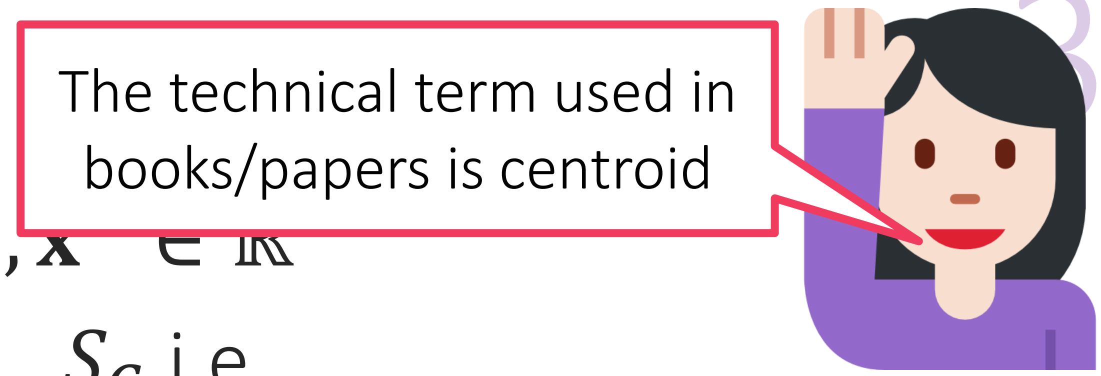
$x^2/2$

$x^2/2$



No labels here – that is why clustering is often called an *unsupervised* learning problem

Clustering



Given a set S of n data points $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n \in \mathbb{R}^d$

Split this set into C disjoint *clusters* $S_1, \dots S_C$ i.e.

Assign every data point $\boldsymbol{i}$ *to one of the subsets, say* $z_i \in [C]$ *(note that every data point is assigned to exactly one cluster) so that*

Data points assigned to the same subset are "similar" to each other, e.g.

If $z_i = z_j = c$ *for some* $c \in [C]$ *then* $\left\|\mathbf{x}^i - \mathbf{x}^j\right\|_2$ *is small*

The K-means problem asks this problem a bit differently

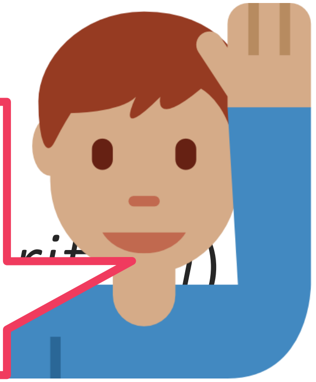
Split S into C *clusters* $S_1, \dots S_C$ *and find a* ~~*prototype*~~ centroid *for each cluster i.e.* $\boldsymbol{\mu}^c \in \mathbb{R}^d$ *s.t. if* $\mathbf{x}^i$ *is assigned to cluster* $\boldsymbol{c}$ *i.e.* $z_i = c$, *then* $\left\|\mathbf{x}^i - \boldsymbol{\mu}^c\right\|_2^2$ is small i.e. $\mathbf{x}^i$ *is close to* ~~*prototype*~~ centroid *of its cluster*

K-means clustering

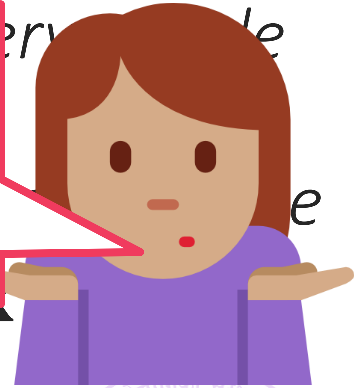
$$\min_{\{\boldsymbol{\mu}^c \in \mathbb{R}^d\},\{z_i \in [C]\}} \sum_{c=1}^{C} \sum_{i:z_i=c} \left\| \mathbf{x}^i - \boldsymbol{\mu}^c \right\|_2^2$$

K-MEANS/L

1. Initialize m
2. For $i \in [n]$, update z_i using $\{\boldsymbol{\mu}^c\}$
   1. Let $z_i = \arg\min_c \left\| \mathbf{x}^i - \boldsymbol{\mu}^c \right\|_2^2$
3. Let $n_c = \#$ points assigned to c
4. Update $\boldsymbol{\mu}^c = \frac{1}{n_c} \sum_{i:z_i=c} \mathbf{x}^i$
5. Repeat until convergence



The Lloyd's algorithm offers *monotonic progress.* It always lowers (or keeps the same) the value of $\sum_{c=1}^{C} \sum_{i:z_i=c} \left\| \mathbf{x}^i - \boldsymbol{\mu}^c \right\|_2^2$ i.e. the objective function in each iteration. Can you show this?



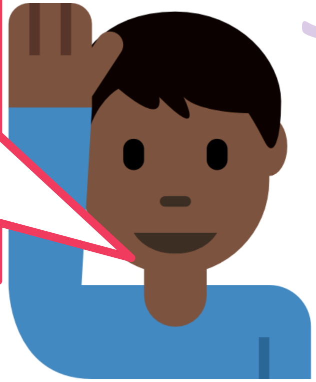
This looks a bit like coordinate minimization where we fix all but one coordinate and update that one coordinate to its optimal value

hat optimal value of $\boldsymbol{\mu}^c$ is simply the

True, coordinate minimization can be thought of as a special case of alternating optimization ☺

K-means++

Note that a k-means++ always initializes centroids as actual data points. Also, no data point can be selected twice – if a data point $\mathbf{x}^i$ gets selected once, then for all subsequent iterations, we will have $d_i = 0 = p_i$



Initializes k-means

Provable guarant

Widely used in practice: especially beneficial if k is large

K-MEANS++ INITIALIZER

1. Select first centroid randomly
   $\boldsymbol{\mu}^1 = \mathbf{x}^i$, where $i \sim \text{UNIF}([n])$
2. For $j = 2, \ldots, k$
   1. For all $i \in [n]$, calculate $d_i = \min_{l \in 1,\ldots,j-1} \left\| \mathbf{x}^i - \boldsymbol{\mu}^l \right\|_2$
   2. Set $\boldsymbol{\mu}^j = \mathbf{x}^i$ where i is chosen with probability $p_i = \frac{d_i^2}{\sum_{s=1}^n d_s^2}$

Some applications of clustering

Can be used to make LwP a more powerful algorithm

Learn more than one prototype per class e.g. k prototypes by clustering data of each class into k clusters and using the centroids returned by the clustering algorithm as prototypes

A test point is assigned the class of its closest prototype

Note: this will increase training time, test time, and model size a bit

Seamlessly gives us the 1NN algorithm if we demand as many clusters (and hence as many centroids) as there are data points

Some applications of clustering

Identify *subpopulations* in data and improve ML performance

- ***Example**: have data for 1M customers but don't know age/gender*
- *However, we suspect that age/gender significantly affects behaviour*

Instead of running an ML algo (say SVM) on entire training data, first cluster training data and run ML algo separately on each cluster

- *If k clusters then k models will get learnt. For test data points, first find to which cluster they belong (using distance to centroid) and use that model*
- *Increases model size and test time a bit but may increase accuracy too!*
- *If we cluster these customers according to their onsite behaviour (which items did they view/like/buy), possible that we may accidentally discover gender/age groups within our data without knowing these details directly*
- *Groups may not be perfectly clean but should improve ML performance*

Some applications of clu

Features for features!!

Reduce number of features (also called *dimensionality reduction*)

Example: have 1M featu ... *many of*
these features are redu

Example: synonyms in b

Can cluster features tog

Once we have $\hat{d}$ feature clusters, say $C_1, \ldots, C_{\hat{d}}$, we can create $\hat{d}$ new features, e.g. by taking average of features within each cluster i.e. for each data point $\mathbf{x} \in \mathbb{R}^d$, create a new feature vector $\tilde{\mathbf{x}} \in \mathbb{R}^{\hat{d}}$ where $\tilde{\mathbf{x}}_l = \frac{1}{|C_l|} \sum_{j \in C_l} \mathbf{x}_j$ for all $l \in [\hat{d}]$

To do this, we first need

Method 1: represent feature j using values it takes on the n train data

$\mathbf{z}^j = [x_j^1, x_j^2, \ldots, x_j^n] \in \mathbb{R}^n$ *for a*

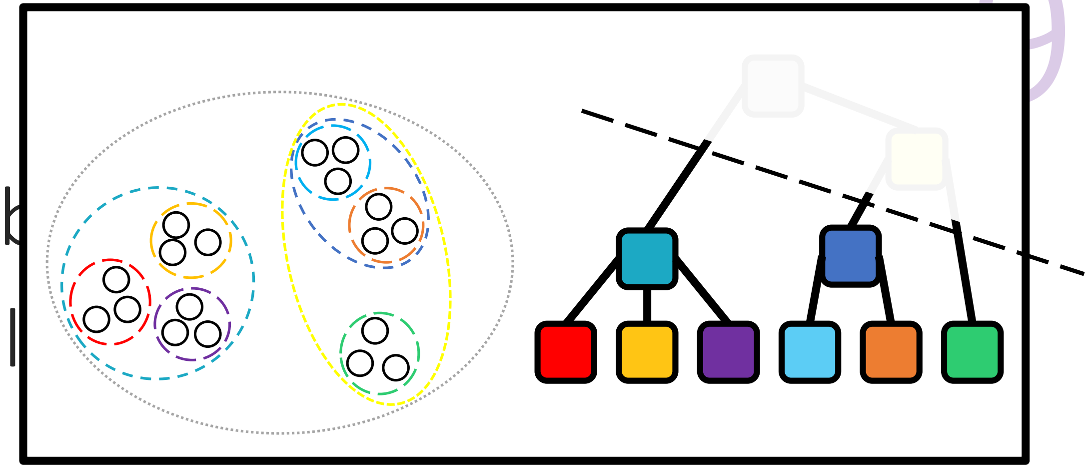
Method 2: represent feature j u ... *ch*

Let n_c *denote number of train d*

This trick is often called *feature clustering or feature agglomeration* and is a form of dimensionality reduction. Will see other dimensionality reduction techniques later

$\mathbf{z}^j = \left[\frac{1}{n_1}\sum_{i:y^i=1} \mathbf{x}_j^i, \frac{1}{n_2}\sum_{i:y^i=2} \mathbf{x}_j^i, \ldots, \frac{1}{n_C}\sum_{i:y^i=C} \mathbf{x}_j^i\right] \in \mathbb{R}^C$ *for all* $j \in [d]$

Variations in clustering



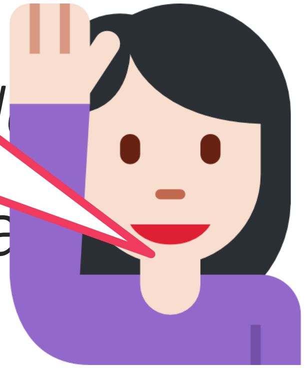
Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May

Aggl

Notice that imposing (resp. encouraging) balance among the clusters can be seen as a form of a constraint (resp. regularization) on clustering. The k-medoid problem also can be seen as a form of *constrained* clustering



Might
– clustering with *Bregman divergences*

Several other problem variants known e.g. k medoids (uses general $d(\mathbf{x}^i, \boldsymbol{\mu}^c)$ instead of $\left\|\mathbf{x}^i - \boldsymbol{\mu}^c\right\|_2^2$ and $\boldsymbol{\mu}^c$ must be one of the data points), *soft* k-means (a data point can belong to multiple clusters)

K-medoids preferable when centroids/prototypes must be real data points