Clustering

CS771: Introduction to Machine Learning

Purushottam Kar

Recap of Last Lecture

Regularization and various techniques to perform regularization

Adding a regularizer (L1/L2), early stopping, adding noise

Multiclassification

Using kNN, DTs, output codes

By converting to several binary classification problems – OVA

Crammer-Singer loss, softmax loss

Clustering

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

Split this set into C disjoint *clusters* $S_1, \ldots S_C$ 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.

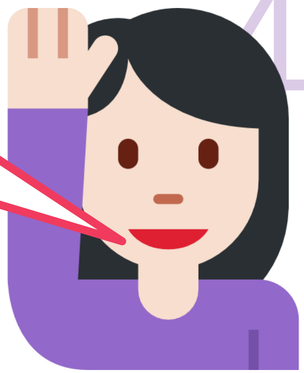
If $z_i = z_j = c$ for some $c \in [C]$ then $\left\|\mathbf{x}^i - \mathbf{x}^j\right\|_2$ is small

The K-means problem asks this problem a bit differently

Split S into C *clusters* $S_1, \ldots S_C$ *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*

Clustering

The technical term used in books/papers is centroid



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

Split this set into C disjoint *clusters* $S_1, \ldots S_C$ 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* $S_1, \ldots S_C$ *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

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

This optimization problem is NP hard to solve ☹

Popular heuristic: *Lloyd's algorithm* (often called *k-means algorithm*)

Uses a technique called alternating minimization

Observation 1: *if we fix all $\boldsymbol{\mu}^c$, obtaining optimal assignments z_i is very simple*

Assign each data point to the cluster whose centroid is closest!

Observation 2: *if we fix all assignments z_i, obtaining optimal centroid simple*

$\min_{\{\boldsymbol{\mu}^c \in \mathbb{R}^d\}} \sum_{c=1}^{C} \sum_{i:z_i=c} \left\| x^i - \boldsymbol{\mu}^c \right\|_2^2$ -- *all that is needed is* $\min_{\boldsymbol{\mu}^c} \sum_{i:z_i=c} \left\| \mathbf{x}^i - \boldsymbol{\mu}^c \right\|_2^2$

Apply first order optimality to deduce that optimal value of $\boldsymbol{\mu}^c$ is simply the average of all data points assigned to the cluster c – repeat for all $c \in [C]$

Keep repeating these two steps again and again

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

K...a

$\min_{\{\boldsymbol{\mu}^c\in\mathbb{R}^d\},\{z_i\in[C$ ... $\sum^C$ ... $\sum$ ... $\|$ ... $c\|^2$

This optimizatio...

Popular heuristi... ...ns algorithm)

Uses a techniqu...

Observation 1: ... *s z_i is very simple*

Assign each da... ... *st!*

Observation 2: ... *centroid simple*

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

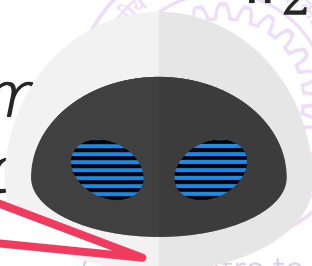
K-MEANS/LLOYD'S ALGORITHM

1. Initialize means $\{\boldsymbol{\mu}^c\}_{c=1\ldots C}$
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

Apply first order optim... ... *im...*

average of all data poi...

Keep repeating these t...



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

K-means++ Initializer

Initializes k-means with centroids that are well spread out

Provable guarantees: Arthur and Vassilvitskii, SODA 2007

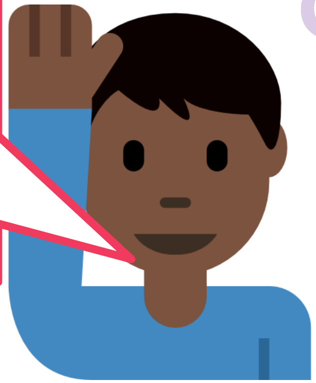
Widely used in practice: especially beneficial if k is large

K-MEANS++ INITIALIZER

1. Select first prototype randomly
   $\boldsymbol{\mu}^1 = \mathbf{x}^i$, where $i \sim \mathrm{UNIF}([n])$
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 $p_i = \frac{d_i^2}{\sum_{s=1}^n d_s^2}$

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 prototype randomly
   $\boldsymbol{\mu}^1 = \mathbf{x}^i$, where $i \sim \mathrm{UNIF}([n])$
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 behavior

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 clustering

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

Example: have 1M features i.e. $\mathbf{x}^i \in \mathbb{R}^d$ *for* $d = 1M$ *but we suspect many of these features are redundant or encode similar information*

Example: synonyms in bag of words ("buy" vs "purchase")

Can cluster features together into $\hat{d} \ll d$ *clusters*

To do this, we first need a way to represent features themselves

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

$\mathbf{z}^j = \left[\boldsymbol{x}_j^1, \boldsymbol{x}_j^2, \ldots, \boldsymbol{x}_j^n\right] \in \mathbb{R}^n$ *for all j* $\in [d]$

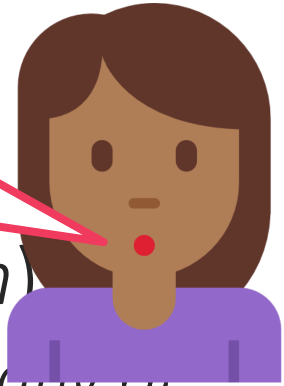
Method 2*: represent feature j using avg value it takes on points of each class*

Let n_c *denote number of train data points that belong to class* $c \in [C]$

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

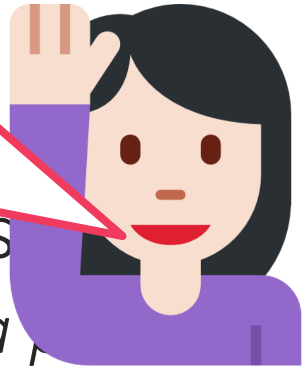
Some applications of clu

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



Features for features!!

Once we have $\hat{d}$ clusters of the features, say $C_1, \ldots, C_{\hat{d}}$, we can create $\hat{d}$ new features, for example by taking average of features within each cluster i.e. for each old 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 $l \in [\hat{d}]$

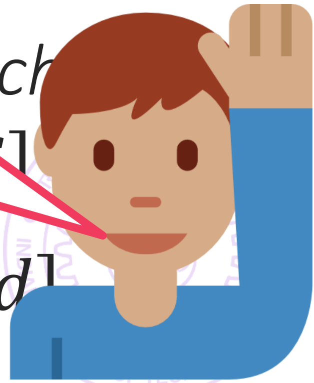


To

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 all* $j \in [d]$

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 – balanced clustering

Might want the algorithm to automatically learn the appropriate number of clusters C

- *May treat C as a hyperparameter and tune it using validation*
- *Agglomerative clustering, Chinese restaurant process automatically do this*

Might not be happy with Euclidean distance as notion of "similarity" – clustering with *Bregman divergences*

Several other problem variants known e.g. k medoids (uses general $d(\mathbf{x}^i, \boldsymbol{\mu}^c)$ instead of $\|\mathbf{x}^i - \boldsymbol{\mu}^c\|_2^2$ 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*

Variations in clustering

Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

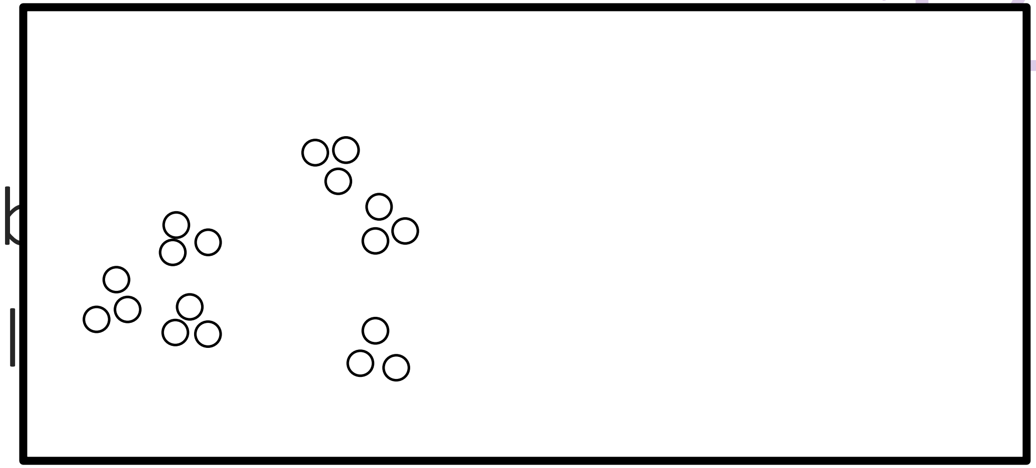
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of "similarity" – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

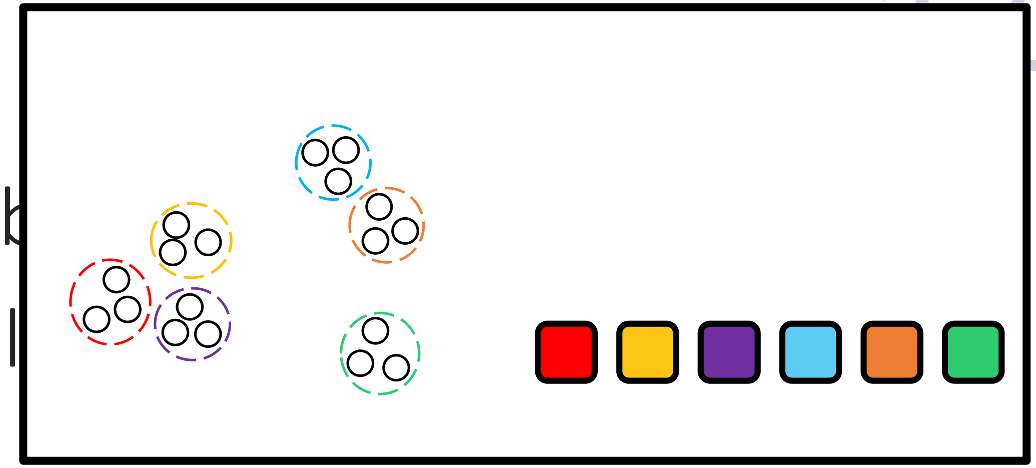
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of “similarity” – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

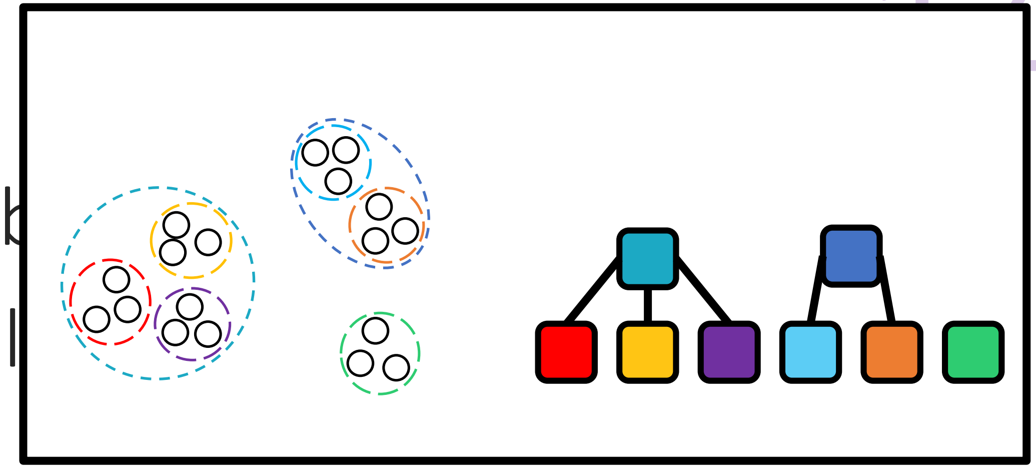
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of “similarity” – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

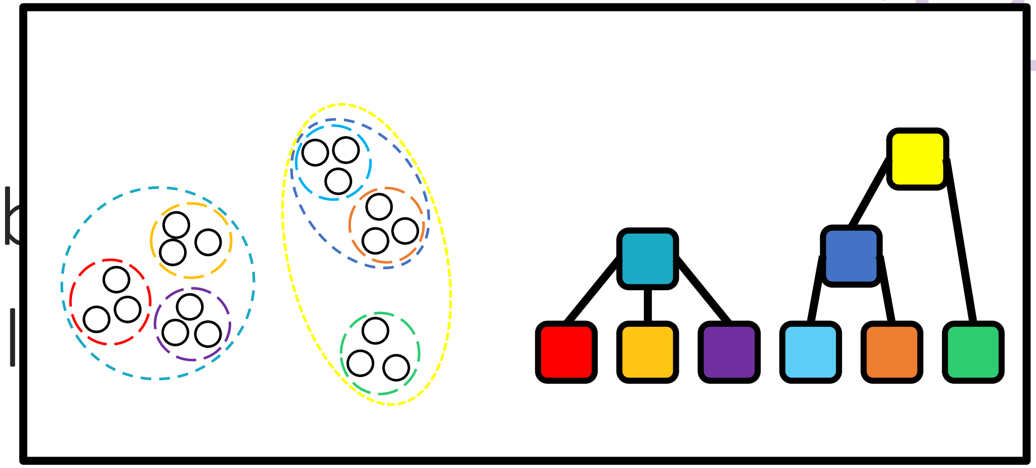
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of "similarity" – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical number of clusters C

May treat C as a hyperparameter and tune it using validation

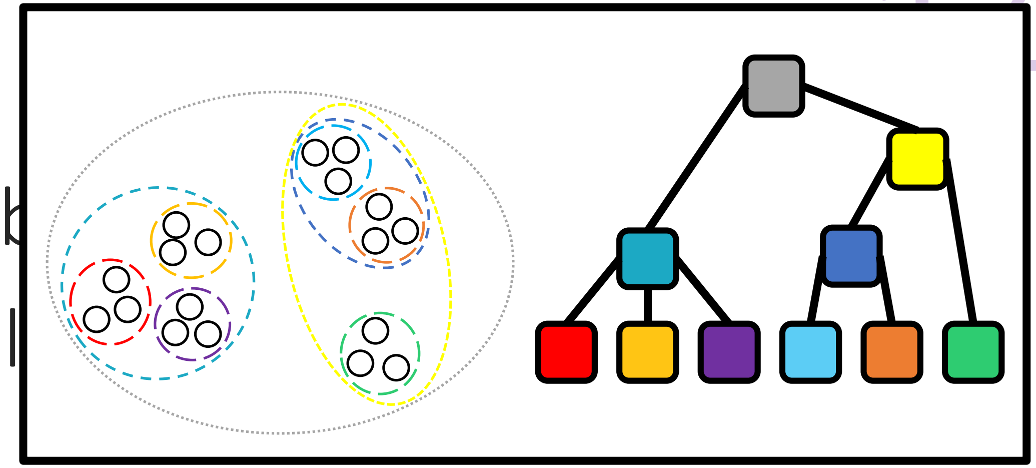
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of “similarity” – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

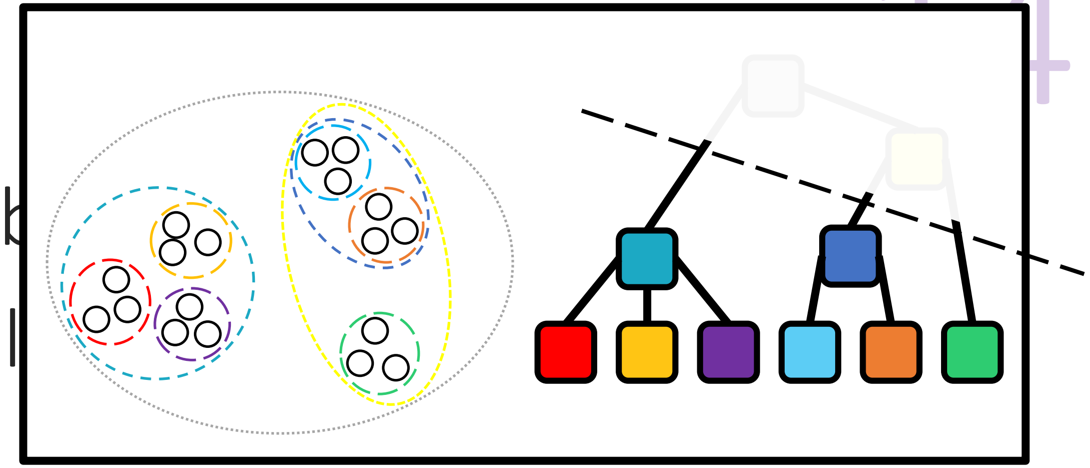
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of "similarity" – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

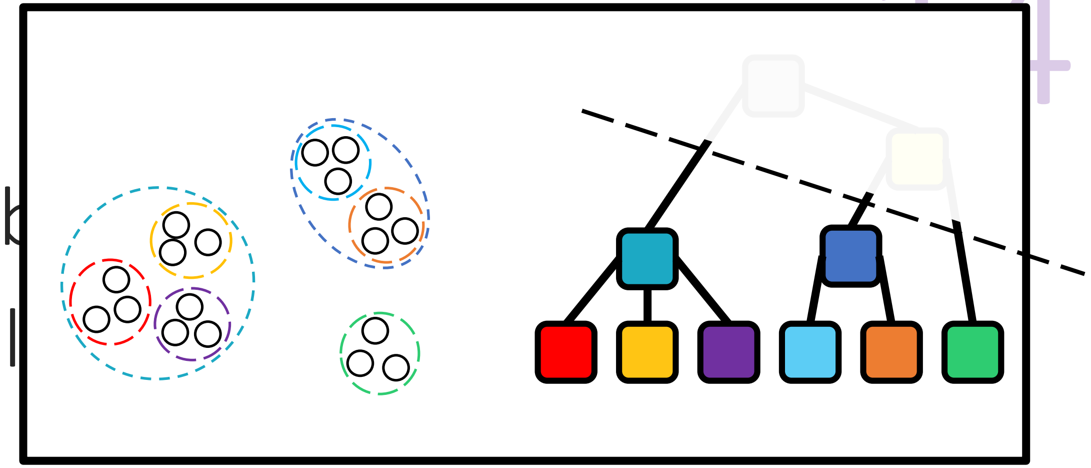
Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of "similarity" – 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

Variations in clustering



Might want to prevent empty clusters – b

Might want the algorithm to automatical
number of clusters C

May treat C as a hyperparameter and tune it using validation

Agglomerative clustering, Chinese restaurant process automatically do this

Might not be happy with Euclidean distance as notion of "similarity" – 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

Probability Theory

What is Probability

Depends on whom we ask this question

- A statistician will claim probability is a way of measuring how frequently does something happen
  - *"If I recommend an iPhone to 1000 female customers aged 25-30 years, roughly 600 of them will make a purchase"*
- A logician will claim that probability is a way of measuring the amount of uncertainty in a certain statement
  - *"If John makes a credit card transaction worth more than ₹10,000, then there is a 70% chance it is fraudulent since he never spends so much"*
- A measure theoretician will claim probability is a way of assigning positive scores in a way so that two scores can be easily compared
  - *"This customer is more likely to be in the 20-30 age group than the 50+ group"*

Machine Learning subscribes to all these views in one way or another

Sample Space

Denotes an exhaustive enumeration of all possible outcomes that either have happened or *could* happen (even if extremely unlikely)

Consider toy setting: we have a website which has 10 products on sale. Customers visit the website, browse and are shown one ad. Depending on their experience, they either purchase one of the 10 products or don't purchase anything. We record gender, age of customer and how many seconds they spend on the website.

Sample Space: $\underbrace{\{\mathrm{M}, \mathrm{F}, \mathrm{T}\}}_{\text{Gender}} \times \underbrace{\mathbb{N}}_{\text{Age}} \times \underbrace{\mathbb{N}}_{\text{Time Spent}} \times \underbrace{\{\mathrm{A0}, \ldots, \mathrm{A9}\}}_{\text{Ad Shown}} \times \underbrace{\{\mathrm{P0}, \ldots, \mathrm{P9}, \emptyset\}}_{\text{Purchase}}$

Sample spaces are usually infinite in size in real settings since they enumerate all possibilities, even very unlikely ones

Events

An *event* is simply a description of useful facts about an outcome

- *A male customer in age group 20-30 years visiting our website is an event*
- *A female customer being shown an ad for a P2 (a laptop) is an event*
- *A customer purchasing something that was shown as an ad is an event*
- *A customer purchasing something that was not shown as an ad is an event*
- *A customer spending more than 20 minutes on the website is an event*

ML can be used to do several useful things

- *Tell us how frequently does an event occur/if one event more likely than other*
  - What fraction of male customers aged 35-40 purchase P6 (a phone) if shown an ad?
  - What fraction of female customers purchase P2 (a laptop) whether ad shown or not?
  - Is it more likely that a purchase will be made if I show a mobile ad or a laptop ad?
  - Is it more likely that a 20-25 year old will purchase if I show a mobile vs laptop ad?
- *Tell us how confident is the ML algorithm while giving the above replies*

Random Variables

Random variables are simply a way to express useful facts about events as numbers so that we can do math with them

Random variables can be categorical or numerical

Categorical: $X = 1$ if female, $X = 2$ if male, $X = 3$ if transgender

Numerical (Discrete): Y = age of person in years

Numerical (Continuous): Z = number of seconds spent on the website

Indicator: $W = 1$ if purchase made on ad shown, $W = 0$ otherwise

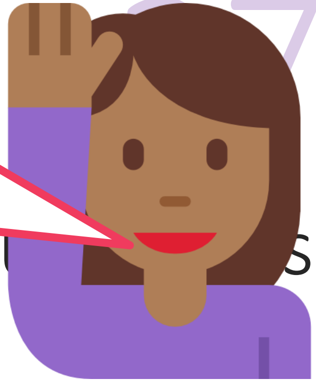
Example Outcome: A male customer aged 25 years spent 18 minutes on our website but did not purchase the product whose ad was shown

$X = 2$, $Y = 25$, $Z = 1080$, $W = 0$

Can arrange several random variables as vectors too – $[2, 25, 1080, 0]$

Ran

I could have also defined a random variable such that S = 1 if purchase made (whether or not on ad shown) and S = 0 otherwise. What I define as a random variable (or even an event) is totally up to my creativity.



Rand s
as numbers so that we can do math with them

Random variables can be categorical or numerical

- *Categorical: X = 1 if female, X = 2 if male, X = 3 if transgender*
- *Numerical (Discrete): Y = age of person in years*
- *Numerical (Continuous): Z = number of seconds spent on the website*
- *Indicator: W = 1 if purchase made on ad shown, W = 0 otherwise*

Example Outcome: A male customer aged 25 years spent 18 minutes on our website but did not purchase the product whose ad was shown

X = 2, Y = 25, Z = 1080, W = 0

Can arrange several random variables as vectors too – [2, 25, 1080, 0]

Probability Distribution

For the purpose of ML, a probability distribution serves two purposes

Given an event it can tell us how likely is that event

This also allows us to ask given two events, which one is more likely

Note that random variables can be used to define events too e.g. $W = 1$ is an event (that a purchase was made on the product whose ad was shown)

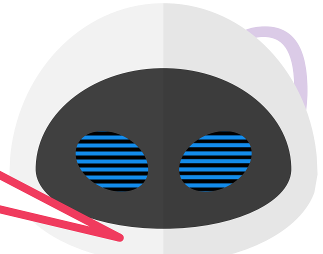
Generate a sample outcome

It is expected that outcomes that are more likely are generated more often than extremely rare outcomes e.g. "a 120 year old man who is shown an ad for P8, spent 1000 seconds but did not purchasing anything" is not very likely

We can also ask for a sample outcome with certain restrictions to be generated e.g. "a female customer who is shown an ad for P6". In this case, we are requesting outcomes that satisfy the above but are more likely, to be generated more often.

Probability Distribu

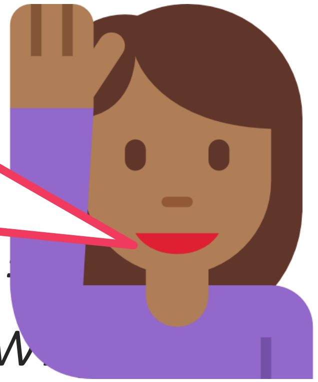
For starters, a 120 year old human being is almost certainly a woman not a man



For the purpose of ML, a probability distribution serves two purposes

Give

In this case, we are interesting in getting samples of female customers who are shown an ad for P6. For example, if such customers are more likely to buy P6 then we would like W = 1 more frequently for these samples too!



Th

No

event (that a purchase was made on the product whose ad was show

Generate a sample outcome

It is expected that outcomes that are more likely are generated more often than extremely rare outcomes e.g. "a 120 year old man who is shown an ad for P8, spent 1000 seconds but did not purchasing anything" is not very likely

We can also ask for a sample outcome with certain restrictions to be generated e.g. "a female customer who is shown an ad for P6". In this case, we are requesting outcomes that satisfy the above but are more likely, to be generated more often.

Getting Started

Sample space: $\{\mathrm{R}, \mathrm{G}, \mathrm{B}\} \times [6]$

$$\mathbb{P}[\mathrm{R}] = \frac{14}{24} = \frac{7}{12}$$

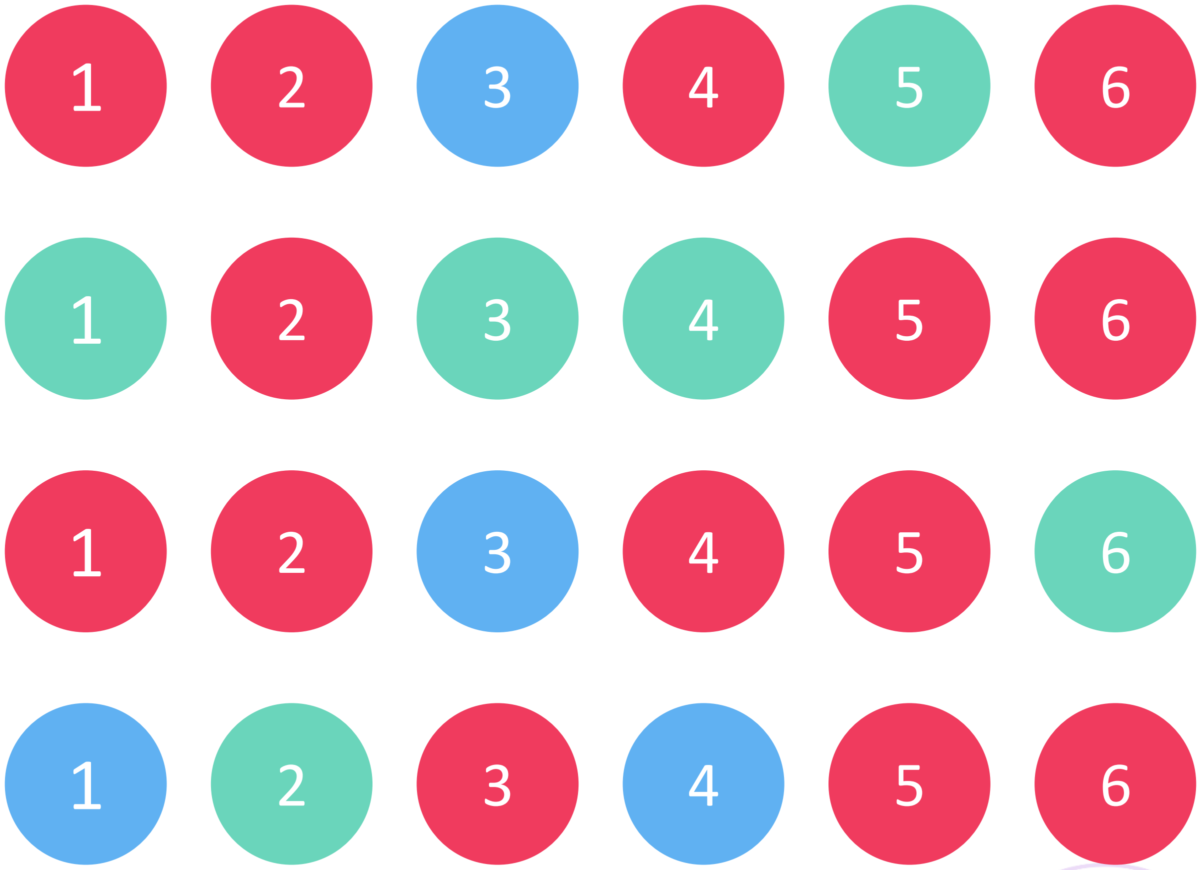
$$\mathbb{P}[\mathrm{B}] = \frac{4}{24} = \frac{1}{6}$$

$$\mathbb{P}[\mathrm{G}] = \frac{6}{24} = \frac{1}{4}$$

Note: $\mathbb{P} \geq 0$ always

$$\mathbb{P}[1] = \frac{1}{6} = \mathbb{P}[2] = \cdots = \mathbb{P}[6]$$

$$\mathbb{P}[\mathrm{R} \wedge 5] = \frac{3}{24} = \frac{1}{8}$$



Initially, to get used to things, it is good to think of probability in terms of *proportions* or *frequency*