

to predict x, need to know

- what factors influence x (and how so (positively/negatively))
- which factors affect more than others

confirmation bias - too small of a sample size to accurately represent the rule (infinitely many patterns to match sample), rules may change over time

positive examples - examples that follow hypothesis (assuming my hypothesis is true, x will follow)

negative examples - examples that do not follow hypothesis (under the hypothesis, x will be false)

= tendency to choose positive examples, need to use negative examples to prove hypothesis = rule

unsupervised learning - finding structure/patterns in data (linear algebraic properties, what does it tell about the data)

supervised learning - making predictions on data

$$X = \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & & \vdots & & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{bmatrix}$$

row = data point
col = attribute/feature

data set with n data points with m features each

feature space = \mathbb{R}^n of X (# of ind cols)

comparing data points


Dissimilarity func - outputs: \uparrow if dissimilar \downarrow if similar

- distance function: all must be true $\begin{cases} d(i,j) = 0 \text{ iff } i=j \\ d(i,j) = d(j,i) \\ d(i,j) \leq d(i,k) + d(k,j) \end{cases}$

- Minkowski Distance: $L_p(x,y) = \left(\sum_{i=1}^d |x_i - y_i|^p \right)^{1/p}$ for d amount of x,y points

$p \geq 1$, but doesn't have to be a whole #

$p=1 \rightarrow$ Manhattan dist 

$p=2 \rightarrow$ Euclidean dist 

aka L_p Norm

vecs
 $\downarrow \downarrow$
- $d(A,B) = \|A-B\|$, $d(0,X) = \|X\|$

Norm = some reference to distance, not all dist funcs create a Norm

Similarity func: \uparrow if similar, \downarrow if dissimilar

- Jaccard Similarity: $JSim(x,y) = \frac{|x \cap y|}{|x \cup y|}$
: $JDist(x,y) = 1 - JSim(x,y)$

* use if data points are mostly similar / small difference is significant

- Cosine Similarity: $s(x,y) = \cos(\theta)$ st θ = angle between x and y

$\cos=1 \rightarrow$ proportional

$\cos=0 \rightarrow$ orthogonal (perpendicular/right angle)

$\cos=-1 \rightarrow$ opposite

* use when direction is more important than magnitude

to get corresponding dissim func:
 $d(x,y) = \frac{1}{s(x,y)}$ or $k - s(x,y)$
for some k

clustering can be ambiguous - no 1 correct answer (but can be wrong ones)

partitional clustering - partition dataset into k partitions, such that variance (cost function) is minimized

cost function - how "good" a clustering is: high = bad (expensive), low = good (cheap)

$$\sum_{i=1}^k \sum_{x \in c_i} d(x, \mu_i)^2$$

$X = \text{dataset} = \{x_1, \dots, x_n\}$
 $k = \text{centers} = \{\mu_1, \dots, \mu_k\}$
 $d = \text{distance}$

k : # clusters, n : # data points

needs to be a balance between cost and # clusters

- can't say lowest cost, bc cost = 0 when $k = n$

- $k = 1$, $k = n$ too easy

- if $X \in \mathbb{R}^n$ st $n > 2$, very difficult (can't visualize)

Lloyd's alg:

1. randomly pick k data points as centers
2. unassign all points
3. assign each point in dataset to closest center
4. compute new centers as means of each cluster
5. repeat 2-4 until done (centers don't change)

Lloyd's always converges, but not always at the optimal solution

- centers might be too close

↳ solve by k-means++

pick random centers, but each point has pr of being selected
proportional to dist from center (outliers more likely)

$$L Pr(x \text{ picked}) = \frac{0(x)^2}{\sum_{x \in X} 0(x)^2} = \text{each point gets } (\text{dist to center})^2 \text{ \# 'entries'}$$

how to choose k:

1. iterate through k to find point of diminishing return

- means gotta do many attempts (not always possible)

* sometimes, lower cost \neq best use

clustering wants:

- similar dps in same
- dissimilar dps in different

k-means only focuses on



= small inner cluster width/distance (a), larger distance between clusters (b)

$$= \underbrace{(b-a) / \max(a,b)}_{\text{want}} : \text{close to 1 when } a \ll \text{ or } b \gg$$

Silhouette Score

for each dp i: a_i = avg dist to neighbor in cluster

b_i = smallest mean dist to every point in another cluster

tags = [string₁, string₂, string₃, ...]

model = SentenceTransformer(model) ← LLM

feature_vectors = model.encode(tags) ← for each tag, converts into array where each index is similarity score to a word in dict

clusters = Kmeans(n_clusters=3, n_init=20).fit(feature_vectors).predict(feature_vectors)

runs Kmeans with 3 centers, 20 times

equal to .fit_predict(feature_vectors)

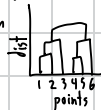
for KMeans: the only 'randomness' comes from initial center selection

↳ same centers = same result

must use mean distance func to correctly cluster

↳ diff func will do something, but not what's intended

Hierarchical Clustering: clustering in tiers/thresholds
 Outputs dendrogram



agglomerative - start w/ each point is own cluster

- compute dist between clusters
- merge 2 closest clusters
- repeat until \forall in the same cluster

divisive - start with \forall in 1 cluster

- at each step, split
- repeat until each point is its own cluster

recursively apply
 kmeans with $k=2$

threshold distance isn't very practical

↳ hard to know what it means

more-so to view data that can't be plotted
 bc dp's in very high dimensions

* $d(x, y)$ - distance between points
 $D(a, b)$ - distance between clusters
 μ_n = mean / avg dp pos in C_n

how to:

1. make matrix of all points vs themselves
2. compute distance between all points
3. group together 2 closest points
4. redefine matrix, include combined points as clusters
5. redefine dist to \forall other points
6. add closest dp to cluster
7. repeat 4-7 until \forall in 1 cluster

ways to calc $D(C_1, C_2)$

1. single-length dist: min dist between any points in C_1, C_2
 - ↳ sensitive to noise in between dps
 - ↳ creates elongated clustering

2. complete length dist: max dist between any points in C_1, C_2
 - ↳ opposite to SL: handles noise well
 - ↳ sensitive to outliers

3. average-link dist: $\frac{1}{|C_1| \cdot |C_2|} \sum d(p_1, p_2)$

4. centroid dist: similar to kmeans
 - ↳ pos of C is average position of $\forall p \in C$

5. Ward's Dist: $\sum_{p \in C_{12}} d(p, \mu_{12}) - \sum_{p \in C_1} d(p, \mu_1) - \sum_{p \in C_2} d(p, \mu_2)$
 - ↳ merging C_1 and C_2

(spread around mean in merged) - (spread around mean in C_1) - (spread around mean in C_2)

Density Based (DB Scan)

density: take radius of length ϵ around point

↳ if # points within region $>$ threshold: dense ✓

core point: ϵ -neighborhood contains $\geq \langle \text{min_points} \rangle$

border point: in ϵ -neighborhood of core point

noise point: neither core or border

since *density definition* is defined, won't pick up on differently dense areas = mark as outliers
 ↳ $\epsilon, \langle \text{min_points} \rangle$ ↗

↳ thus makes clusters of the same density

DFS method: 1. iterate through dataset

2. if dp is core (has $\langle \text{min_points} \rangle$ of neighbors within ϵ)

- iterate through neighborhood

- if neighbor is core: repeat ↗

↳ else: neighbor is border, add to cluster

- continue iterating through neighborhood

- once no more undiscovered nodes in neighborhood,

repeat step 1 with next undiscovered dp in dataset

Color reduction - reducing image to x colors, st its still recognizable

- use RGB values of each pixel as $\underbrace{\text{coordinate points}}_{\substack{\text{normalize vals} \\ \text{prob}}} = \text{color space}$

HSV

LAB

L - lightness

A - green-red

B - blue-yellow

- run kmeans on color space \rightarrow each centroid is color to
set all closest points to

Soft Clustering

- each subset in a data group has different distributions
- say that labels for each dist is lost: how to determine which belongs to which
 - need: distributions of each subset, how many points for each dist

$$\text{total data set proportion: } P(X) = \underbrace{\sum_j P(S_j)}_{\text{proportion of } S} \cdot \underbrace{P(X|S_j)}_{\text{prob of } X \text{ within } S_j}$$

Mixture Distribution:

- going to assume each dist follows Normal Distribution
- but how to determine mean and std deviation for each?

Comparing Clusterings: Clustering Aggregation - are points x and y in the same clusters in their respective clusterings
 ↳ output of a clustering alg

$$\text{Disagreement Dist: } \sum_{x,y} \mathbb{I}_{P,C}(x,y)$$

- want to minimize

$$\begin{cases} 1 & \text{if } P \text{ and } C \text{ disagree on which clusters } x \text{ and } y \text{ belong to} \\ 0 & \end{cases}$$

SVD - single value decomposition

- want to look for factor, such that changing it only changes the outcome and not another factor
 - ↳ likely not possible

- want to eliminate linear relationship between variables
 - ↳ so changing one var usually doesn't change another

try to narrow down data set (via eliminating vars) so that the remaining data follows the same general pattern/variance

↳ reduce Rank = faster computations

- reduce matrix such that it contains similar data, but all cols linearly independent
- same amount of data points, less variables
 - ↳ remove redundant info
- once cols linearly ind., vary intensity of each point to distinguish them

3/19 - Classification

↳ understand relationship between factors that result in output
predictors/features → class

depending on data: multiple/no answers

↳ could be lack of data, incorrect features, or simply impossible

no correct answers = ok, might still be useful

good predictors:

- distinctive distribution between predictors

$$\text{- correlation: } r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{(\sqrt{\sum (x_i - \bar{x})^2})(\sqrt{\sum (y_i - \bar{y})^2})}$$

$r = 0 \neq$ no relationship

\bar{x} = mean of feature

range: $-1 \leq r \leq 1$ } -1 = neg. corr.
 0 = no corr.
 1 = pos. corr.

if features continuous but class: distinct

$y = \{\text{yes, no}\} = \text{nominal} \rightarrow$ look at means of x + how they differ among each val of y
 $y = \{\text{ok, good, bad}\} = \text{ordinal} \rightarrow$ compare to position of y rather than give numerical vals

map to $0 = \text{no}$
 $1 = \text{yes} = \text{want to predict}$

good set of predictors:

- choose features that are related to class, not related to each other

good classification (test w/o cheating, learning \neq memorizing)

- split ds into training/testing

↳ shipping model trained on whole ds = risk overfitting to that subset of data
↳ too specific to ds

underfitting = model too simple

correlation = increase x , expect y to increase

↳ could be due to hidden factor, or counterexample

causation = requires specific testing with control group

Classifiers

- instance based = training set = model itself

↳ search training for testing entry, or get similar results and aggregate class to get output

distance threshold is arbitrary, will get varying levels of neighbors

- K Nearest Neighbors

↳ get k # of similar/closest training data, aggregate classes = prediction

↳ closeness scales based on attributes
(if 1 factor has wider scale/range = dominant, want to scale so all attributes similar)

ex: weigh based on closeness

↳ use libraries k-NN

↳ high dimensions might cause issues - Curse of Dimensionality

- Decision Trees

↳ take subsets of data + classes = Hunt's Alg

↳ subsets might not be perfect, ok to generalize

↳ can split binary or multi-split

↳ how to evaluate "goodness" of split?

- GINI(t) = $1 - \sum p(t|j)^2$ (low # = better, between 0 and 0.5)

- GINI of split = $\sum_{j=1}^n \frac{n_j}{n} \text{GINI}(t_j)$

can result in overfitting
- terminate at x layers
- build full tree, then trim
↳ use validation set to not overfit on testing

Naive Bayes

- $P(\text{class result} | \text{attribute(s)})$

predict class that maximizes $P(\text{class} | \{n \text{ attributes}\})$

Union of n attributes

- assume attributes are independent

- cont attributes: binning / multi-way split

- pdf estimation (use N with sample mean, sample variance)

Model Eval

build cost matrix: associate cost with correct/incorrect prediction

		Prediction	
		Yes	No
Actual	Yes	1	0
	No	0	1

↳ false negative = big penalty
 ↳ false positive = small penalty

$$\text{accuracy} = \frac{(\text{yes, yes}) + (\text{no, no})}{\sum_{i \in \text{yes, no}} (\sum_{j \in \text{yes, no}} (i, j))}$$

how to get accurate estimate model on unseen data

- split into testing data

- cross validation: continuously train on entire ds - 1 output

- use validation

Ensemble methods

- combine the results/outputs of a bunch of classifiers

- bagging: divide testing into a bunch of smaller testing data

- boosting: each iteration, emphasize d's that prev classifier struggled with
↳ each classifier has weight

Support Vector Machine -

- find v_1 and v_2 such that $\overbrace{w_1 x_1 + w_2 x_2 + b}^{w^T x + b} = 0$ divides + and - points
- for any $[u_i]$, plug into eq: if result is < 0 = left, > 0 = right
- want street to be as far from d's as possible
 - ↳ gutters = line eq = -1 and line eq = 1; multiplying by const widens gutters
 - ↳ want max size of st

but ds aren't perfect: option 1 - allow for some misclassification
2 - warp space to allow correctness

only certain points contribute towards street length = support vectors
want: $\tilde{w} \cdot \tilde{x}_+ + b \geq 1$ and $\tilde{w} \cdot \tilde{x}_- + b \leq -1 \Rightarrow y_i (\tilde{w}_i \cdot \tilde{x}_i + b) - 1 = 0$

width of street = distance between support vectors

1. info about users + movies

↳ recommend similar movies to those the user has watched or a similar user has watched

$$\text{Primal form: } \min_{w, b} \frac{1}{2} \|w^2\| \quad \text{s.t. } (w^T x_i + b) y_i \geq 1$$

$$= \min \frac{1}{2} \|w^2\| + \text{penalty}$$

$$\begin{cases} 0 & \text{if } (w^T x_i + b) y_i \geq 1 \\ \infty & \text{otherwise} \end{cases}$$

$$\max_{\alpha_i \geq 0} \{ \alpha_i [1 - (w^T x_i + b) y_i] \}$$

$$= \max_{\alpha_i \geq 0} \min_{w, b} \left\{ \frac{1}{2} \|w^2\| + \sum_{i=1} \alpha_i [1 - (w^T x_i + b) y_i] \right\}$$

$$f(w, b, \alpha)$$

$$\frac{\partial f}{\partial w} = w - \sum_i \alpha_i x_i y_i = 0 \Rightarrow w = \sum_i \alpha_i x_i y_i$$

$$\frac{\partial f}{\partial b} = -\sum_i \alpha_i y_i = 0$$

$$\max_{\alpha_i \geq 0} f(w, b, \alpha_i) = \max_{\alpha_i \geq 0} \left\{ \frac{\|w^2\|}{2} + \sum_i \alpha_i [1 - (w^T x_i + b) y_i] \right\}$$

$$= \frac{\|\sum_i \alpha_i x_i y_i\|^2}{2} + \sum_i \alpha_i - \alpha_i w^T x_i y_i + \alpha_i b y_i$$

$$\begin{aligned} k(a, b) &= (a, b)^3 = ((a_1, a_2) \cdot (b_1, b_2))^3 \\ &= (a_1 b_1 + a_2 b_2)^3 \\ &= a_1^3 b_1^3 + 3 a_1^2 b_1^2 a_2 b_2 + 3 a_1 b_1 a_2^2 b_2 + a_2^3 b_2^3 \end{aligned}$$