Clustering & Retrieval:

A machine learning perspective

Nearest Neighbor Search:

Retrieving Documents

Retrieving documents of interest

Document retrieval

- Currently reading article you like

Document retrieval

- Currently reading article you like
- **Goal**: Want to find similar article

Document retrieval

Challenges

- How do we measure similarity?
- How do we search over articles?

Retrieval as k-nearest neighbor search

What is retrieval?

Search for related items

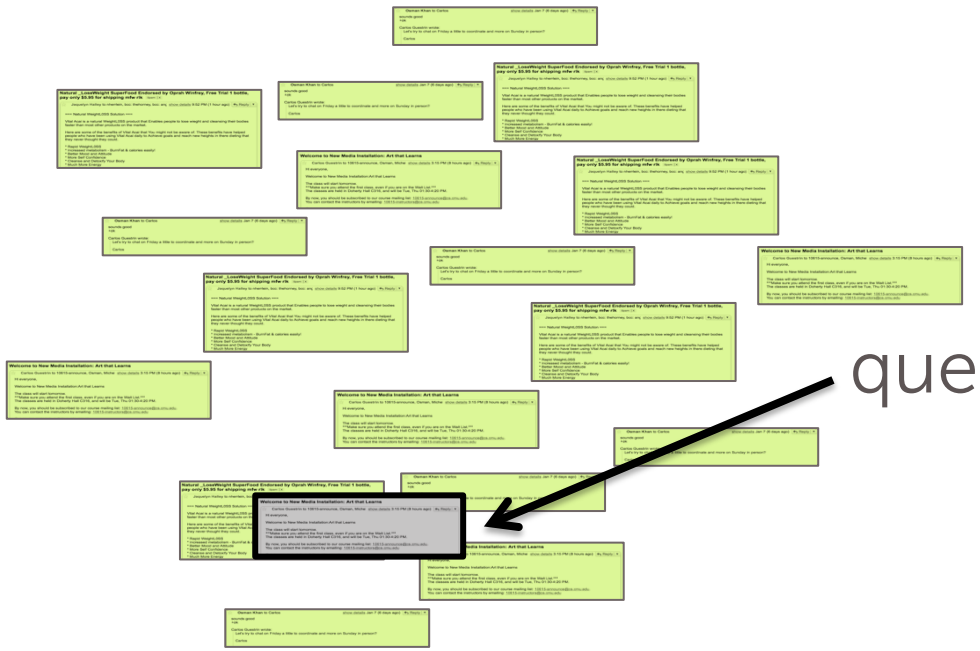
Input **x**,{**x'**}:
features for query point
+
features of all other datapoints

Compute distances to other **x'**

Output $\mathbf{x}^{NN}$:
"nearest" point or set of points to query

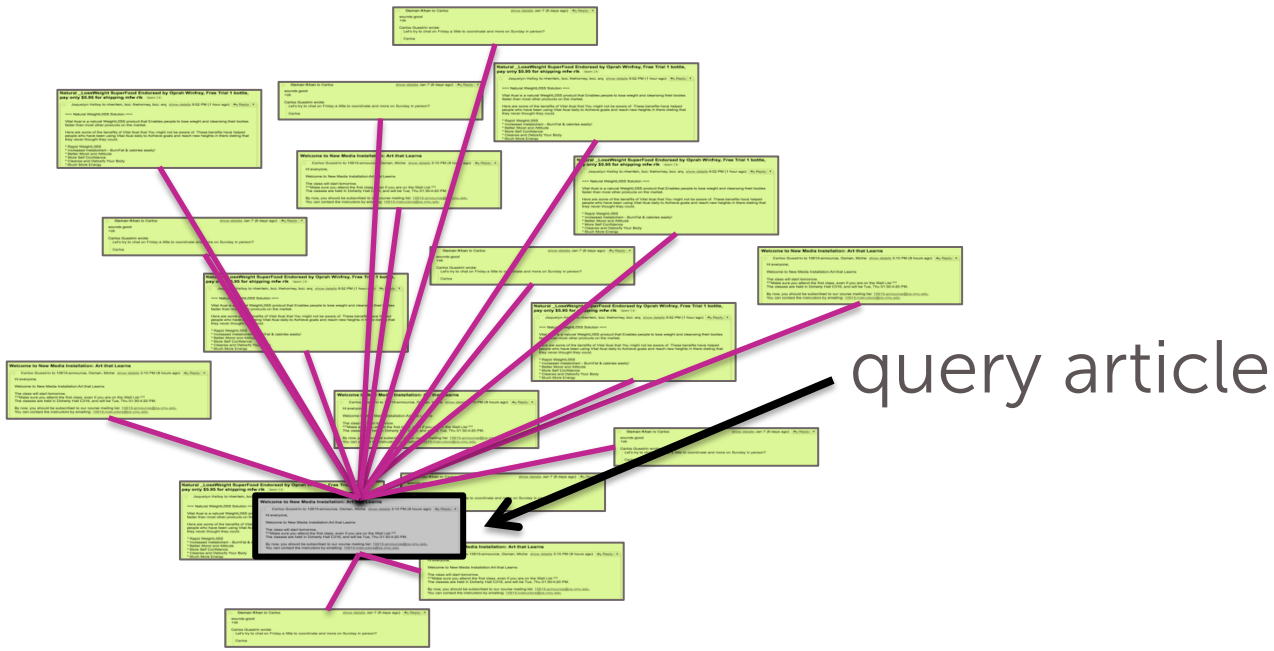
1-NN search for retrieval

Space of all articles,
organized by similarity of text



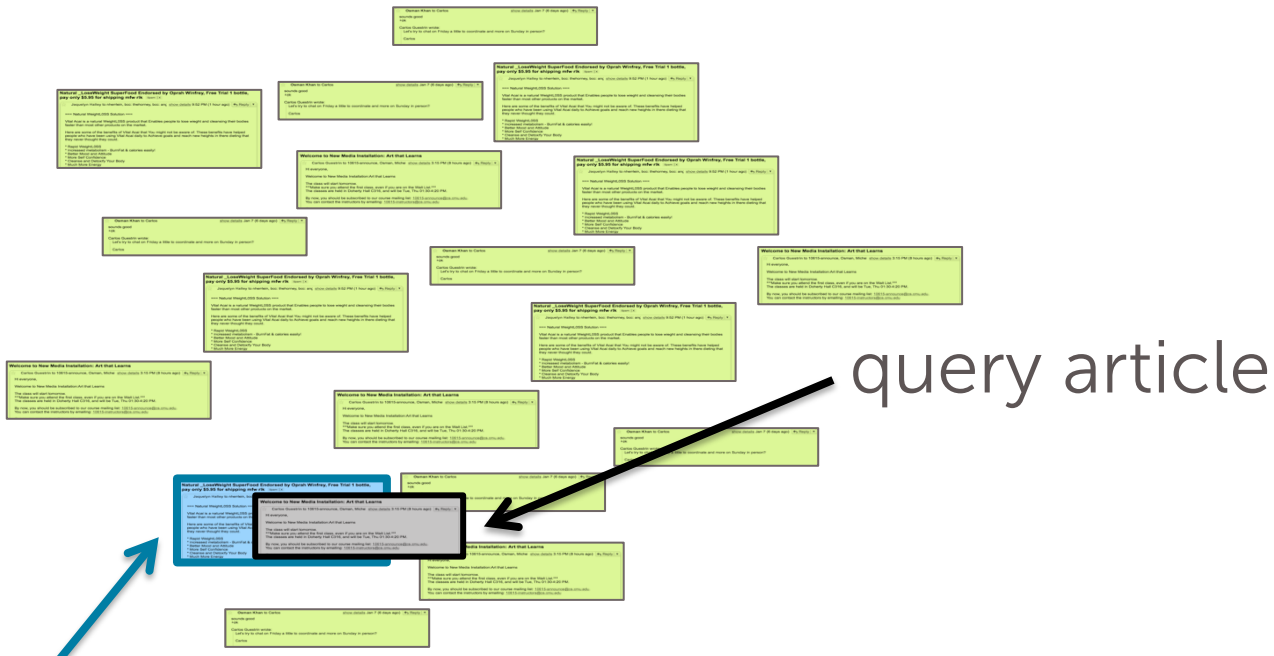
Compute distances to all docs

Space of all articles,
organized by similarity of text



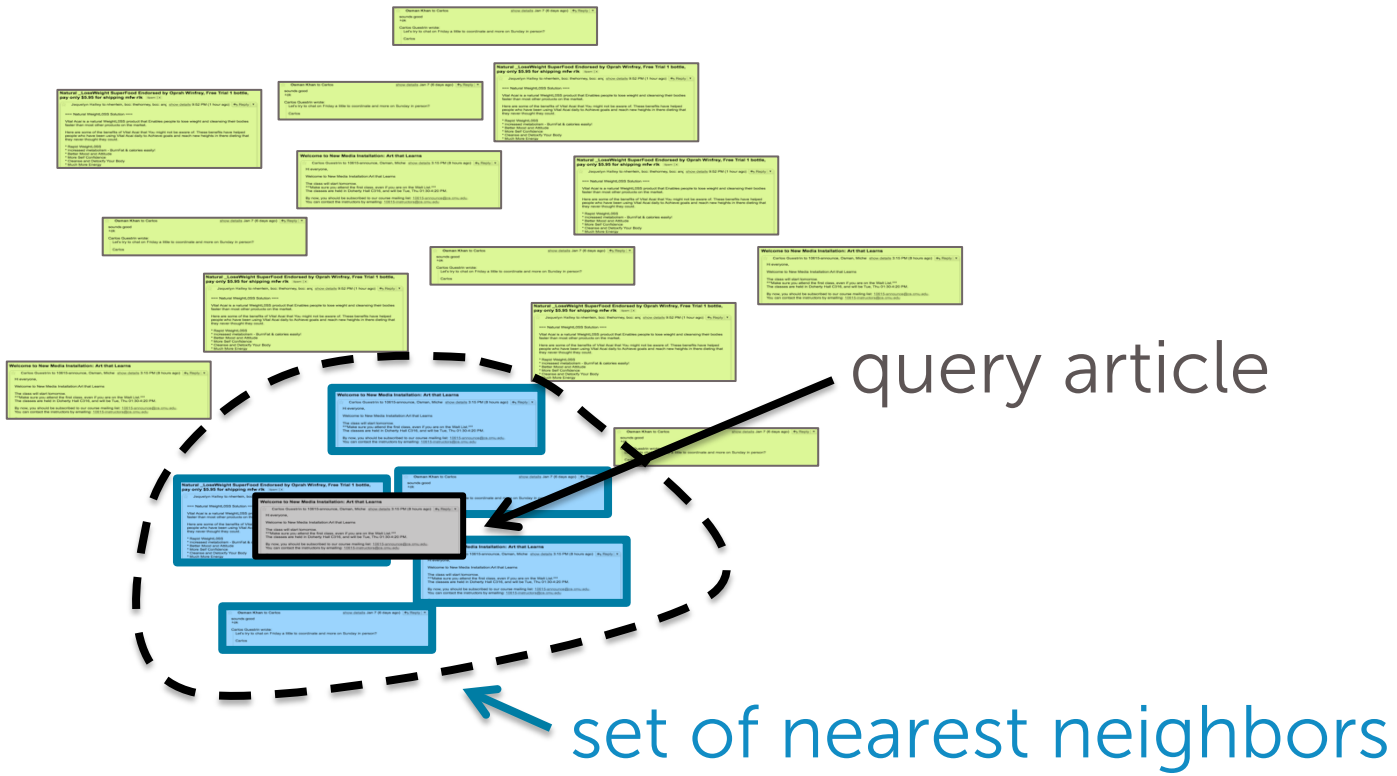
Retrieve “nearest neighbor”

Space of all articles,
organized by similarity of text



Or set of nearest neighbors

Space of all articles,
organized by similarity of text



1-NN algorithm

1 – Nearest neighbor

- **Input:** Query article  : $\mathbf{x}_q$

  Corpus of documents (N docs)

   : $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

- **Output:** *Most* similar article  $\leftarrow x^{NN}$

Formally: $x^{NN} = \min_{x_i} \text{distance}(x_q, x_i)$

1-NN algorithm

closest document

Initialize **Dist2NN** = ∞,  = ∅

For i=1,2,…,N

query document

x_q

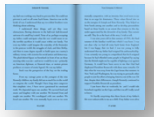
x_i

Compute: δ = **distance(**  ,  **)**

document i
from corpus

If δ < **Dist2NN**

x_i

set  = 

set **Dist2NN** = δ

Return most similar document 

x^{NN}

closest document in
corpus to query article

k-NN algorithm

k – Nearest neighbor

- **Input:** Query article  : $\mathbf{x}_q$

  Corpus of documents

   : $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

- **Output:** *List of k* similar articles

Formally:

$X^{NN} = \{X^{NN_1}, \ldots, X^{NN_k}\}$

for all x_i not in X^{NN}, $\text{distance}(x_i, x_q) \geq \max_{X^{NN_j}, j=1..k} \text{distance}(X^{NN_j}, x_q)$

k-NN algorithm

sort first **k documents** by distance to query doc

Initialize **Dist2kNN** = sort($\delta_1,\ldots,\delta_k$) ← list of sorted distances

= sort() ← list of sorted docs

For i=k+1,…,N

query doc

Compute: δ = **distance(**  ,  **)**

If δ < **Dist2kNN**[k] ← distance to k^{th} NN (furthest NN in set)

find j such that δ > **Dist2kNN**[j-1] but δ < **Dist2kNN**[j]

remove furthest house and shift queue:

 [1:k] =  [1:k-1]

inserting new article

Dist2kNN[j+1:k] = **Dist2kNN**[j:k-1]

set **Dist2kNN**[j] = δ and  [j] = 

closest k docs to query doc

Return k most similar articles 

Critical elements of NN search

Item (e.g., doc) representation

$\mathbf{x}_q \leftarrow$ 

Measure of distance between items:

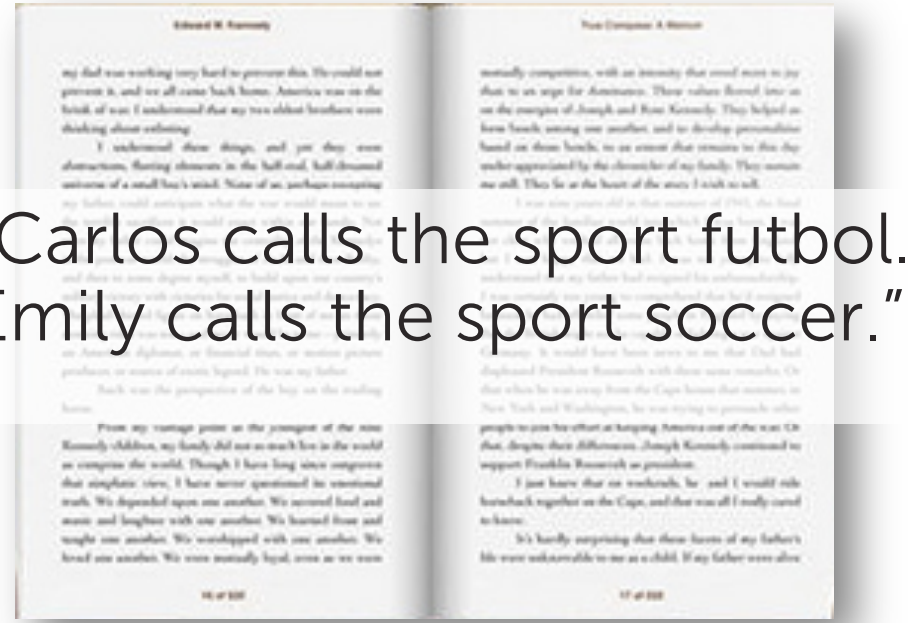
δ = distance($\mathbf{x}_i$, $\mathbf{x}_q$)

Document representation

Word count document representation

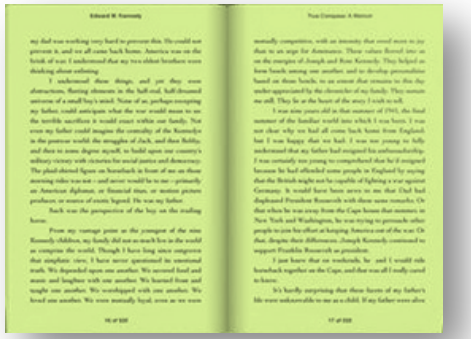
Bag of words model

- Ignore order of words
- Count # of instances of each word in vocabulary



"Carlos calls the sport futbol. Emily calls the sport soccer."

Issues with word counts – Rare words



Common words in doc: “the”, “player”, “field”, “goal”

Dominate rare words like: “futbol”, “Messi”

TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)

- Appears rarely in corpus (rare globally)

TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)

  Term frequency = word counts

- Appears rarely in corpus (rare globally)

TF-IDF document representation

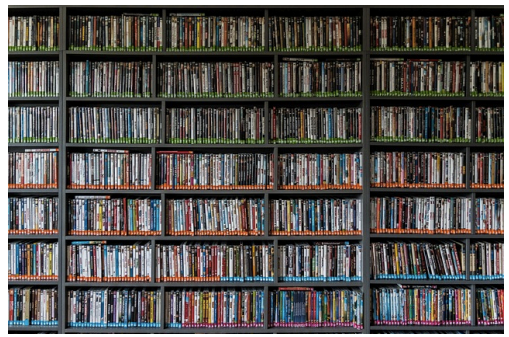
Emphasizes important words

- Appears frequently in document (common locally)

  Term frequency = word counts

- Appears rarely in corpus (rare globally)

  Inverse doc freq. = $\log \frac{\#\ \text{docs}}{1 + \#\ \text{docs using word}}$



TF-IDF document representation

Emphasizes important words

- Appears frequently in document (common locally)

Term frequency = word counts

- Appears rarely in corpus (rare globally)

Inverse doc freq. = $\log \frac{\#\text{ docs}}{1 + \#\text{ docs using word}}$

tf * idf

Trade off: local frequency vs. global rarity

Distance metrics

Distance metrics: Defining notion of "closest"

In 1D, just Euclidean distance:

$$\text{distance}(x_i, x_q) = |x_i - x_q|$$

In multiple dimensions:

- can define many interesting distance functions
- most straightforwardly, might want to weight different dimensions differently

Weighting different features

Reasons:

– Some features are more relevant than others

 # bedrooms
 # bathrooms
sq.ft. living
sq.ft. lot
floors
year built
year renovated
waterfront

Weighting different features

Reasons:

– Some features are more relevant than others

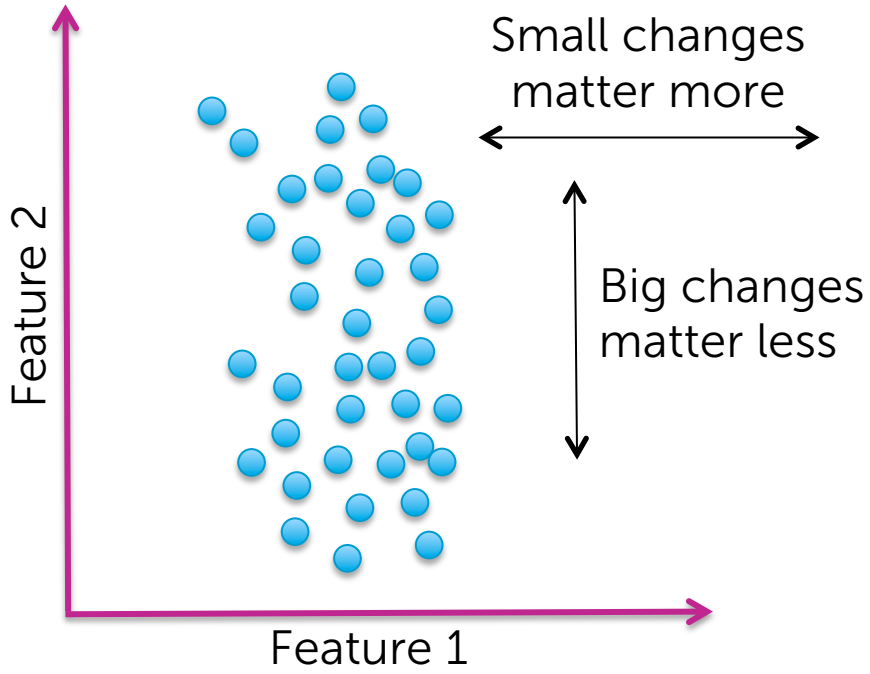
title
abstract
main body
conclusion

Weighting different features

Reasons:

– Some features are more relevant than others

– Some features vary more than others



Specify weights as a function of feature spread

For feature j:

$$\frac{1}{\max_i(\mathbf{x}_i[j]) - \min_i(\mathbf{x}_i[j])}$$

Scaled Euclidean distance

Formally, this is achieved via

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$

weight on each feature
(defining relative importance)

Effect of binary weights

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$

Setting weights as 0 or 1 is equivalent to **feature selection**

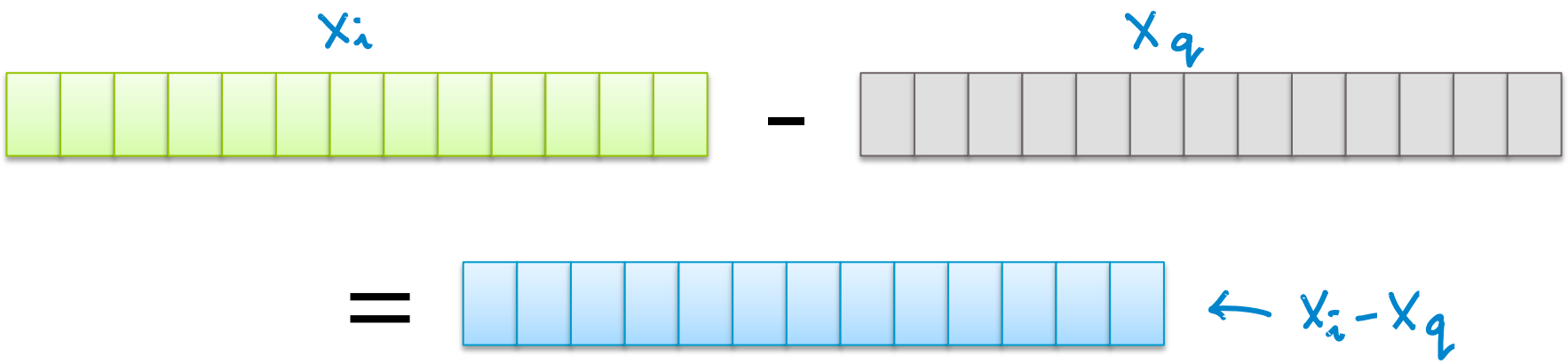
Feature engineering/ selection is **important**, but **hard**

(non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T(\mathbf{x}_i - \mathbf{x}_q)}$$

$$= \sqrt{(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \ldots + (\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$



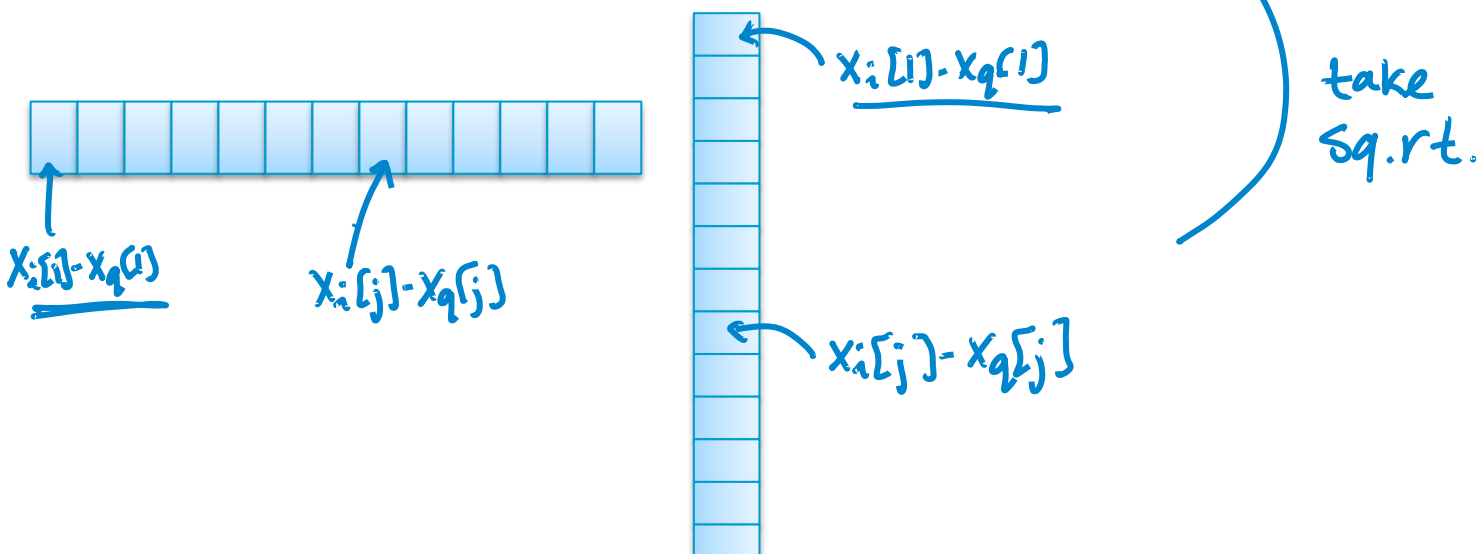
(non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T(\mathbf{x}_i - \mathbf{x}_q)}$$

$$\sqrt{(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \ldots + (\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$

distance^2 =

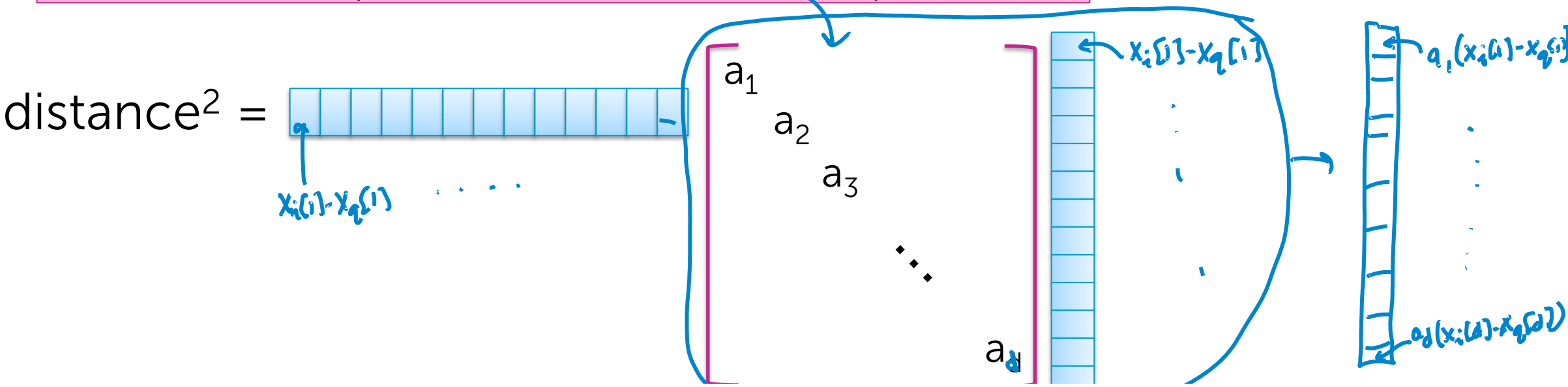


Scaled Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_q)}$$

$$= \sqrt{a_1(\mathbf{x}_i[1] - \mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_i[d] - \mathbf{x}_q[d])^2}$$



Another natural inner product measure

x_q

1	0	0	0	5	3	0	0	1	0	0	0	0

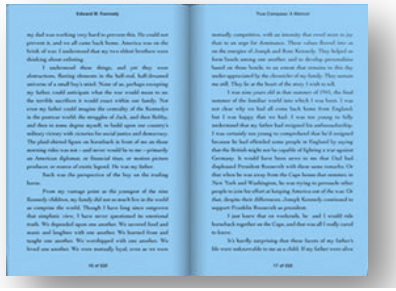
Similarity

$$= \mathbf{x}_i^T \mathbf{x}_q$$

$$= \sum_{j=1}^{d} \mathbf{x}_i[j]\, \mathbf{x}_q[j]$$

3	0	0	0	2	0	0	1	0	1	0	0	0

x_i



= 13

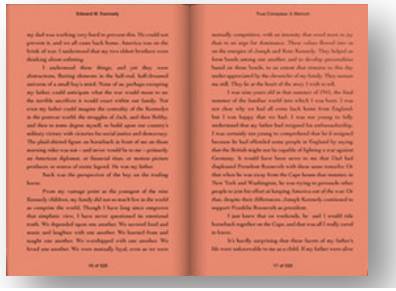
Another natural inner product measure

1	0	0	0	5	3	0	0	1	0	0	0	0

Similarity

= 0

0	0	1	0	0	0	9	0	0	6	0	4	0



Cosine similarity – normalize

Similarity = $$\frac{\sum_{j=1}^{d} \mathbf{x}_i[j]\,\mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^{d} (\mathbf{x}_i[j])^2}\sqrt{\sum_{j=1}^{d} (\mathbf{x}_q[j])^2}}$$

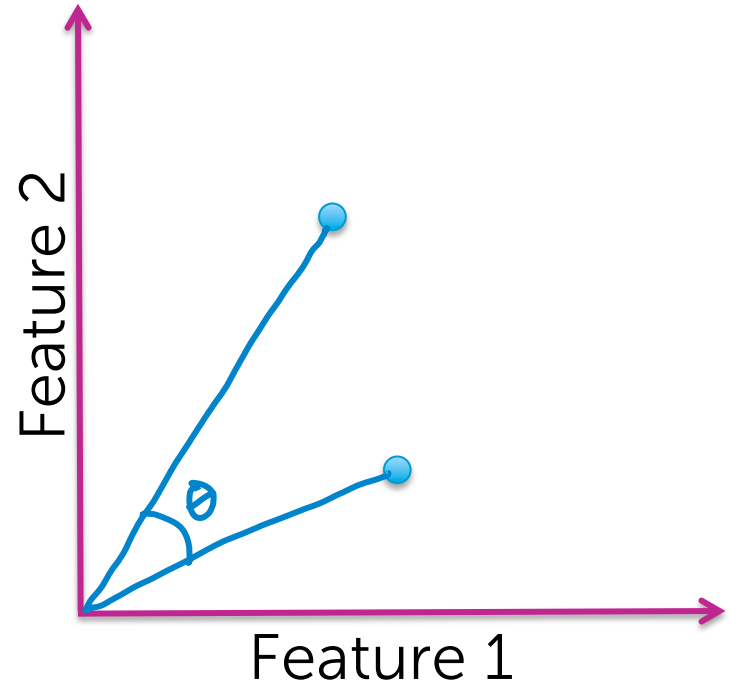
$$\mathbf{x}_i^T \mathbf{x}_q = \cos(\theta)$$

$$\frac{\mathbf{x}_i^T \mathbf{x}_q}{||\mathbf{x}_i||\,||\mathbf{x}_q||}$$

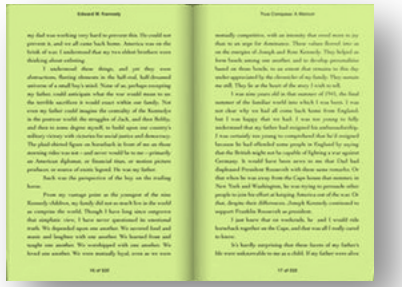
$$= \left(\frac{x_i}{\|x_i\|}\right)^T \left(\frac{x_q}{\|x_q\|}\right)$$ first normalize

$a^T b = \|a\|\,\|b\| \cos(\theta)$

- Not a proper distance metric
- Efficient to compute for sparse vecs



Normalize



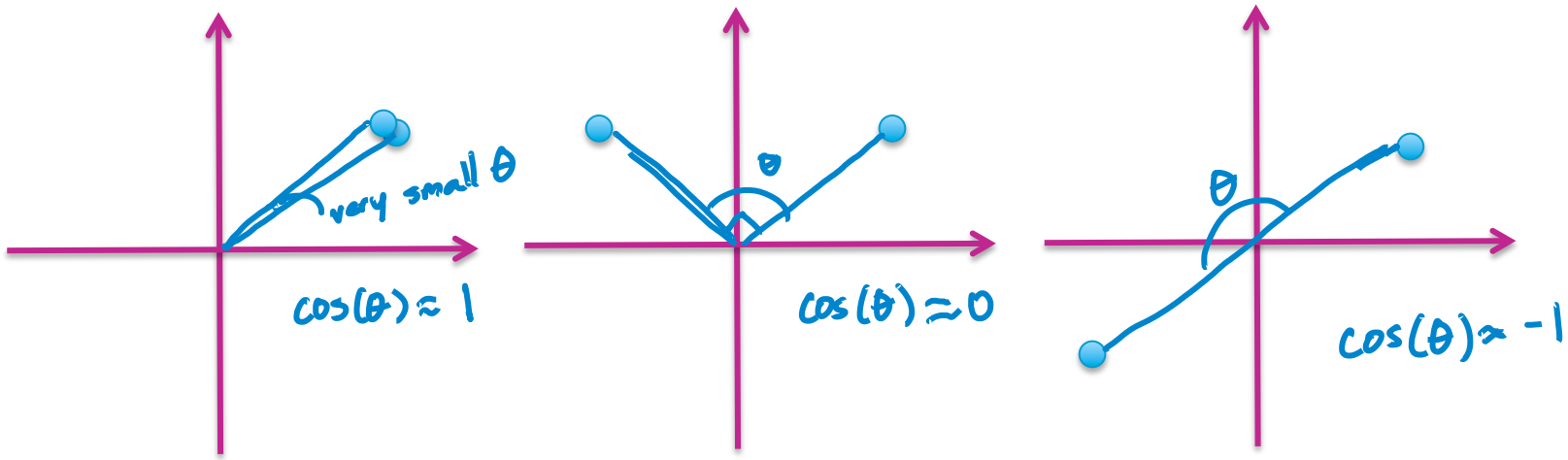
1	0	0	0	5	3	0	0	1	0	0	0	0

$\leftarrow x_i$

$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)} \leftarrow \|x_i\| = \sum_{j=1}^{d} x_i[j]^2$$

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0

Cosine similarity



In general, $-1 <$ similarity < 1

For positive features (like tf-idf)
$0 <$ similarity < 1 } our focus

Define **distance = 1-similarity**

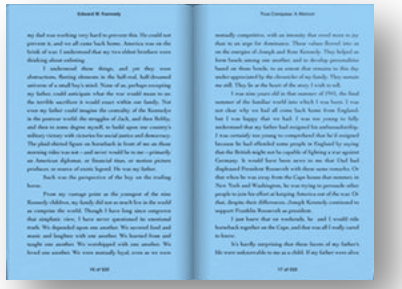
To normalize or not?

1	0	0	0	5	3	0	0	1	0	0	0	0

2	0	0	0	10	6	0	0	2	0	0	0	0

3	1	0	0	2	0	0	1	0	1	0	0	0

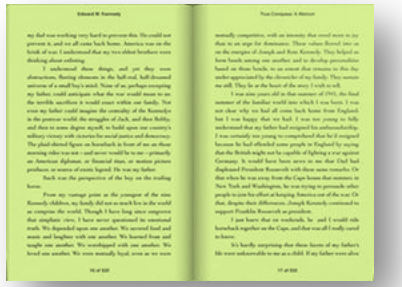
Similarity = 13



6	2	0	0	4	0	0	2	0	2	0	0	0

Similarity = 52

Normalize



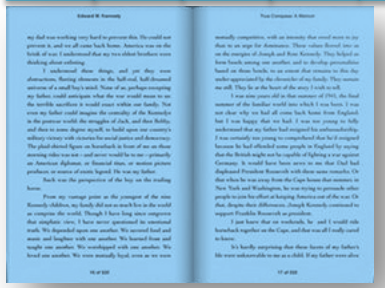
1	0	0	0	5	3	0	0	1	0	0	0	0

$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)}$$

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0

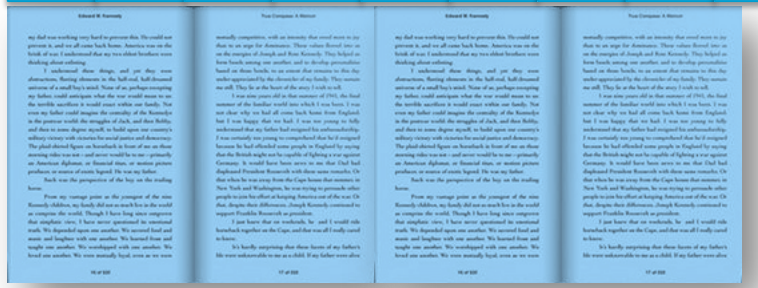
In the normalized case

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0
3/4	1/4	0	0	1/2	0	0	1/4	0	1/4	0	0	0



Similarity = 13/24

1/6	0	0	0	5/6	3/6	0	0	1/6	0	0	0	0
3/4	1/4	0	0	1/2	0	0	1/4	0	1/4	0	0	0



Similarity = 13/24

Other distance metrics

- Mahalanobis
- rank-based
- correlation-based
- Manhattan
- Jaccard
- Hamming
- …

Combining distance metrics

Example of document features:

1. Text of document
   - Distance metric: Cosine similarity
2. # of reads of doc
   - Distance metric: Euclidean distance

Add together with user-specified weights

Clustering:
Grouping Related Docs



Motivating clustering approaches

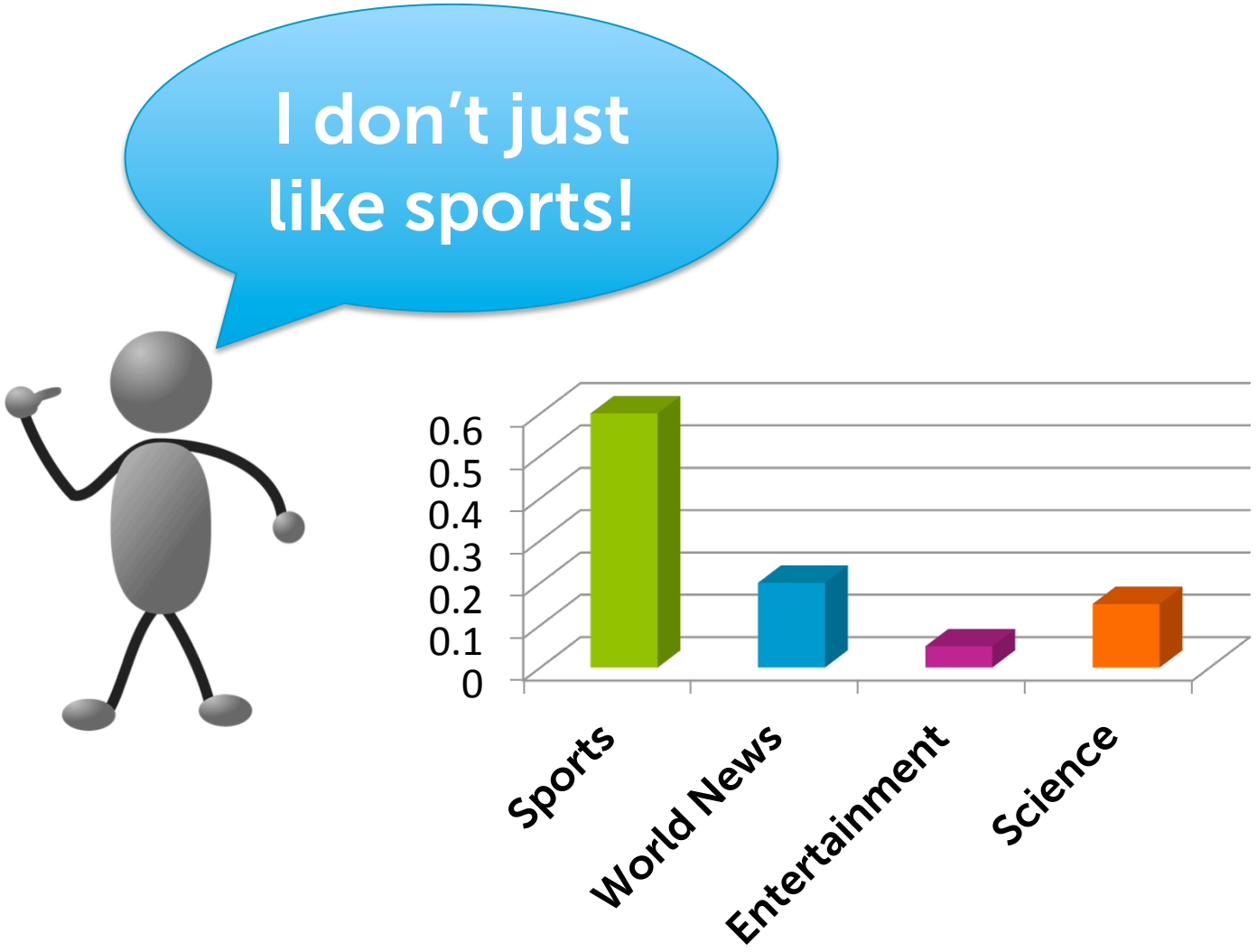
Goal: Structure documents by topic

Discover groups (*clusters*) of related articles

SPORTS

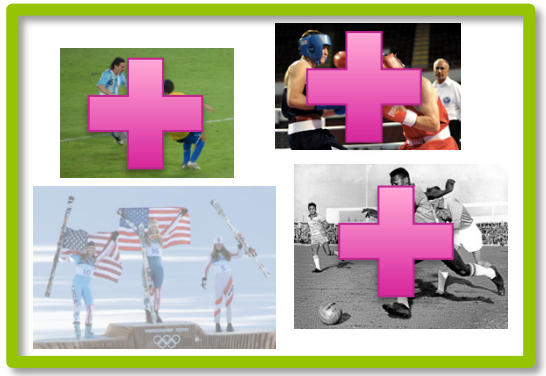
WORLD NEWS

Why might clustering be useful?



Learn user preferences

Set of clustered documents read by user

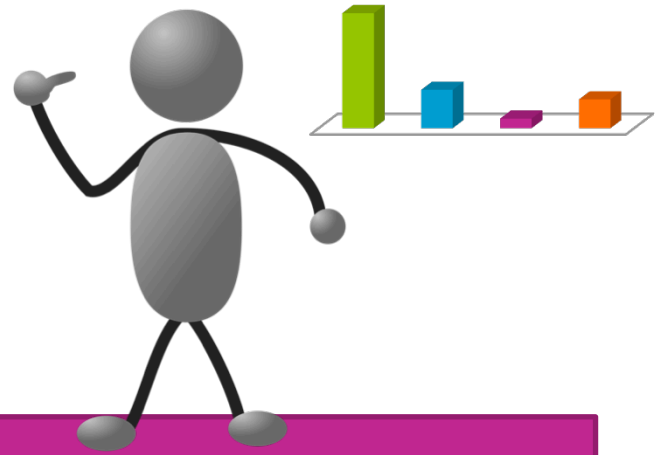


Cluster 1

Cluster 2

Cluster 3

Cluster 4



Use feedback to learn user preferences over topics

Clustering: An unsupervised learning task

What is retrieval?

Search for related items

Input **x**,{**x'**}:
features for query point
+
features of all other datapoints

Compute distances to other **x'**

Output $\mathbf{x}^{NN}$:
"nearest" point or set of points to query

What is clustering?

Discover groups of similar inputs

Input {**x**}:
features for points in dataset

Separate points into disjoint sets

Output {z}:
cluster labels per datapoint

What if some of the labels are known?

Training set of labeled docs

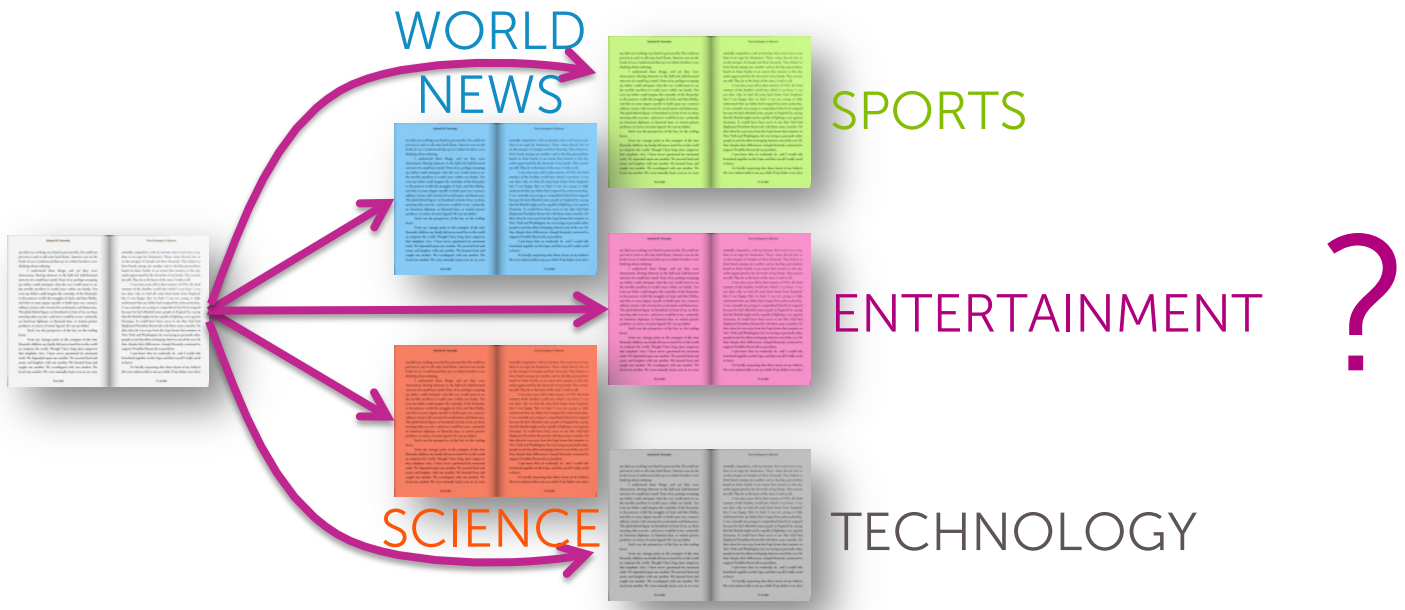
SPORTS

WORLD NEWS

ENTERTAINMENT

SCIENCE

Multiclass classification problem



Example of
supervised learning

Clustering

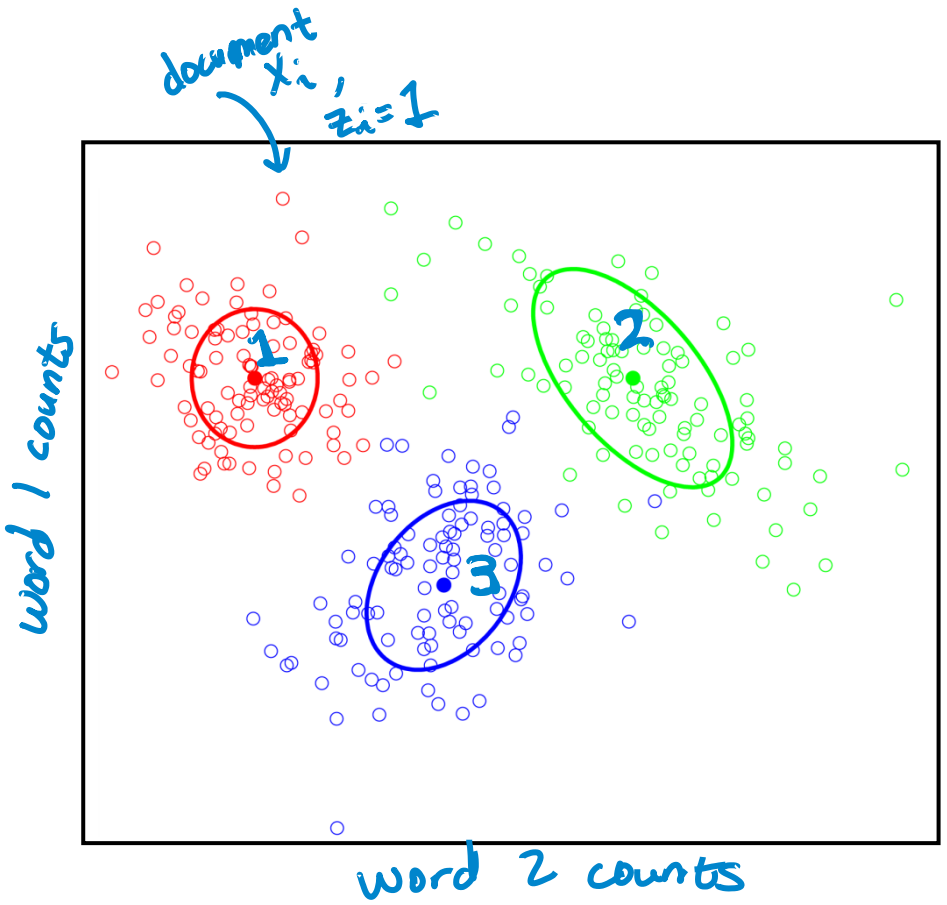
No labels provided

...uncover cluster structure from input alone

Input: docs as vectors $\mathbf{x}_i$

Output: cluster labels z_i

An **unsupervised learning** task

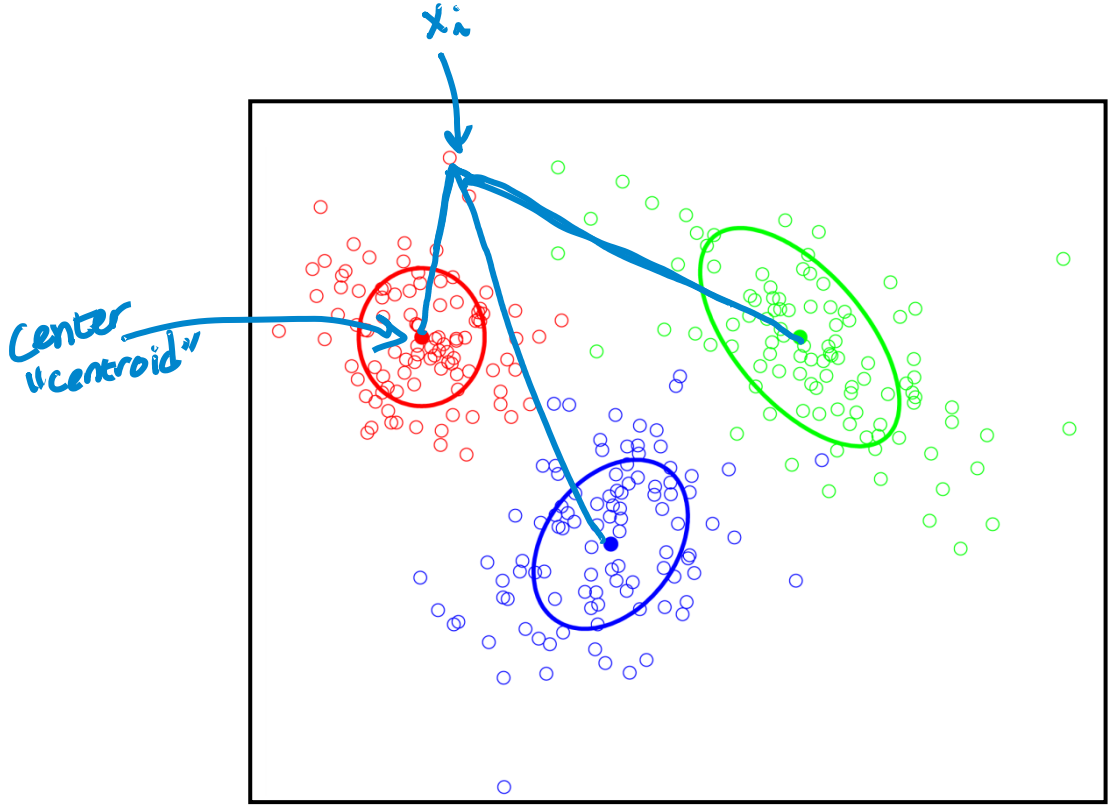


What defines a cluster?

Cluster defined by
center & shape/spread

Assign observation $\mathbf{x}_i$ (doc) to cluster k (topic label) if

- Score under cluster k is higher than under others
- For simplicity, often define score as distance to cluster center (ignoring shape)

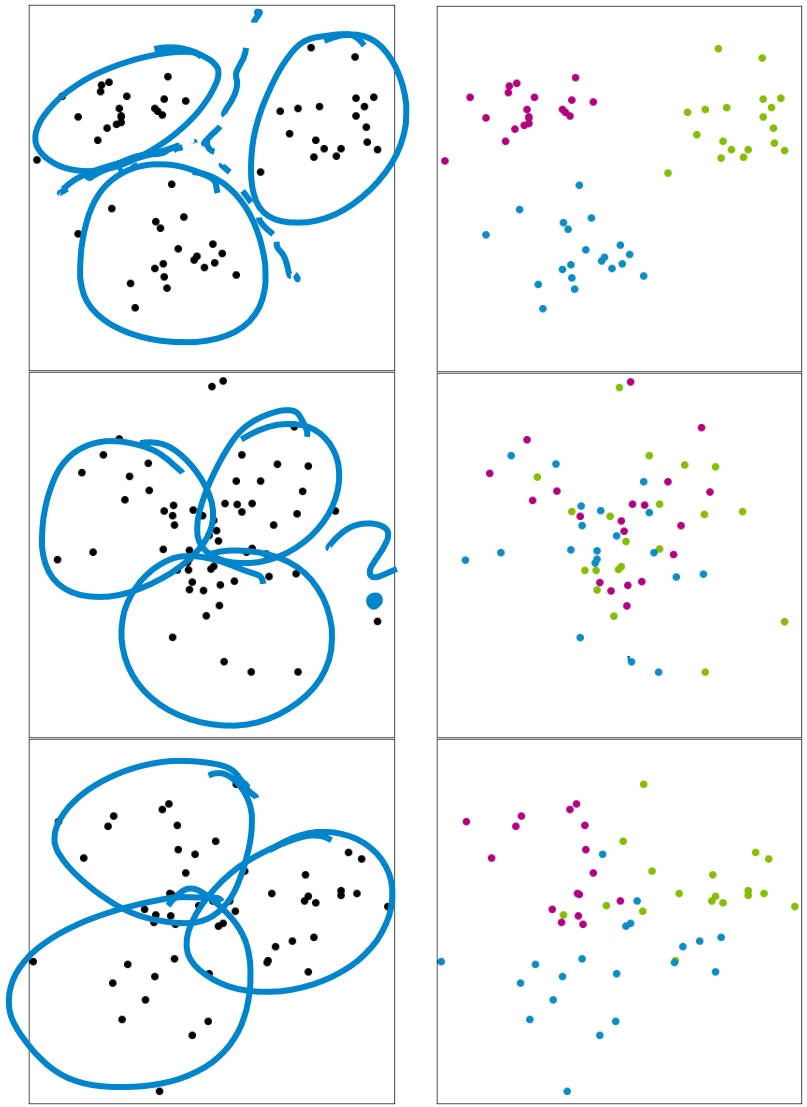


Hope for unsupervised learning

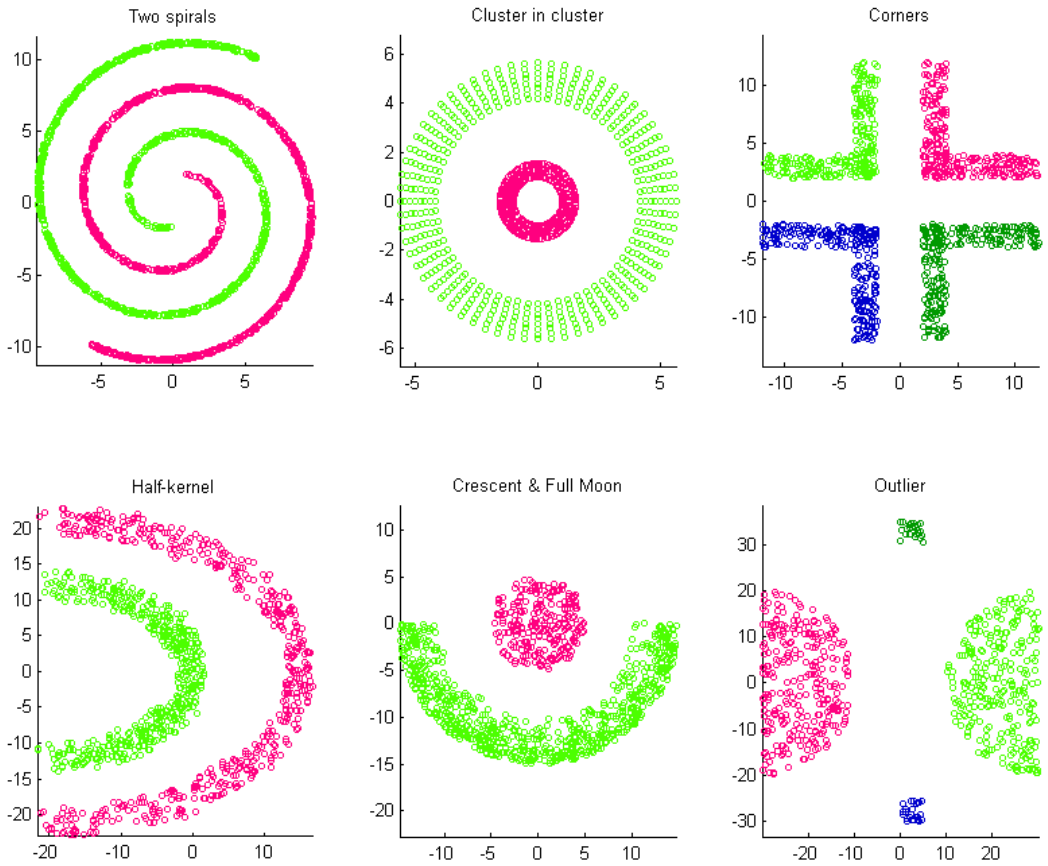
Easy

Impossible

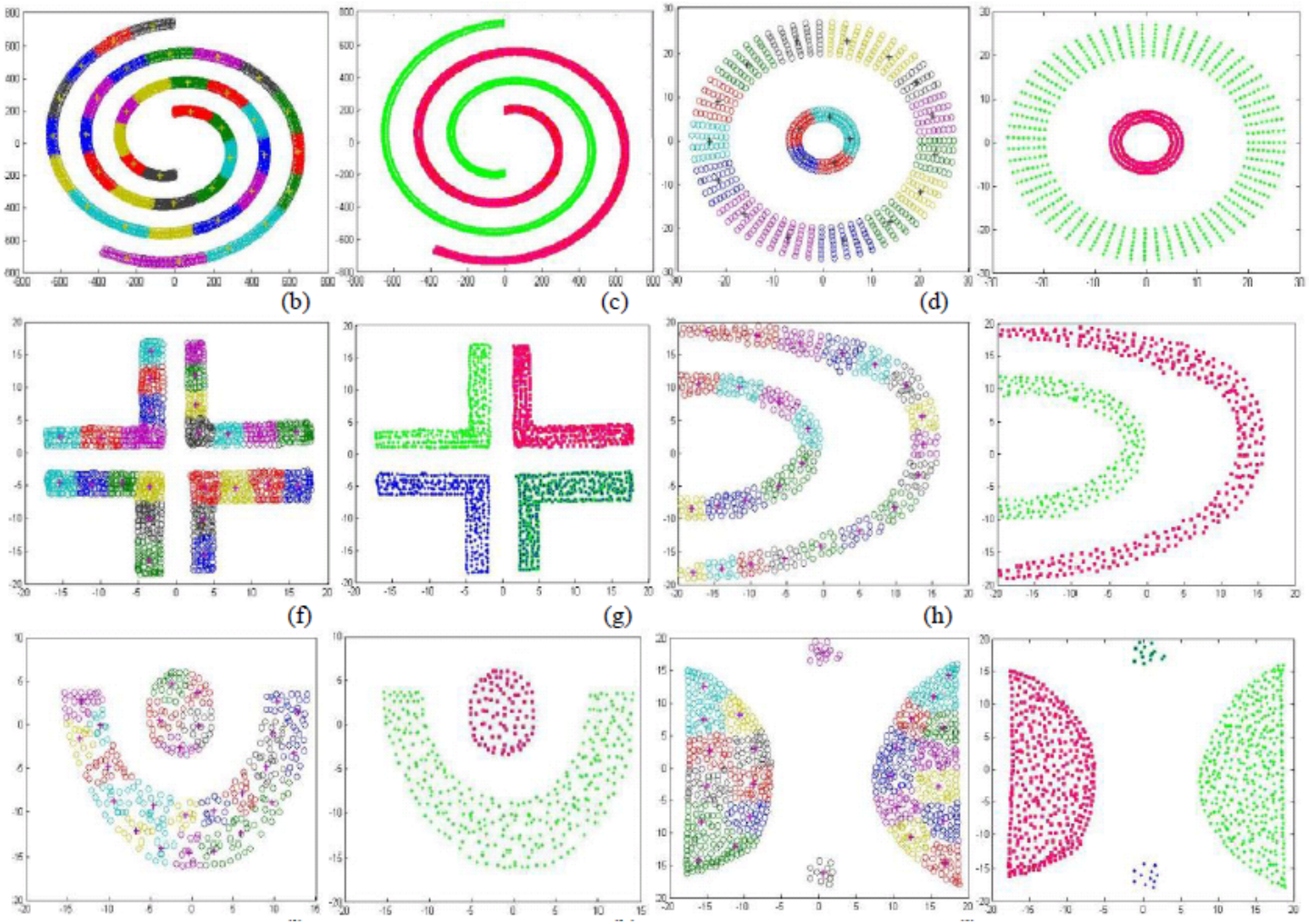
In between



Other (challenging!) clusters to discover…



Other (challenging!) clusters to discover…

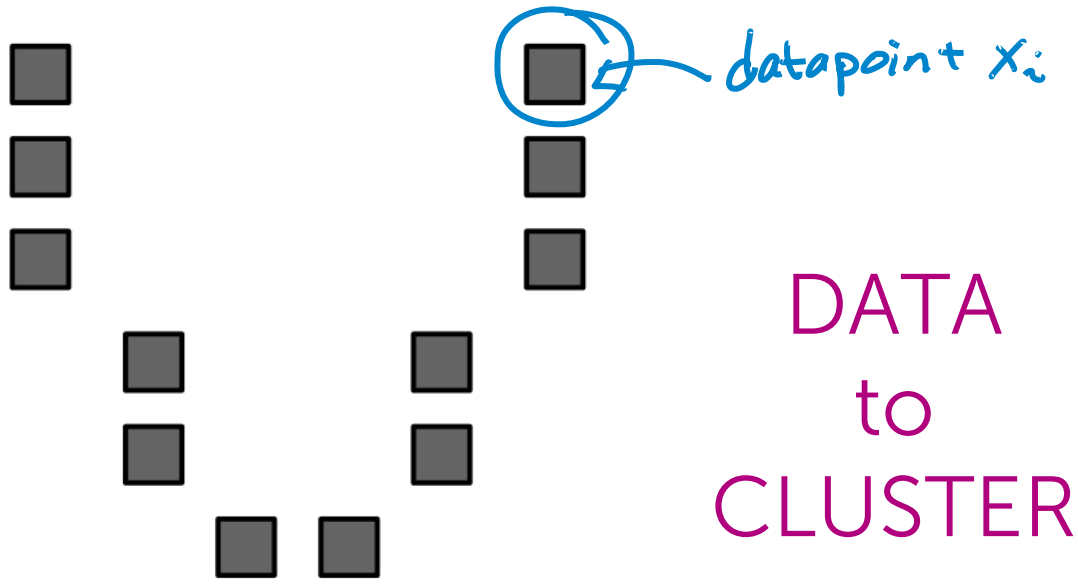


k-means: A clustering algorithm

k-means

Assume

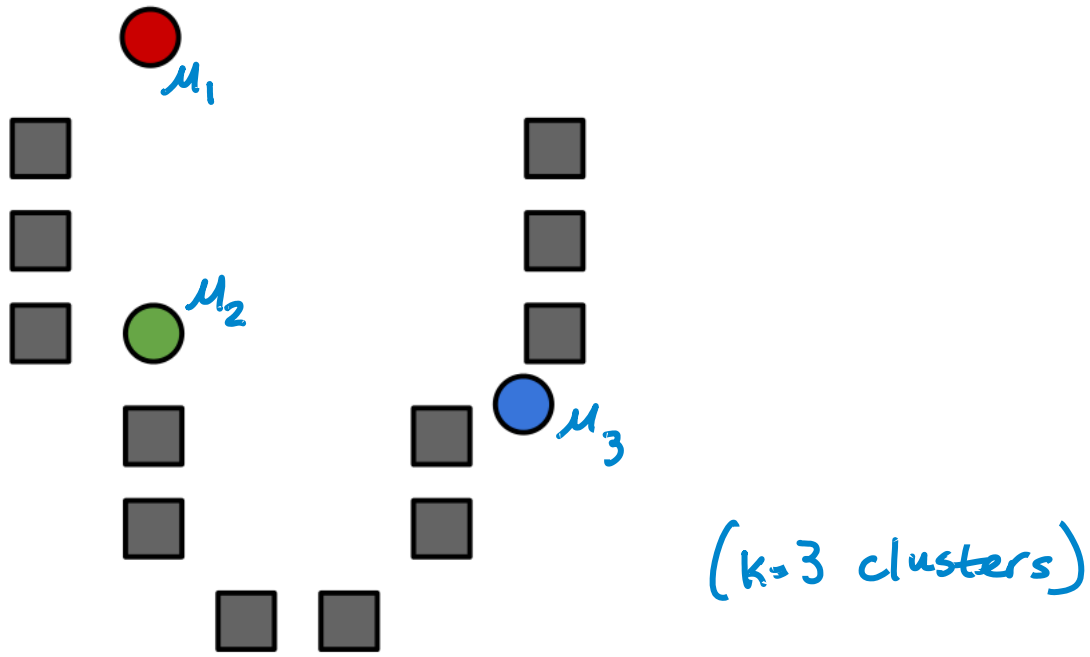
- Score= distance to cluster center (smaller better)



k-means algorithm

0. Initialize cluster centers

$$\mu_1, \mu_2, \ldots, \mu_k$$



k-means algorithm

0. Initialize cluster centers
1. Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

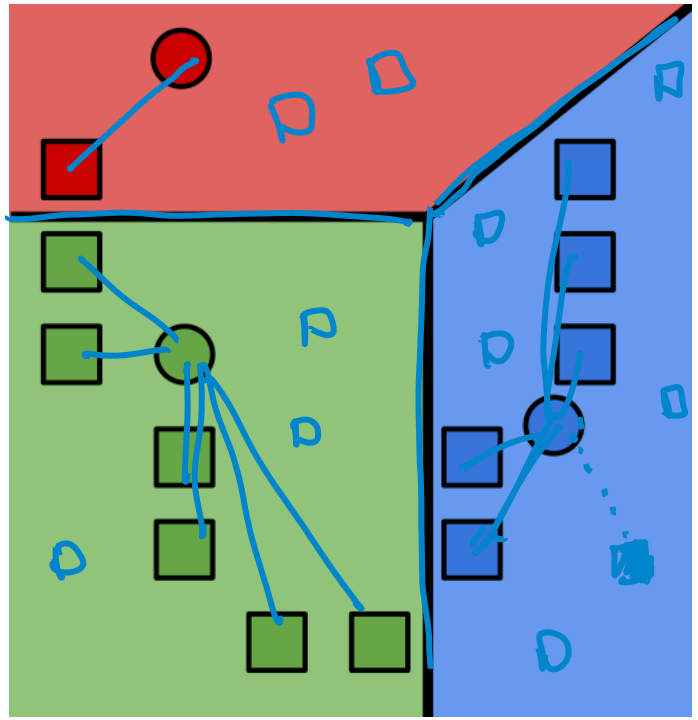
μ_j: jth cluster center (varying)

$\mathbf{x}_i$: ith obs. (fixed)

Inferred label for obs i, whereas supervised learning has given label y_i

return index j of the cluster whose center is closest to obs x_i

(whereas min returning minimum value of $\|\cdot\|_2^2$)



Voronoi tesselation

(for visualization only ... you don't need to compute this)

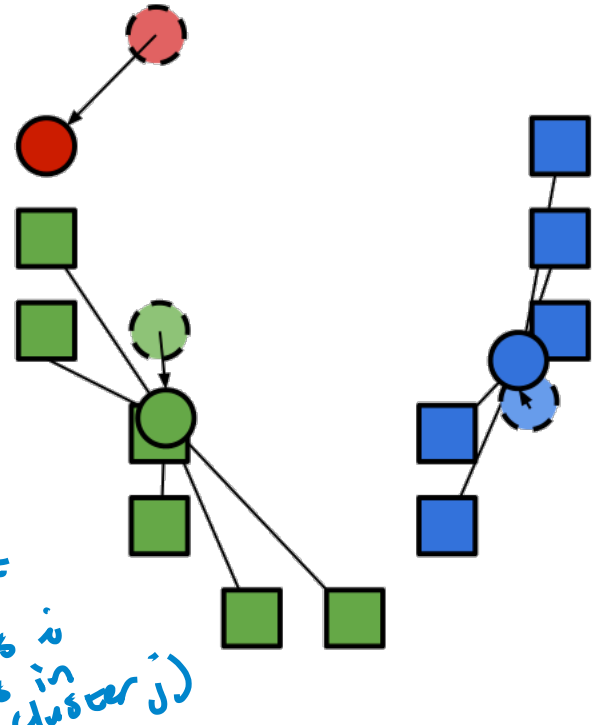
k-means algorithm

0. Initialize cluster centers
1. Assign observations to closest cluster center
2. Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=j} \mathbf{x}_i$$

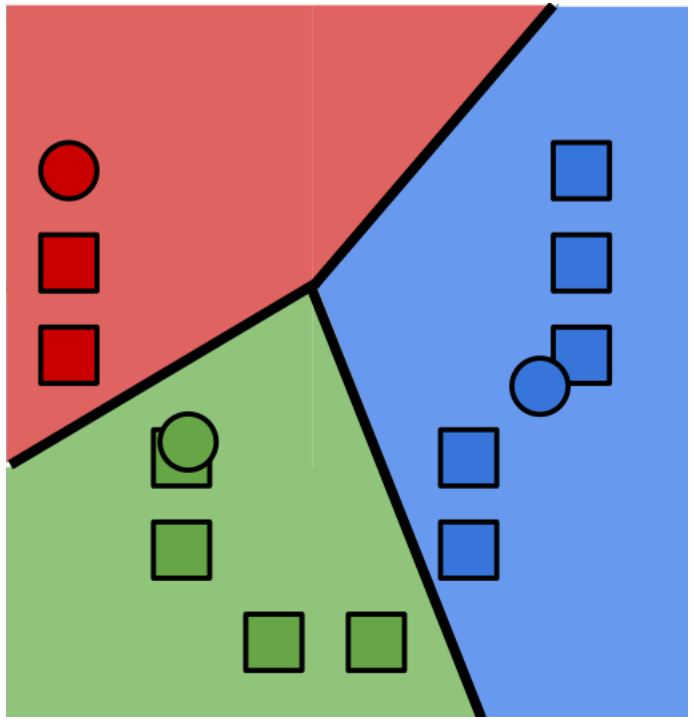
all obs. i such that $z_i = j$ (obs i is in cluster j)

n_j = # of obs. in cluster j



k-means algorithm

0. Initialize cluster centers
1. Assign observations to closest cluster center
2. Revise cluster centers as mean of assigned observations
3. Repeat 1.+2. until convergence



k-means as coordinate descent

A coordinate descent algorithm

1. Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

2. Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=j} \mathbf{x}_i$$

equivalent to

$$\mu_j \leftarrow \arg\min_\mu \sum_{i:z_i=j} ||\mu - \mathbf{x}_i||_2^2$$

A coordinate descent algorithm

1. Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

2. Revise cluster centers as mean of assigned observations

$$\mu_j \leftarrow \arg\min_\mu \sum_{i:z_i=j} ||\mu - \mathbf{x}_i||_2^2$$

A coordinate descent algorithm

1. Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

2. Revise cluster centers as mean of assigned observations

$$\mu_j \leftarrow \arg\min_\mu \sum_{i:z_i=j} ||\mu - \mathbf{x}_i||_2^2$$

Alternating minimization
1. (z given μ) and 2. (μ given z)
= **coordinate descent**

Convergence of k-means

Converges to:

- ~~Global optimum~~
- Local optimum
- ~~neither~~

Smart initialization with k-means++

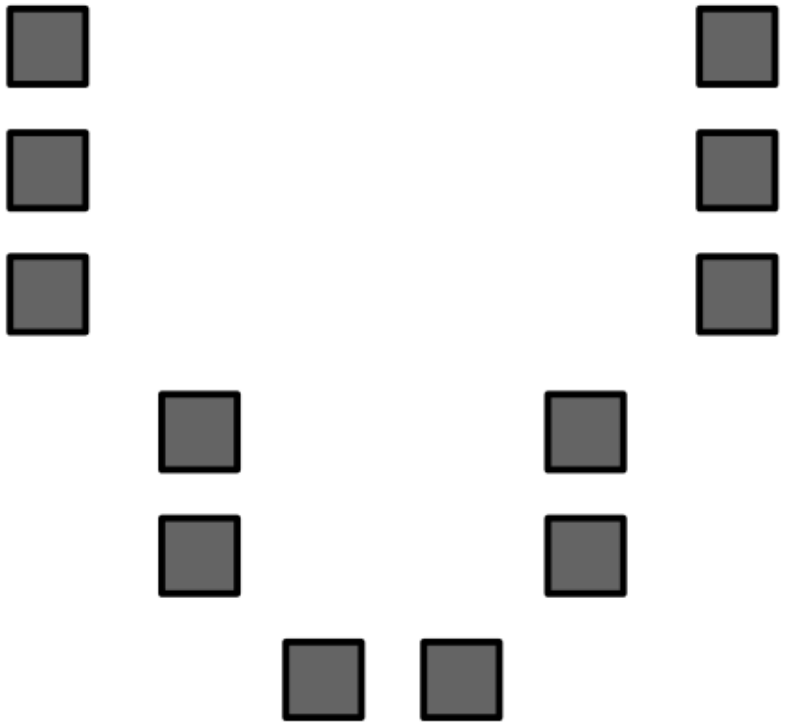
k-means++ overview

Initialization of k-means algorithm is critical to quality of local optima found

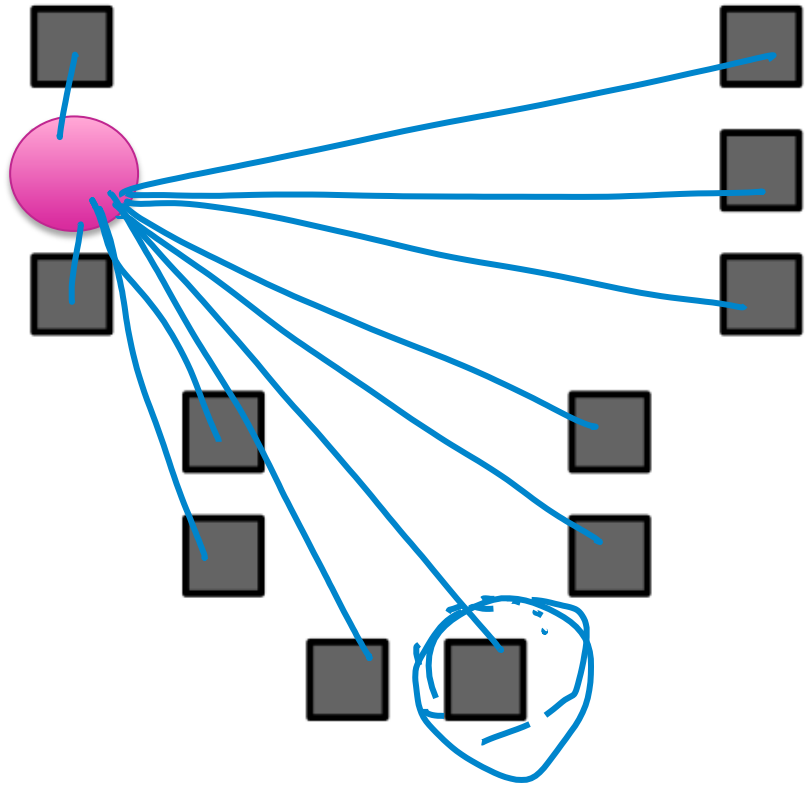
Smart initialization:

1. Choose first cluster center uniformly at random from data points
2. For each obs **x**, compute distance d(**x**) to nearest cluster center
3. Choose new cluster center from amongst data points, with probability of **x** being chosen proportional to $d(\mathbf{x})^2$
4. Repeat Steps 2 and 3 until k centers have been chosen

k-means++ visualized

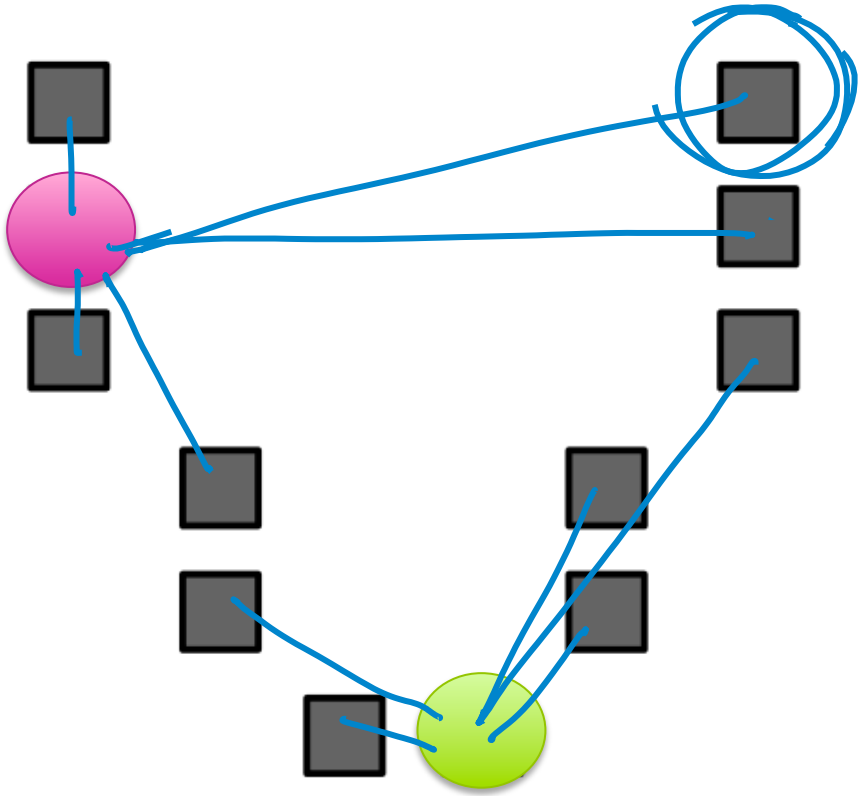


k-means++ visualized

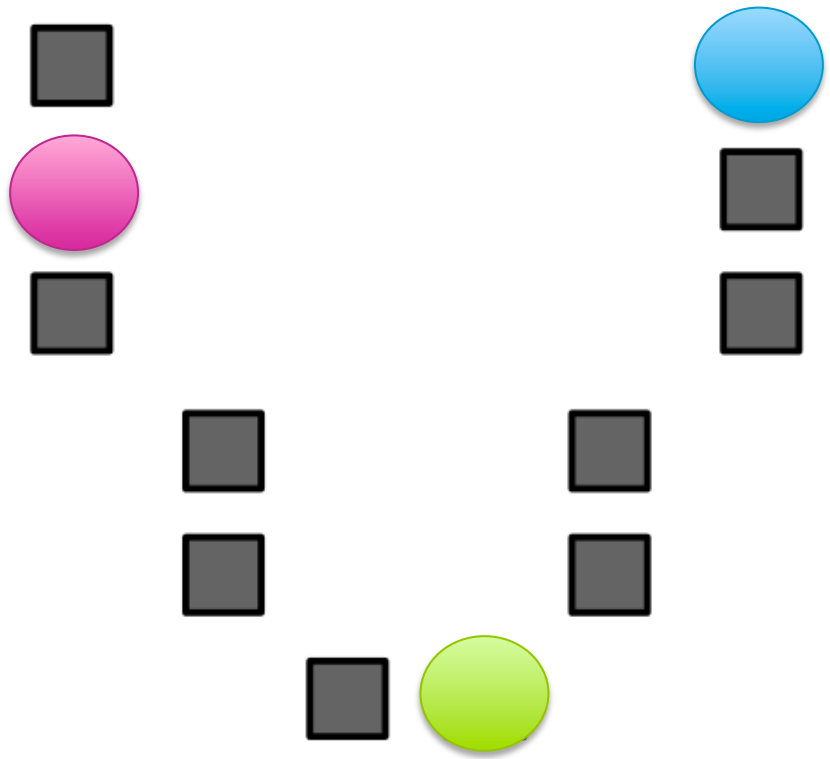


more likely to select a datapoint as a cluster center if that datapoint is far away ($dist^2$ increases this effect)

k-means++ visualized



k-means++ visualized



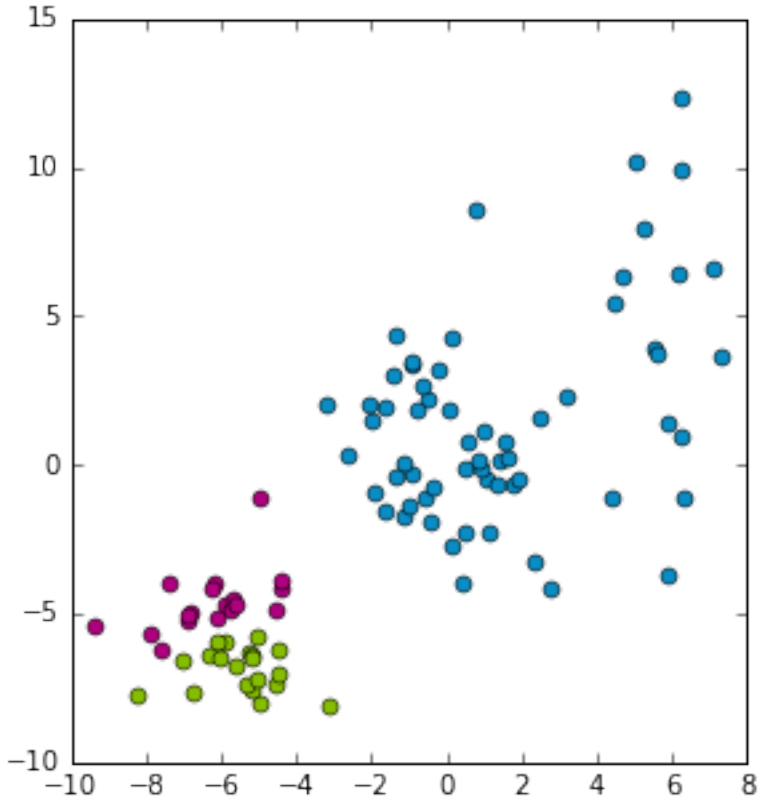
k-means++ pros/cons

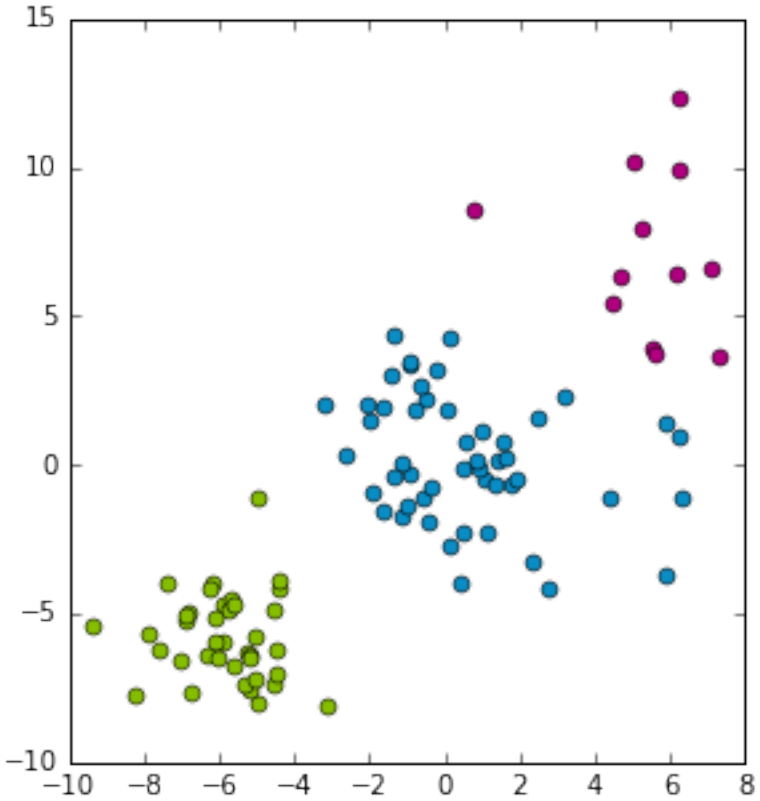
Computationally costly relative to random initialization, but the subsequent k-means often converges more rapidly

Tends to improve quality of local optimum and lower runtime

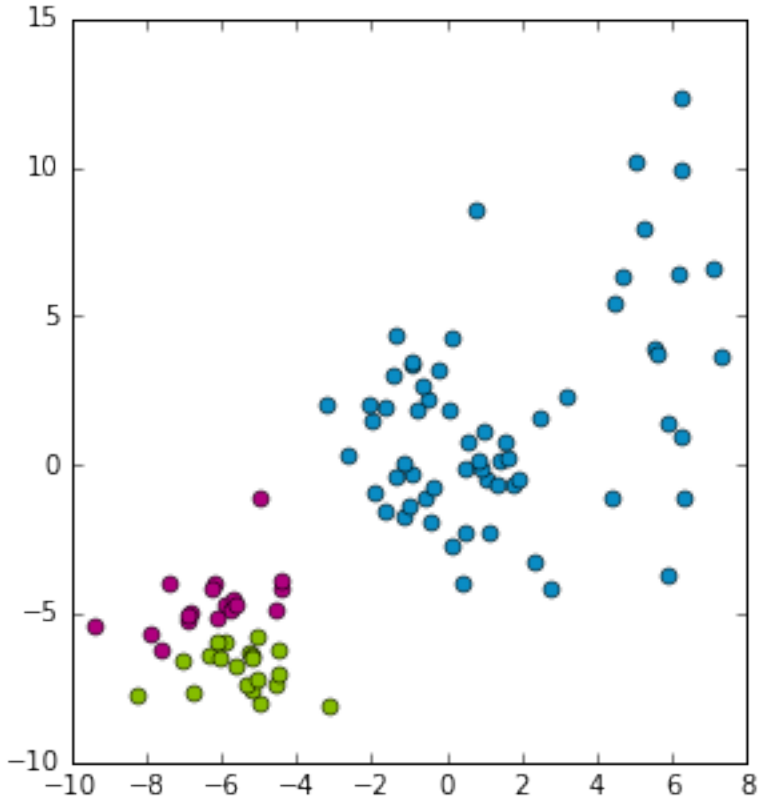
Assessing quality of the clustering and choosing the # of clusters

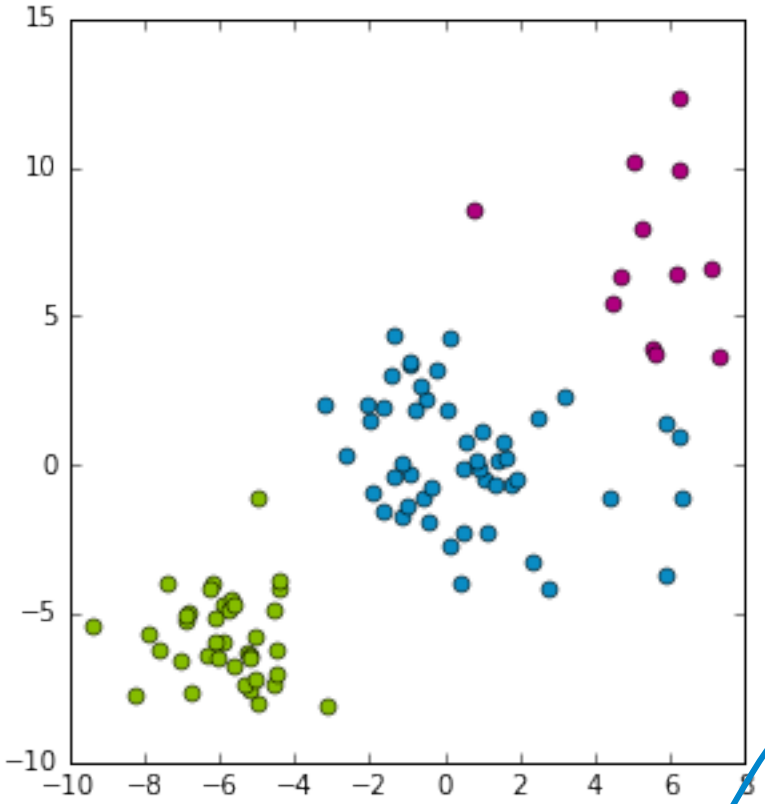
Which clustering do I prefer?





k-means objective





k-means is trying to minimize the sum of squared distances:

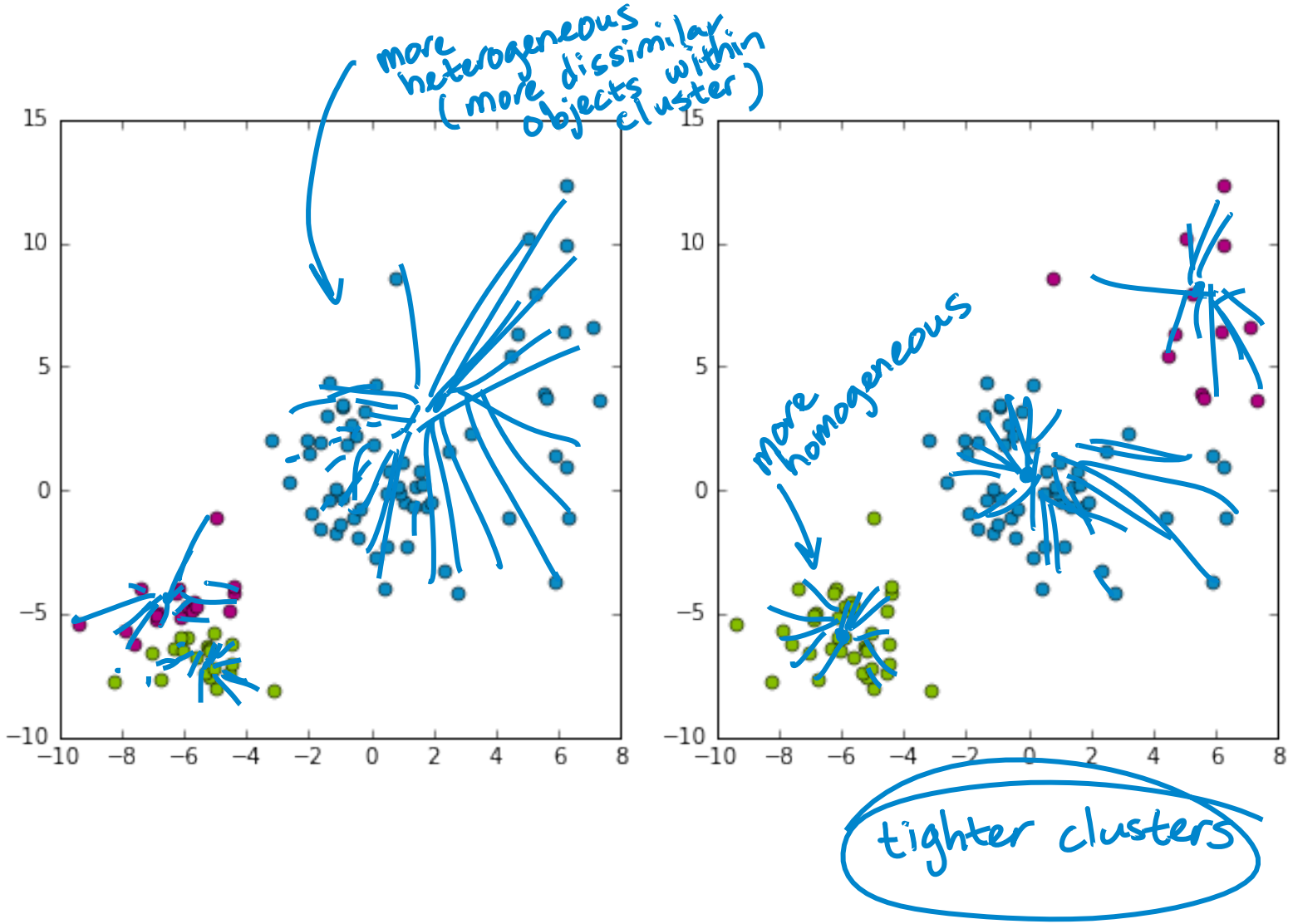
sum over all clusters

sum of squared distances in cluster j

$$\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - \mathbf{x}_i||_2^2$$

$$\min_{\{z_i\}, \{\mu_j\}}$$

Cluster heterogeneity



Measure of quality of given clustering:

$$\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - \mathbf{x}_i||_2^2$$

Lower is better!

What happens as k increases?

Can refine clusters more and more to the data

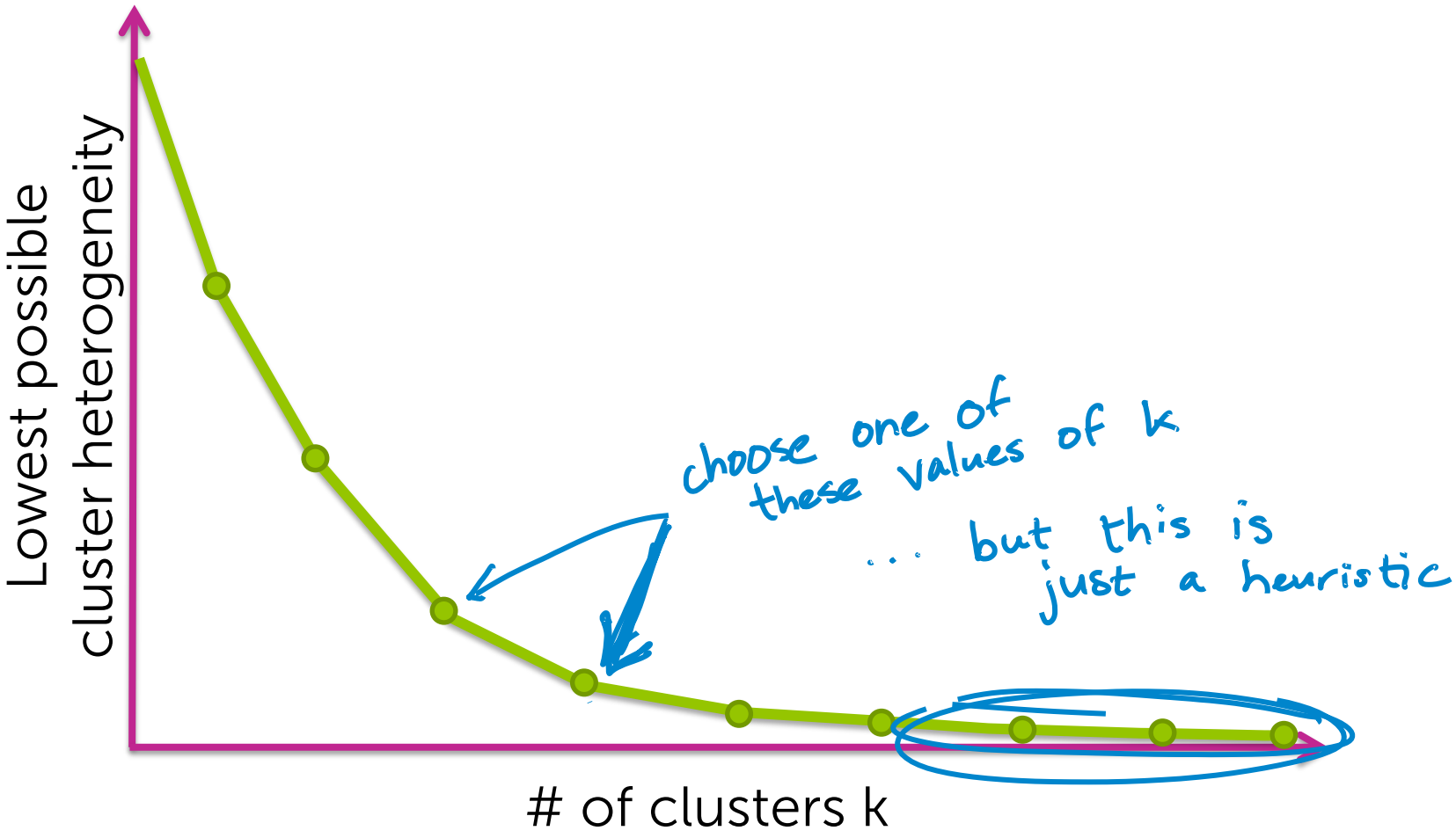
→ overfitting!

Extreme case of k=N: (N: # of observations)

- can set each cluster center equal to datapoint
- heterogeneity = 0 ! (all distances to cluster centers are 0)

Lowest possible cluster heterogeneity decreases with increasing k

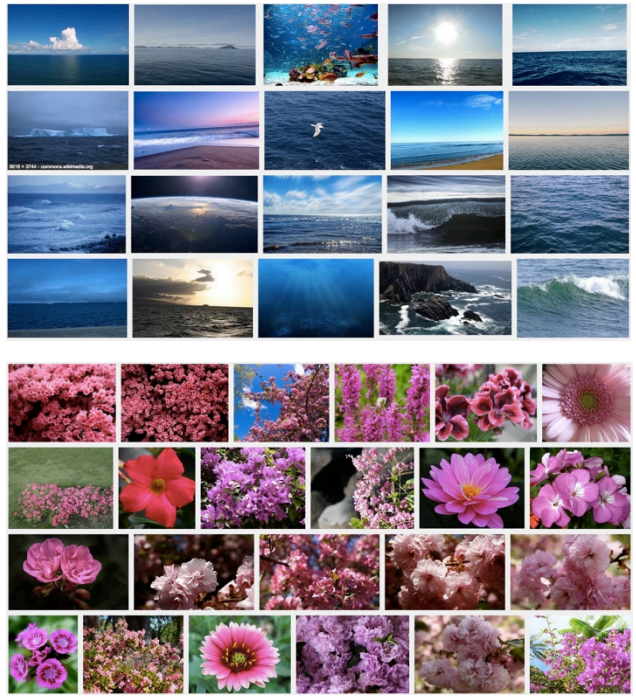
How to choose k?

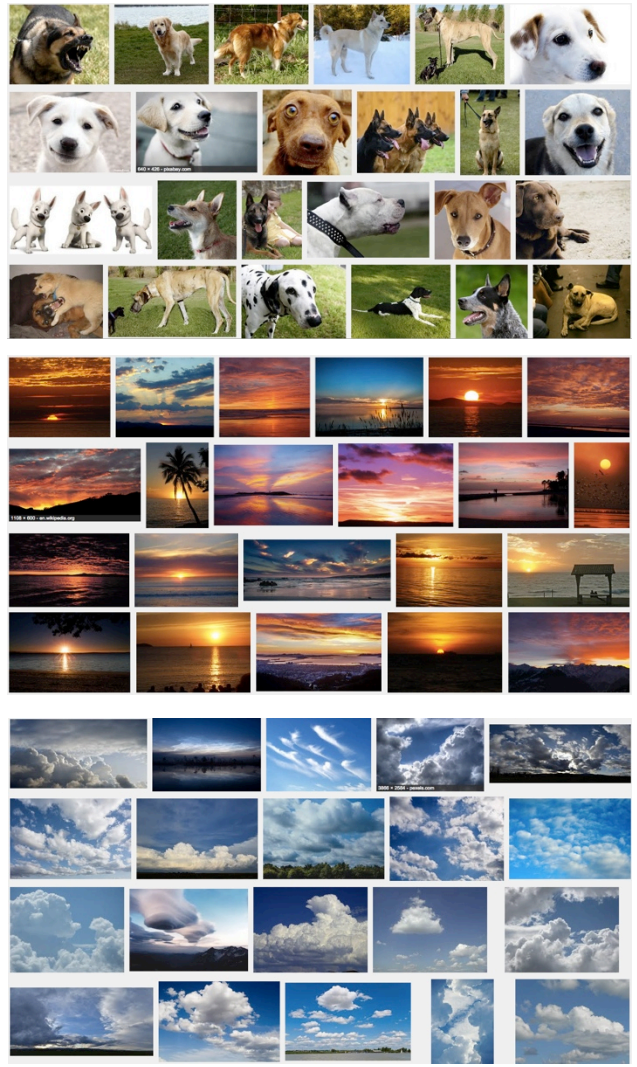


Other examples

Clustering images

- For search, group as:
  - Ocean
  - Pink flower
  - Dog
  - Sunset
  - Clouds
  - …





Structuring web search results

- Search terms can have multiple meanings
- Example: “cardinal”

- Use clustering to structure output

Grouping patients by medical condition

- Better characterize subpopulations and diseases

Products on Amazon

- Discover product categories from purchase histories

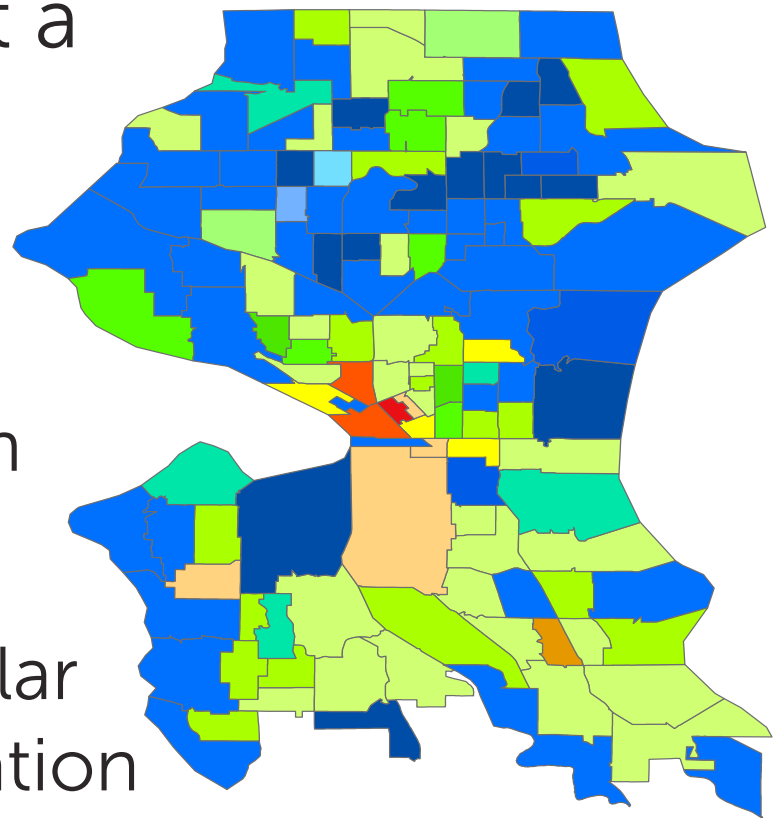
~~"furniture"~~

"baby"

- Or discovering groups of **users**

Discovering similar neighborhoods

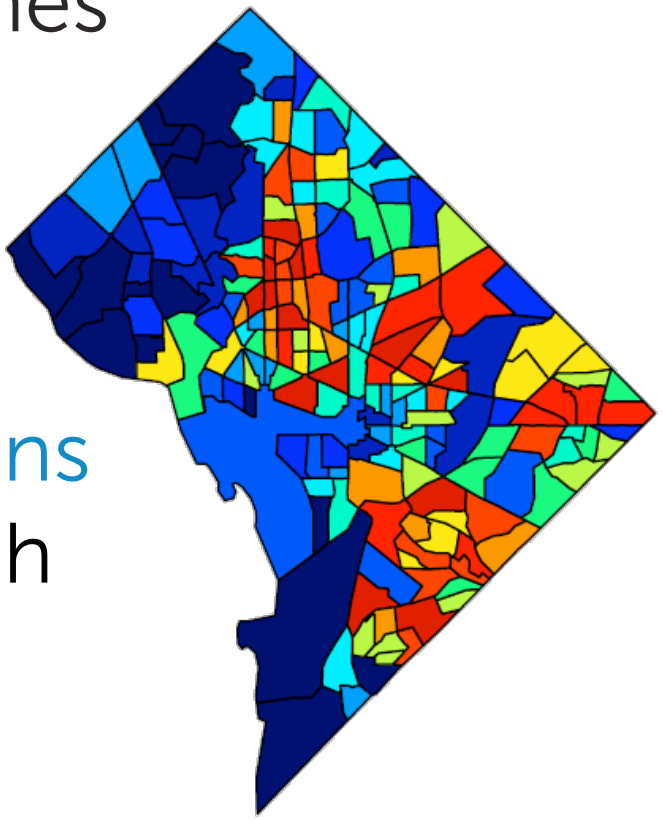
- **Task 1:** Estimate price at a small regional level
- Challenge:
  - Only a few (or no!) sales in each region per month
- Solution:
  - Cluster regions with similar trends and share information within a cluster



City of Seattle

Discovering similar neighborhoods

- **Task 2:** Forecast violent crimes to better task police
- Again, cluster regions and share information!
- Leads to improved predictions compared to examining each region independently



Washington, DC

Summary for k-means

What you can do now...

- Describe potential applications of clustering
- Describe the input (unlabeled observations) and output (labels) of a clustering algorithm
- Determine whether a task is supervised or unsupervised
- Cluster documents using k-means
- Interpret k-means as a coordinate descent algorithm