Clustering:
Grouping Related Docs




Emily Fox & Carlos Guestrin

Machine Learning Specialization

University of Washington

Motivating clustering approaches

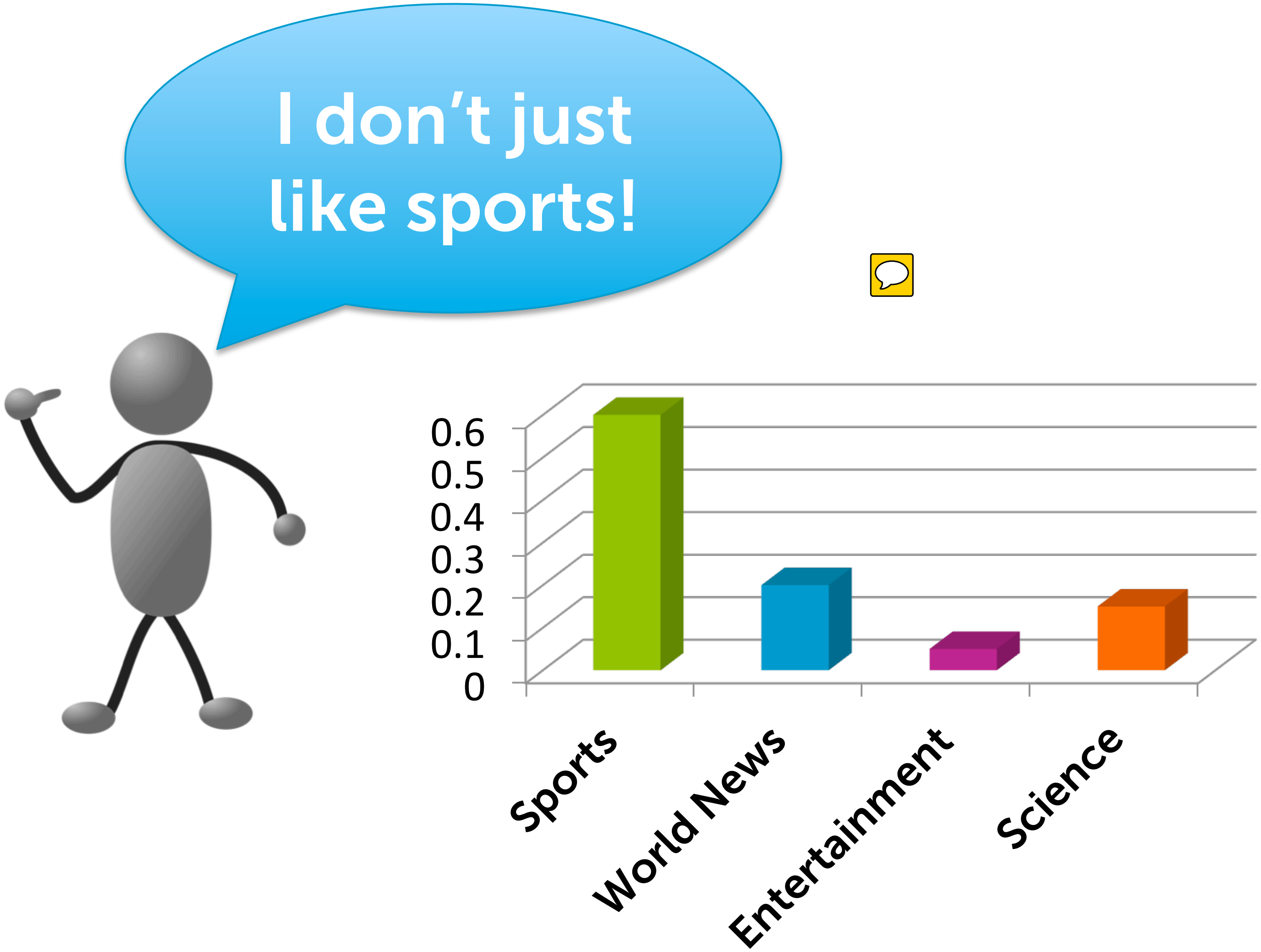
Goal: Structure documents by topic

Discover groups (*clusters*) of related articles

SPORTS

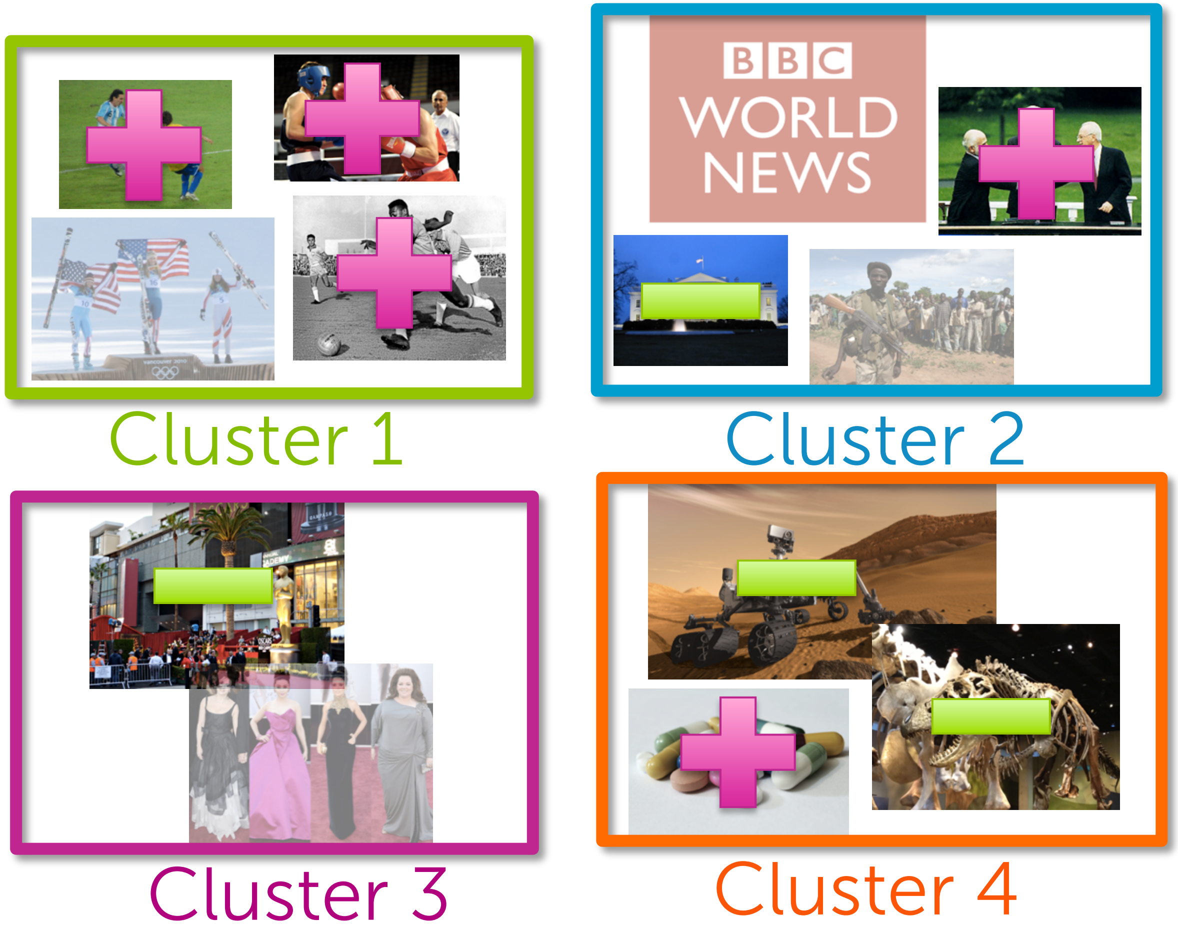
WORLD NEWS

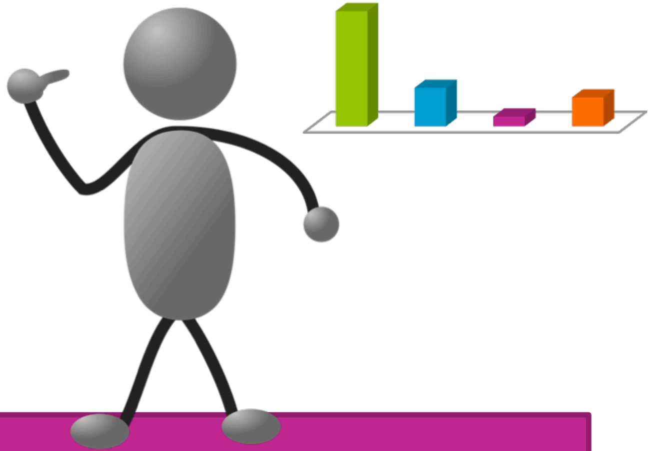
Why might clustering be useful?



Learn user preferences

Set of clustered documents read by user





Use feedback to learn user preferences over topics

Clustering: An unsupervised learning task

What if some of the labels are known?

Training set of labeled docs

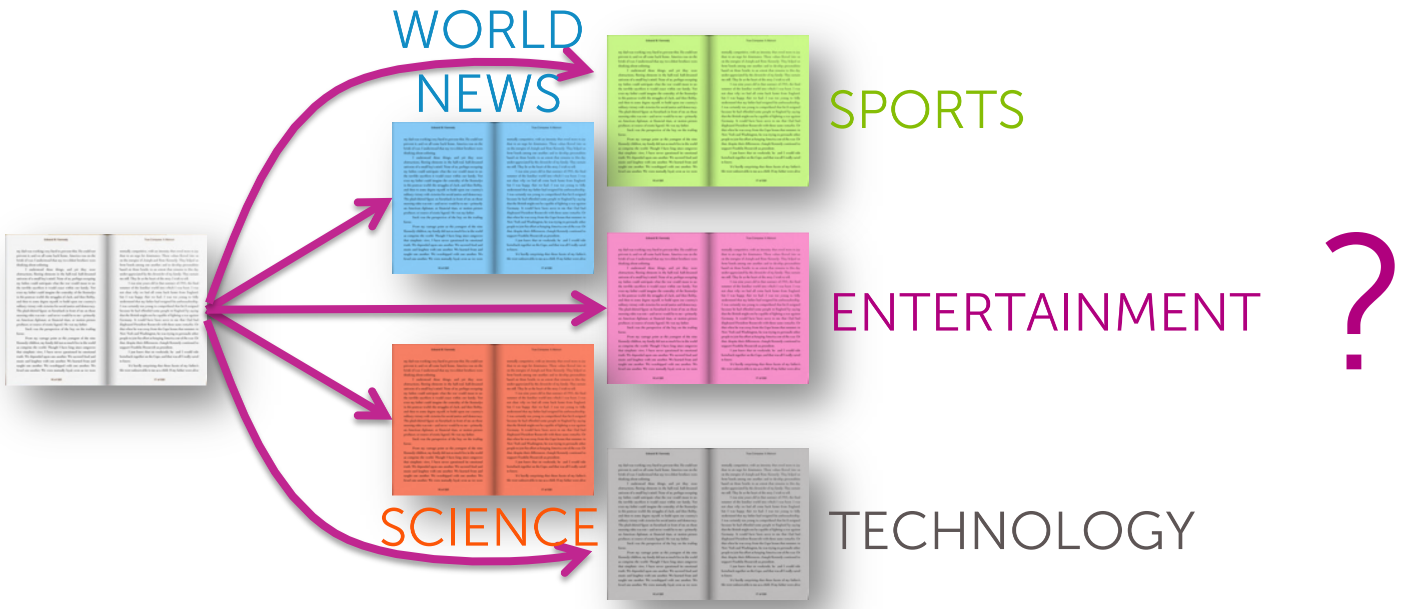
SPORTS

WORLD NEWS

ENTERTAINMENT

SCIENCE

Multiclass classification problem



Example of
supervised learning

Clustering

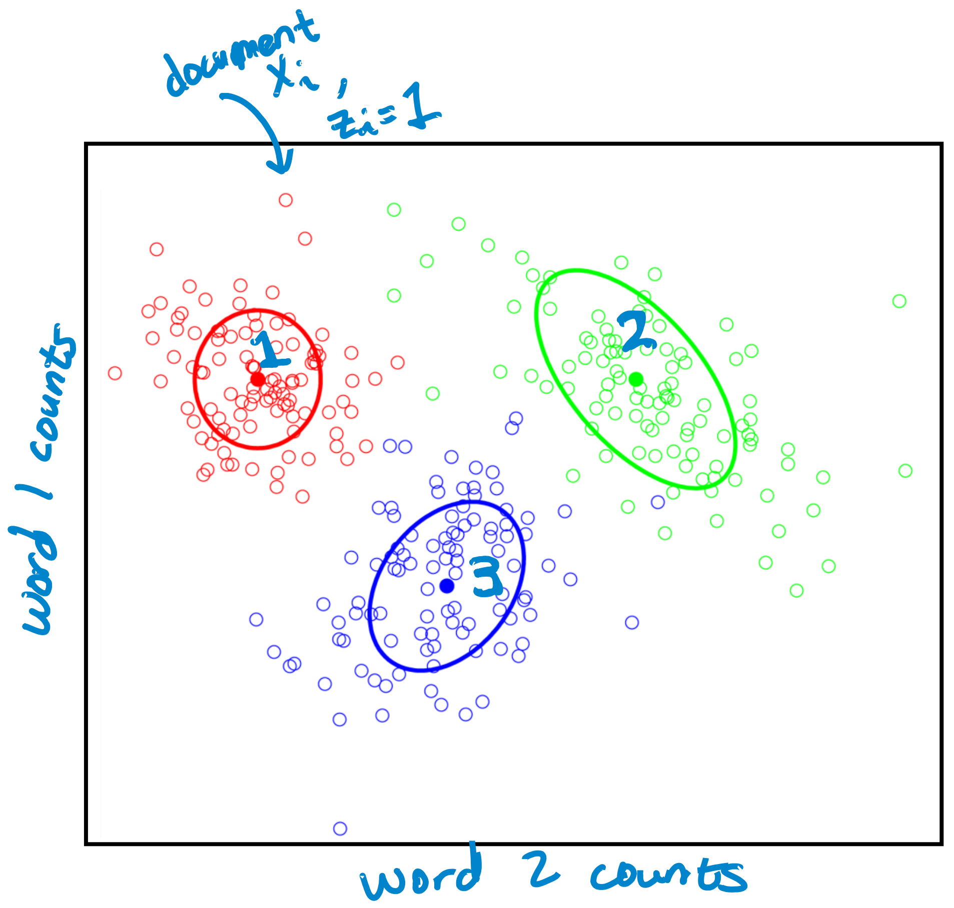
No labels provided

...uncover cluster structure from input alone

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

Output: cluster labels z_i

An **unsupervised learning** task

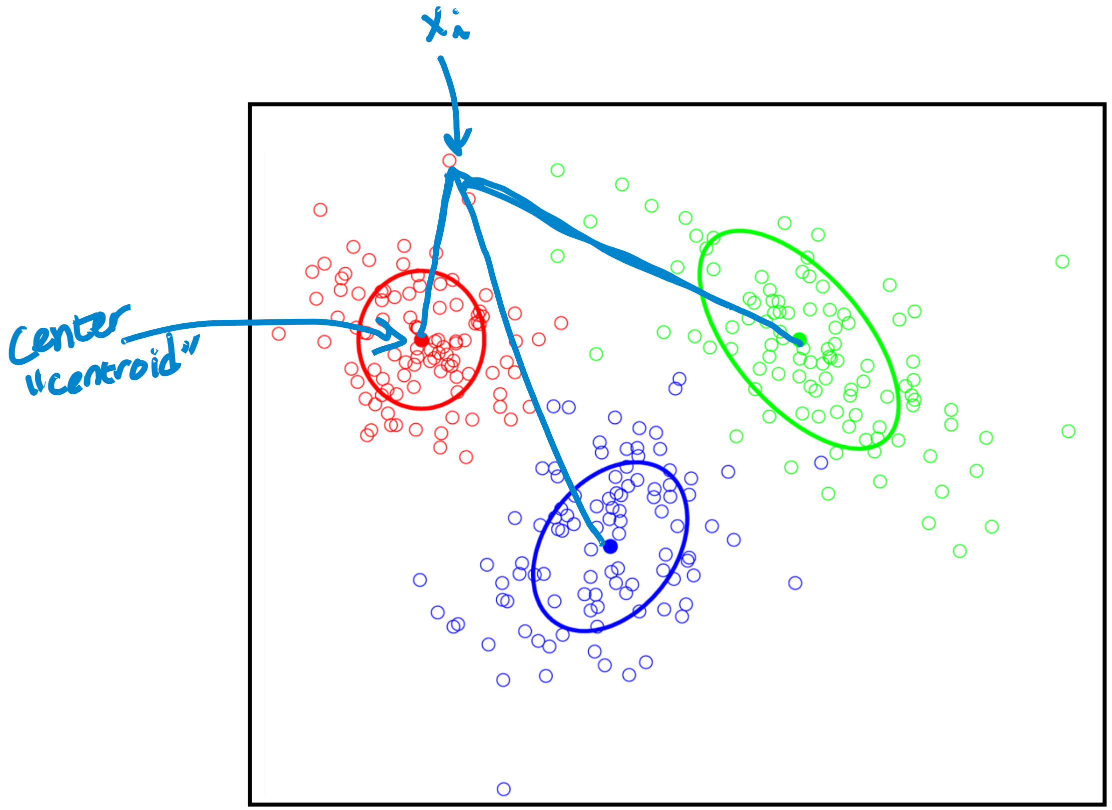


What defines a cluster?

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)

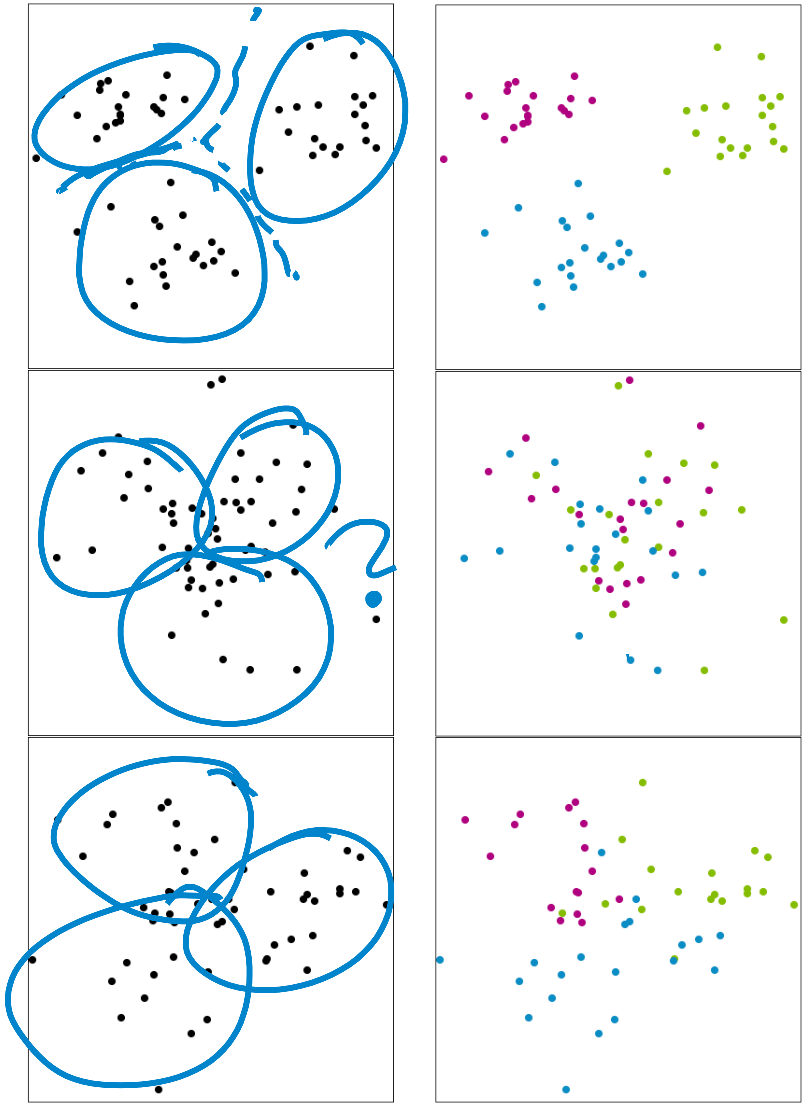


Hope for unsupervised learning

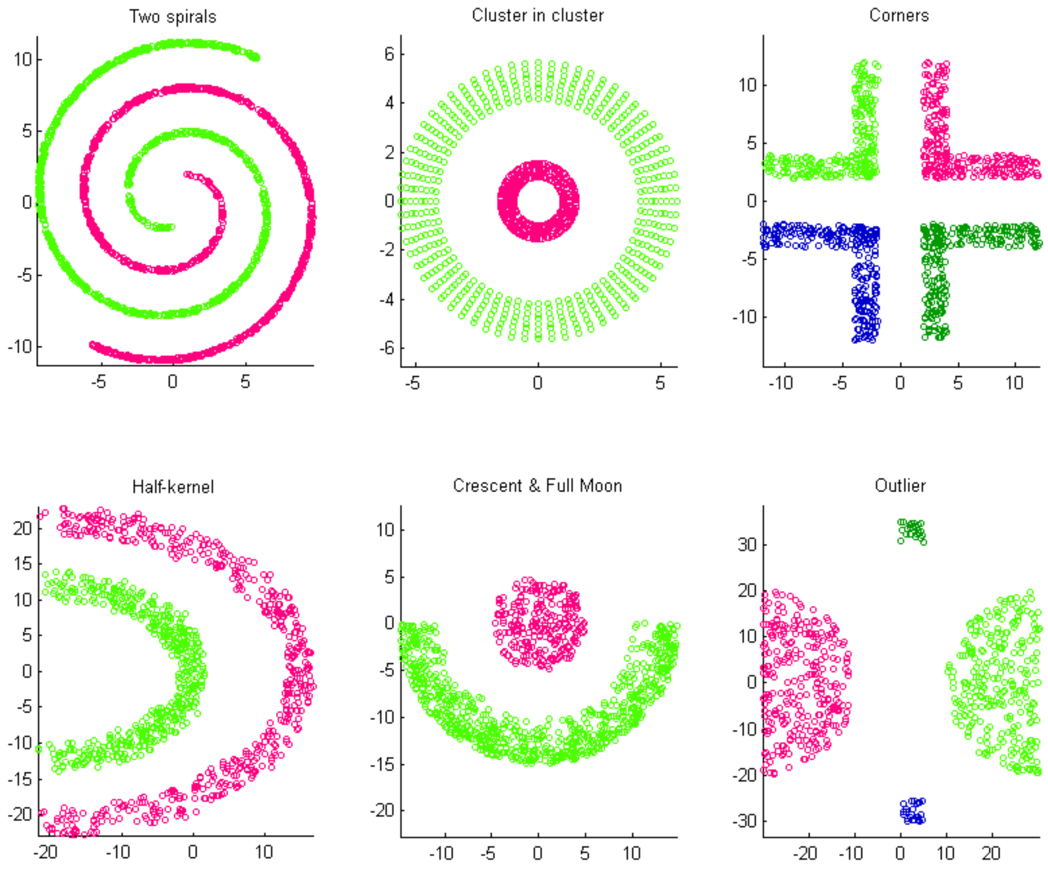
Easy

Impossible

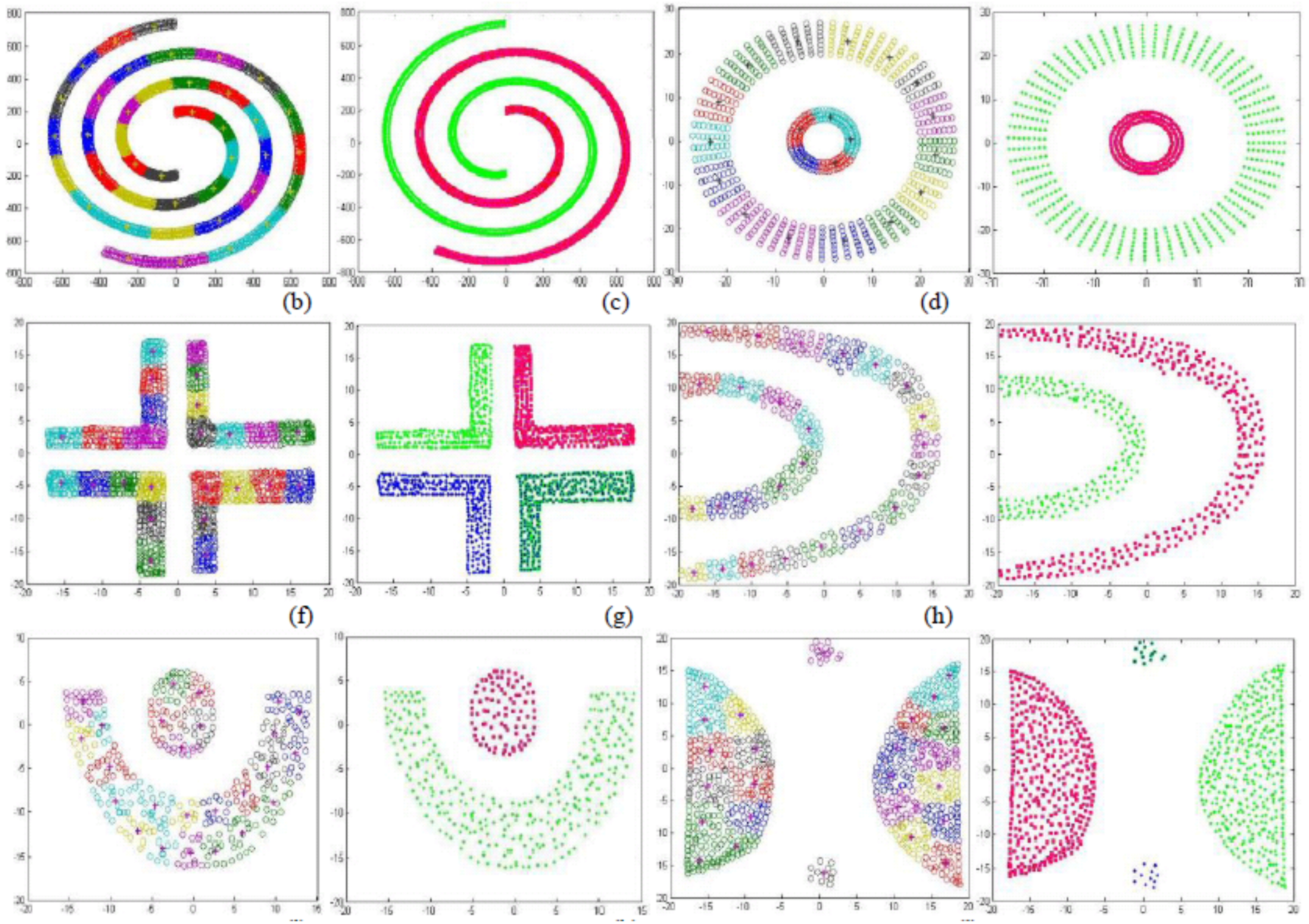
In between



Other (challenging!) clusters to discover…



Other (challenging!) clusters to discover…

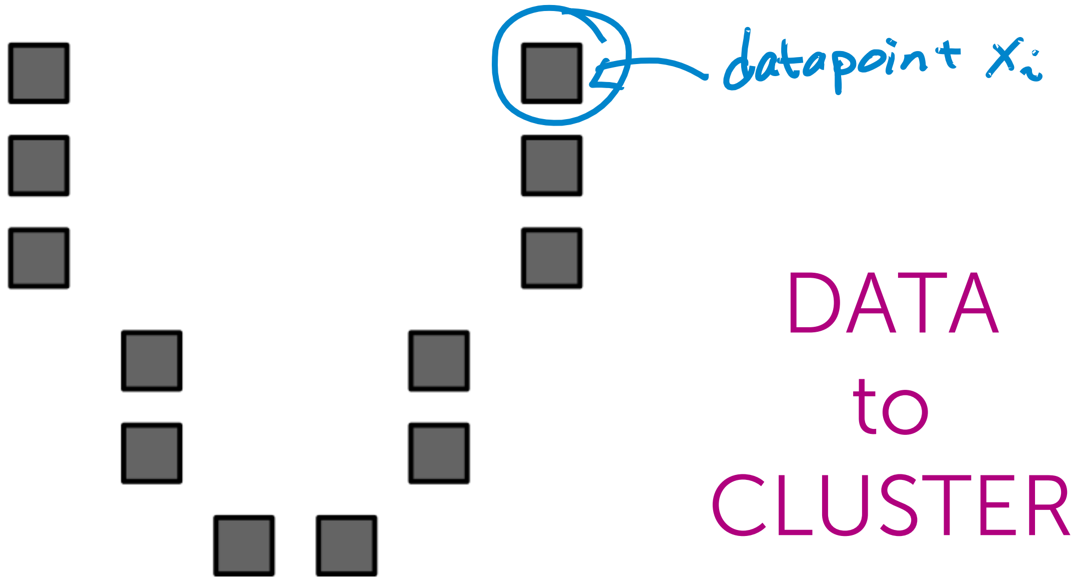


k-means: A clustering algorithm

k-means

Assume

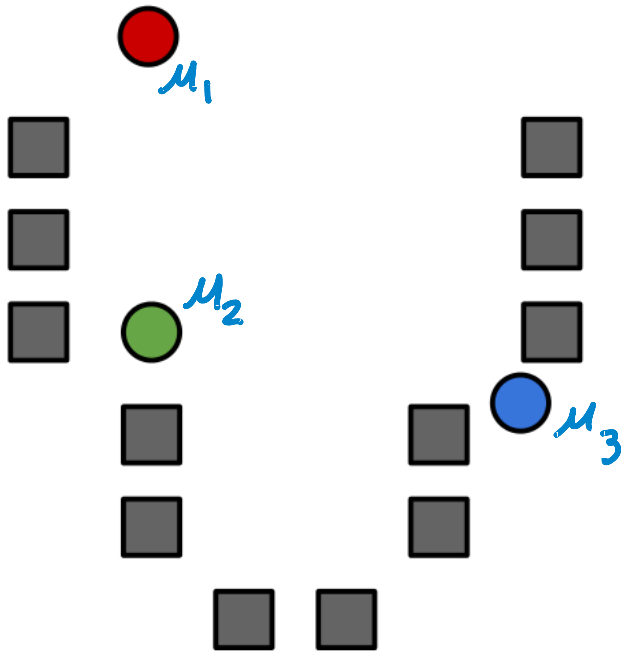
- Score= distance to cluster center (smaller better)



k-means algorithm

0. Initialize cluster centers

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



(k=3 clusters)

k-means algorithm

0. Initialize cluster centers

1. Assign observations to closest cluster center

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

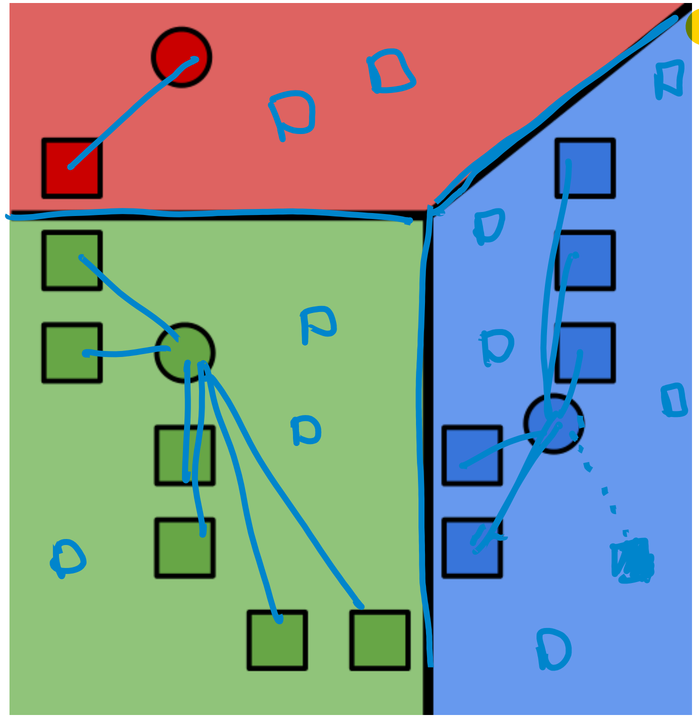
jth cluster center (varying)

ith obs. (fixed)

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

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

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



Voronoi tesselation

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

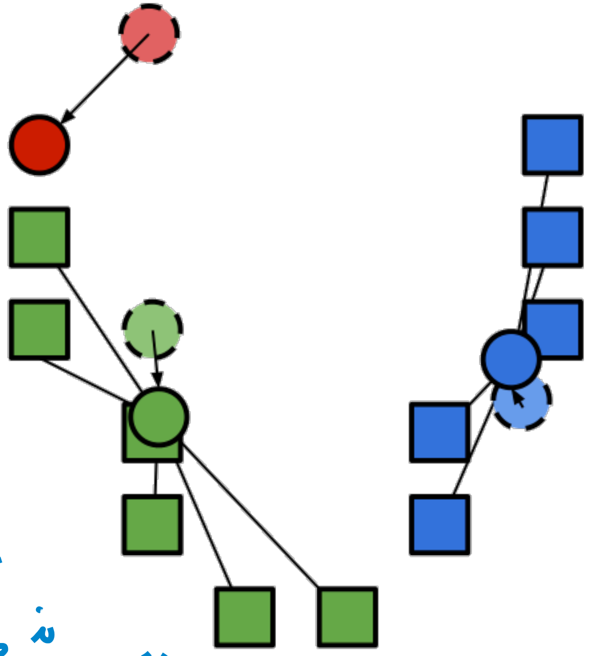
k-means algorithm

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

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

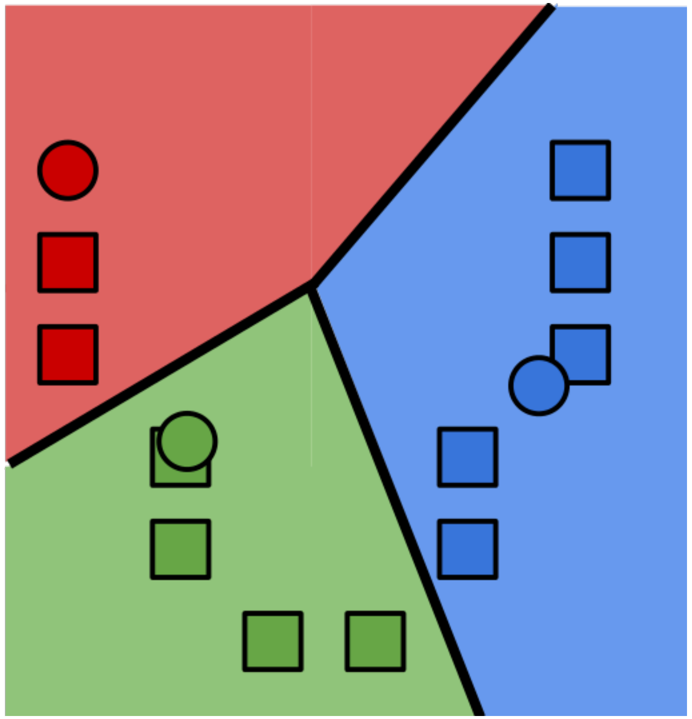
all obs. i such that $z_i = j$ (obs i is in cluster j)

# 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

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

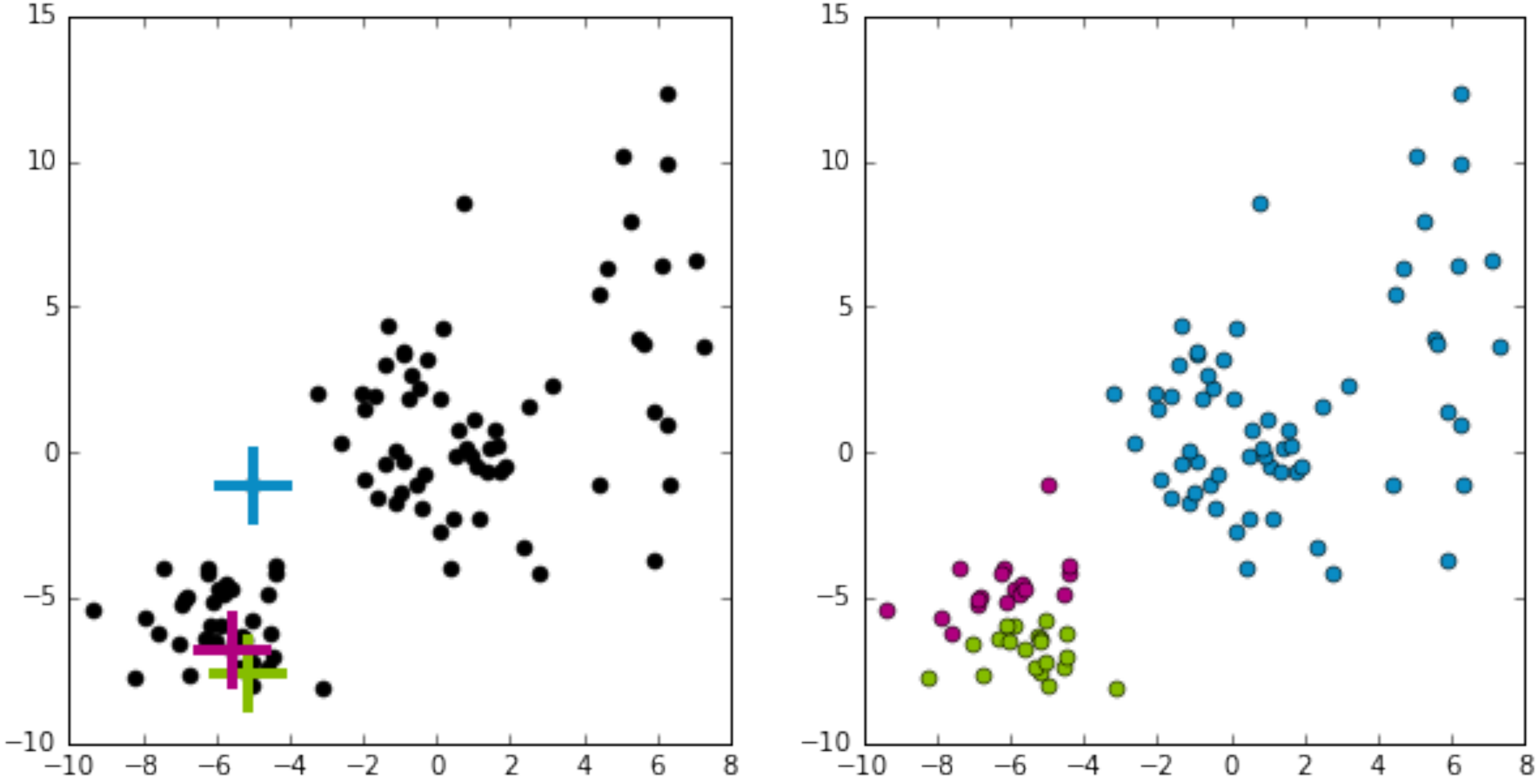
Alternating minimization
1. (z given μ) and 2. (μ given z)
= **coordinate descent**

Convergence of k-means

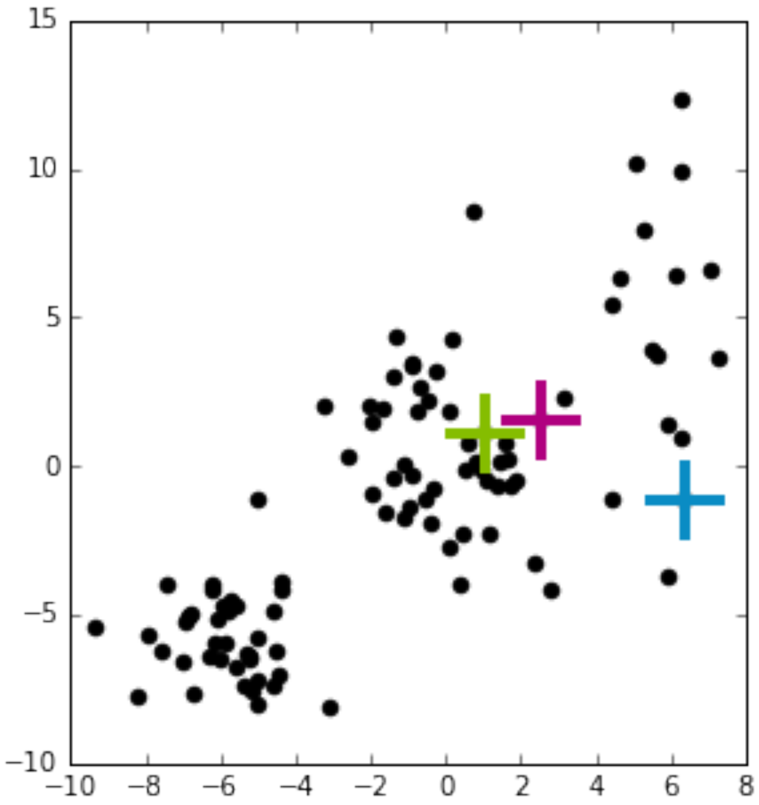
Converges to:

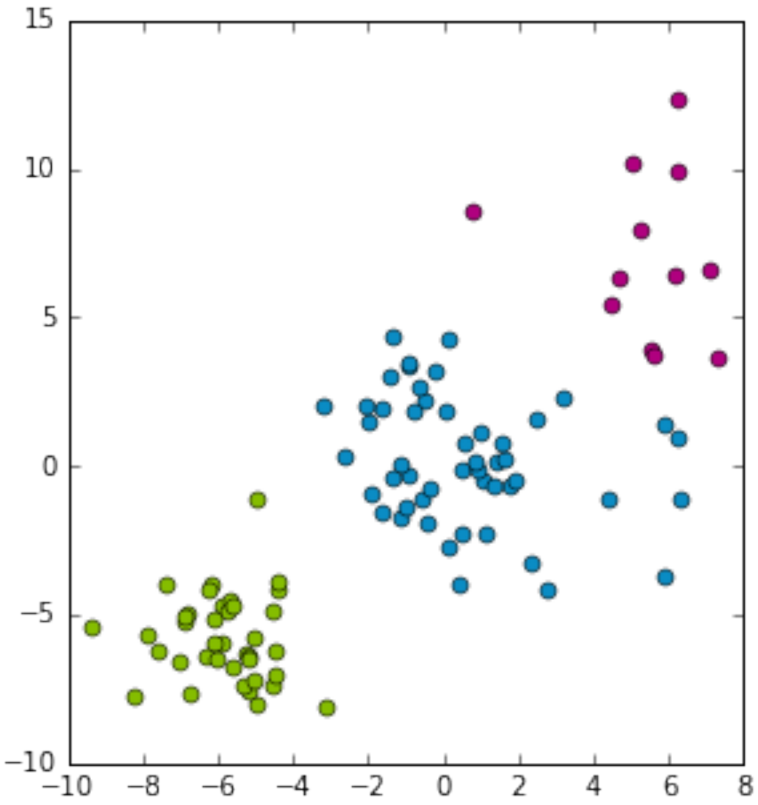
- ~~Global optimum~~
- Local optimum
- ~~neither~~

Convergence of k-means to local mode

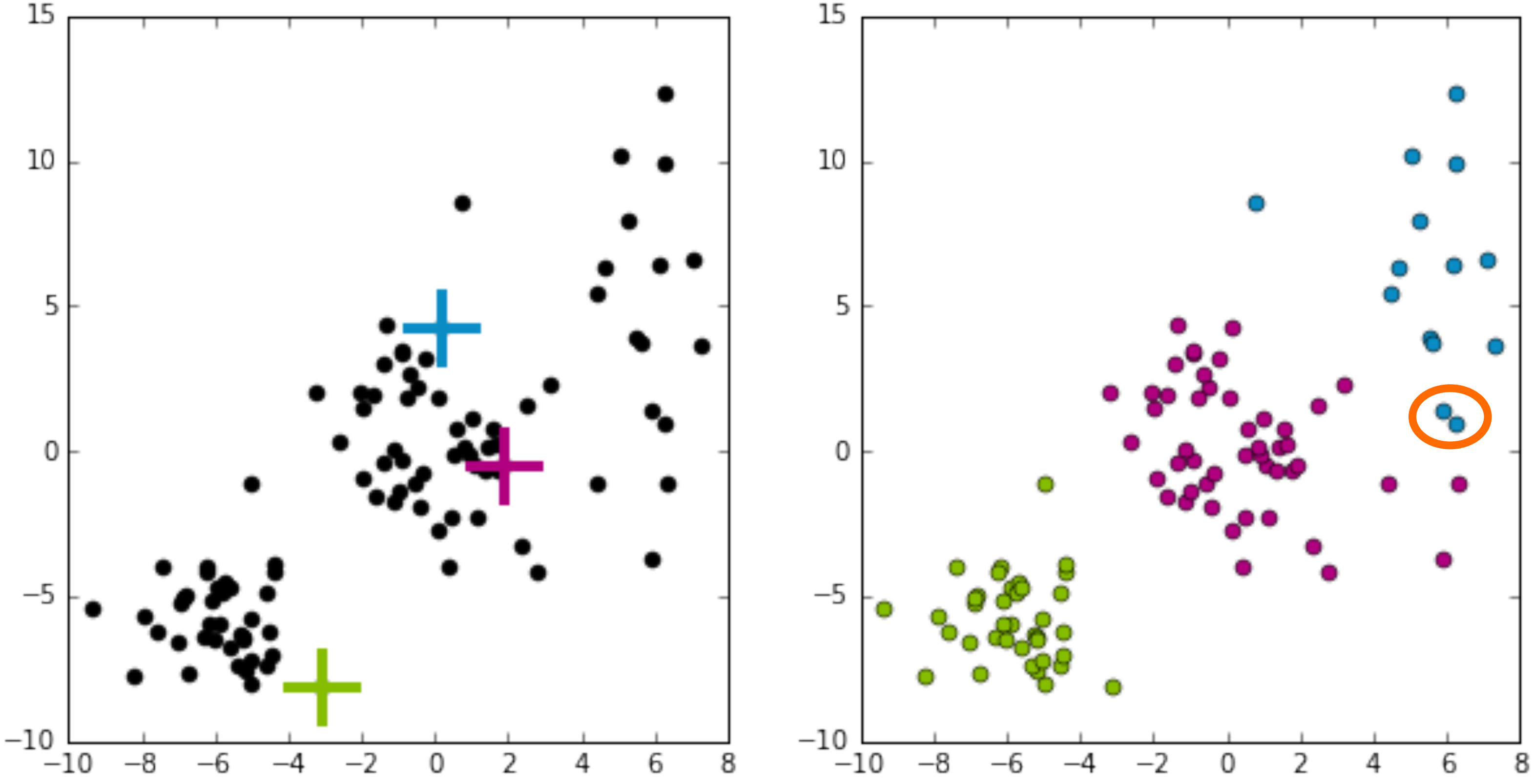


Convergence of k-means to local mode





Convergence of k-means to local mode



Smart initialization with k-means++

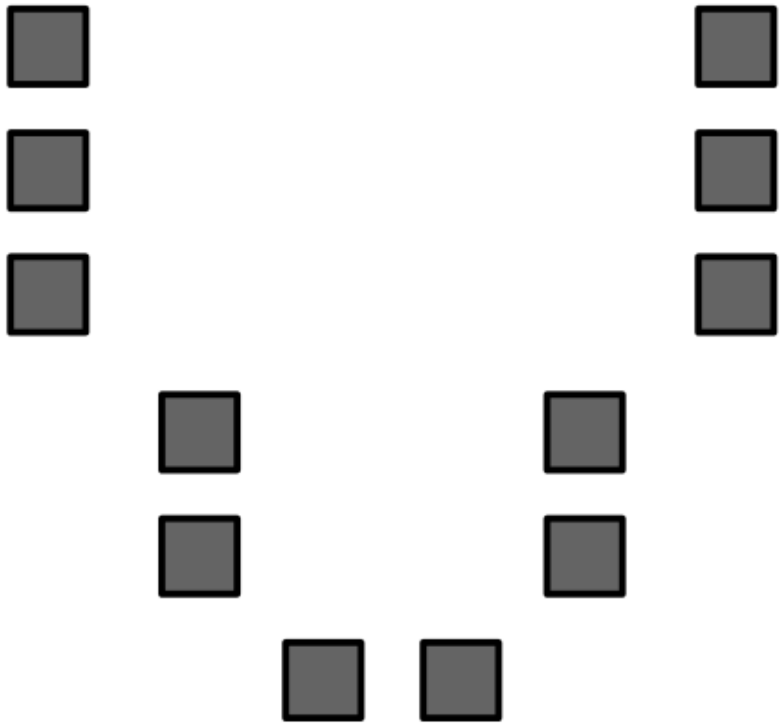
k-means++ overview

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

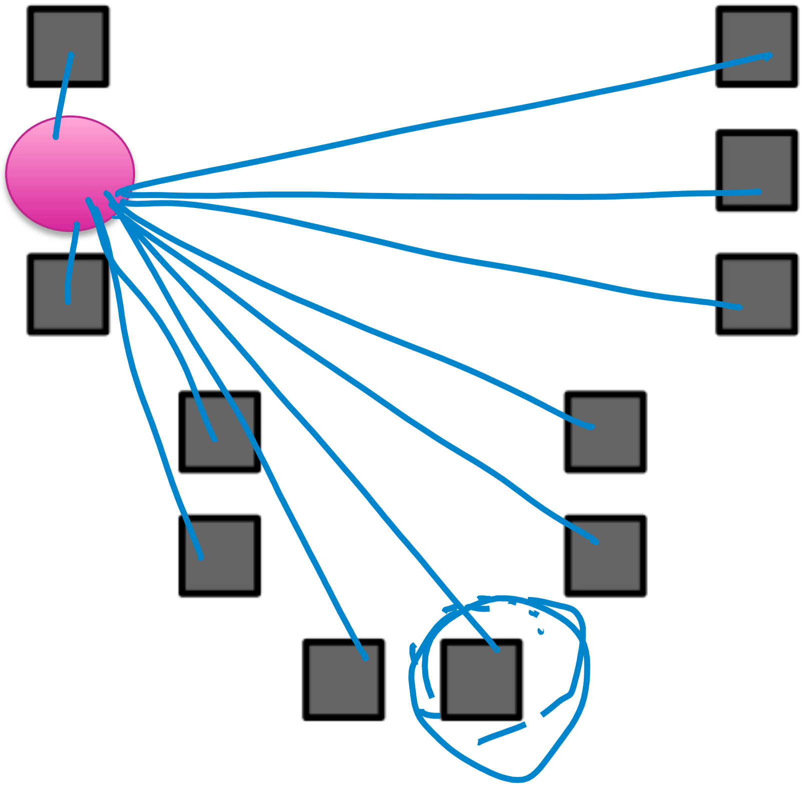
Smart initialization:

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

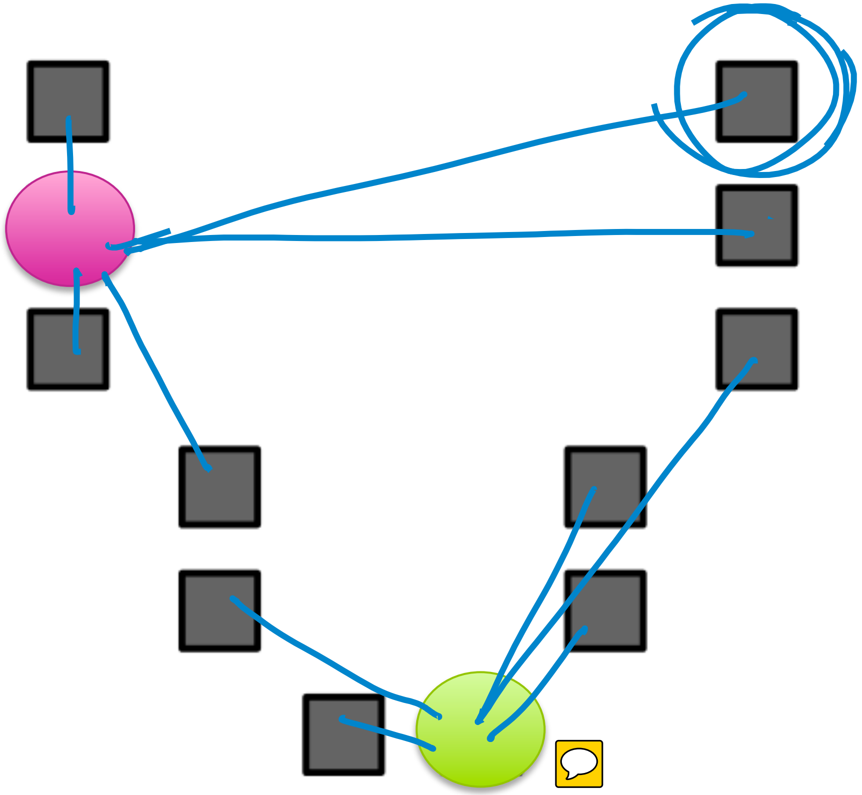


k-means++ visualized

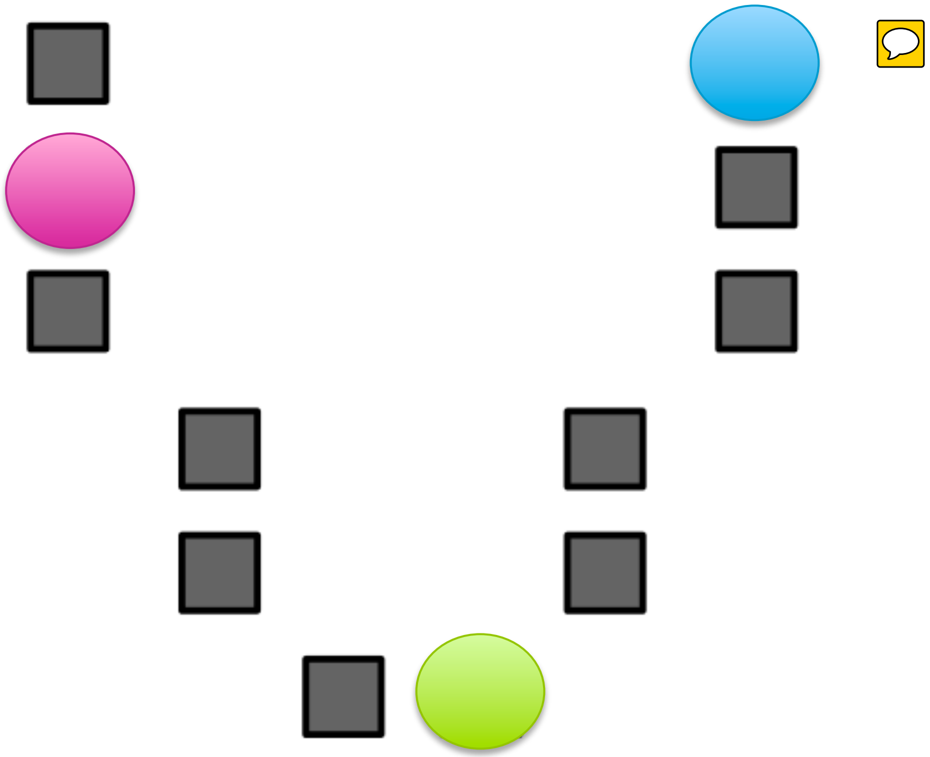


more likely to select a datapoint as a cluster center if that datapoint is far away (dist^2 increases this effect)

k-means++ visualized



k-means++ visualized



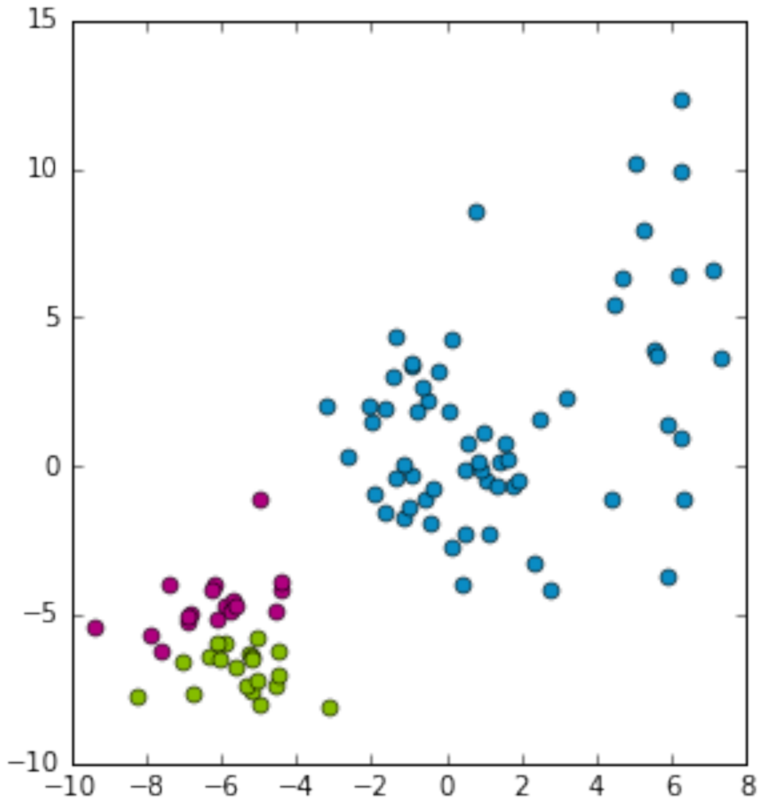
k-means++ pros/cons

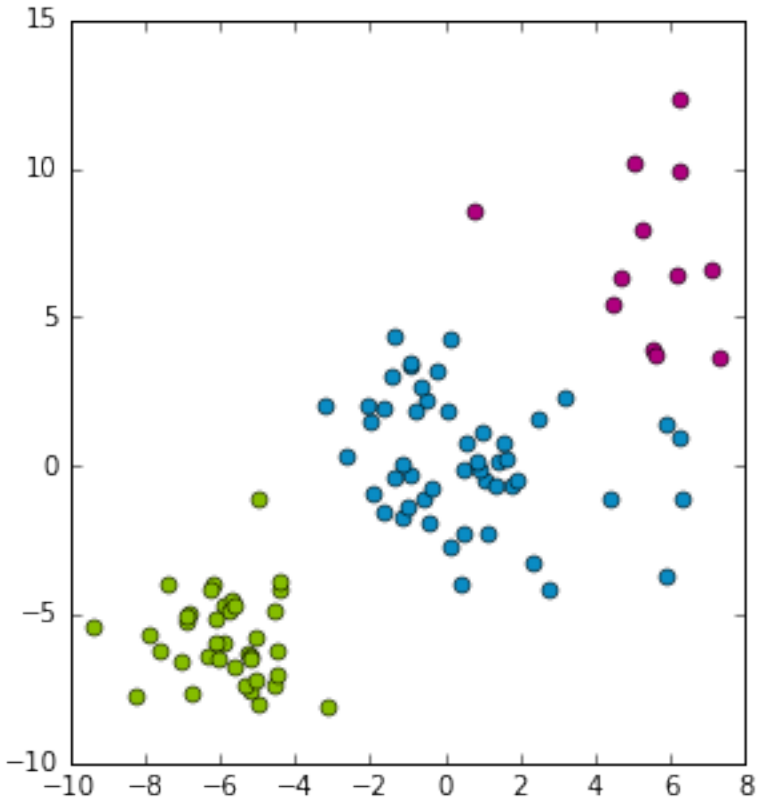
Computationally costly relative to random initialization, but the subsequent k-means often converges more rapidly

Tends to improve quality of local optimum and lower runtime

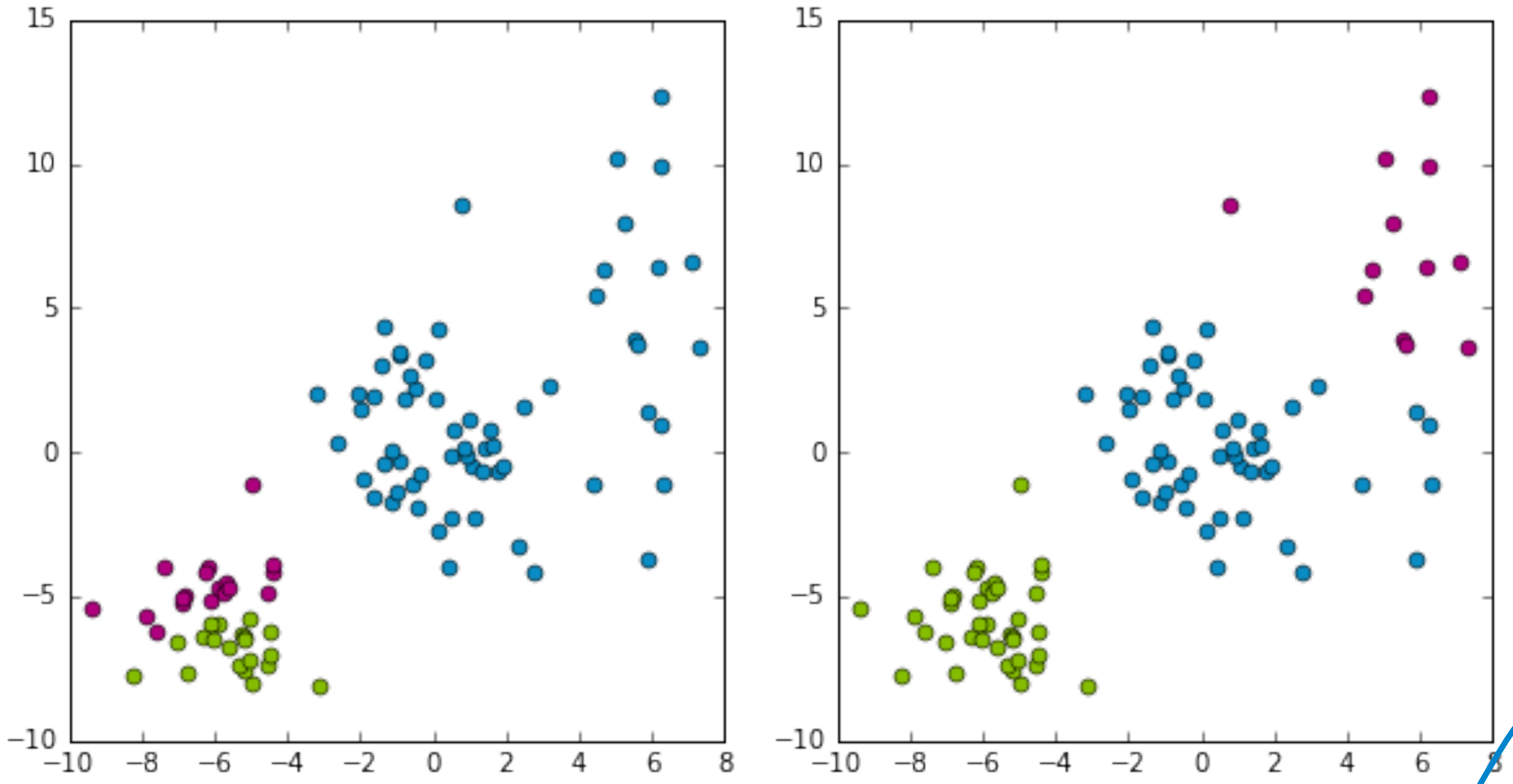
Assessing quality of the clustering and choosing the # of clusters

Which clustering do I prefer?





k-means objective



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

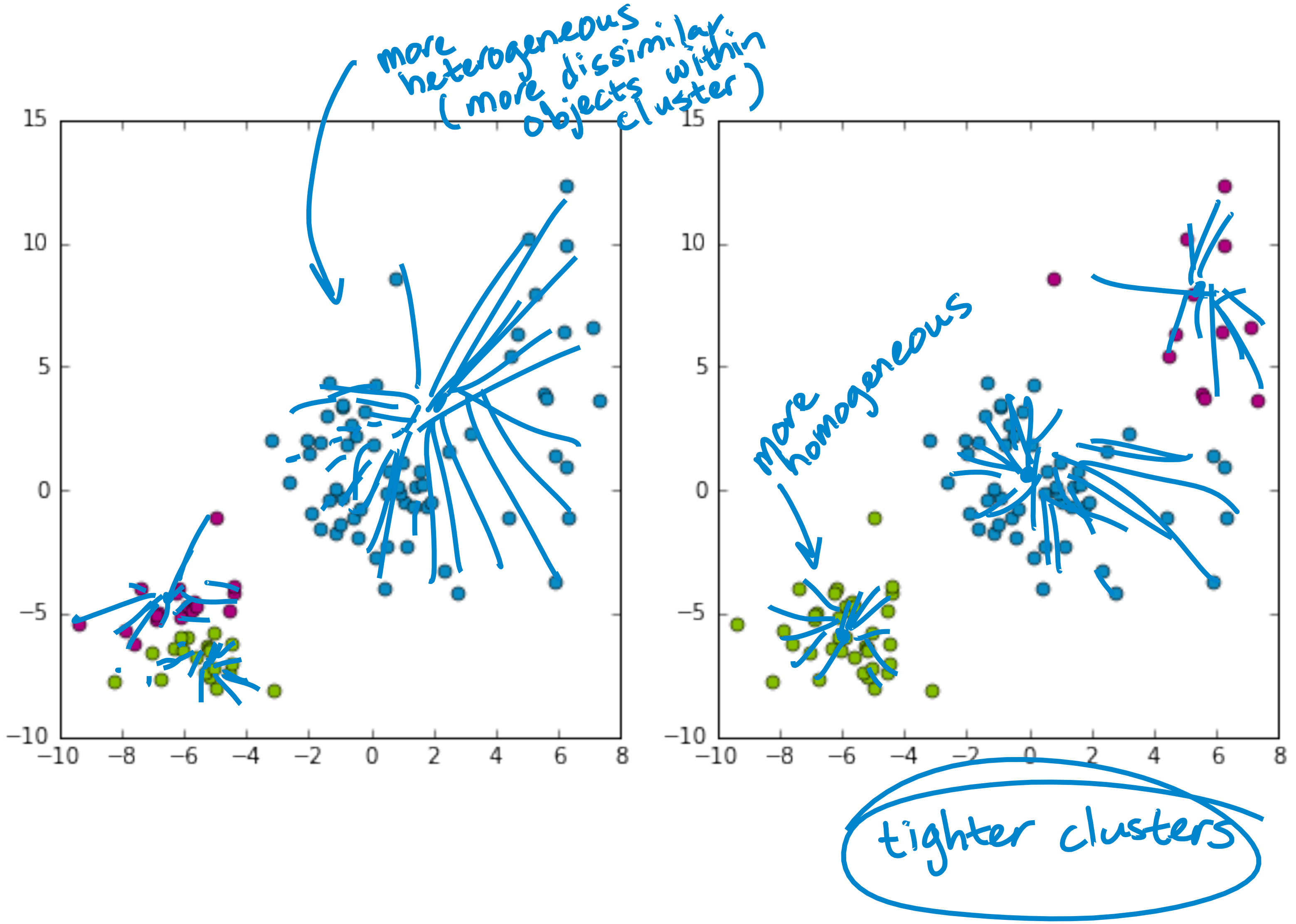
$$\sum_{j=1}^{k} \sum_{i:z_i=j} ||\mu_j - \mathbf{x}_i||_2^2$$

sum over all clusters

sum of squared distances in cluster j

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!

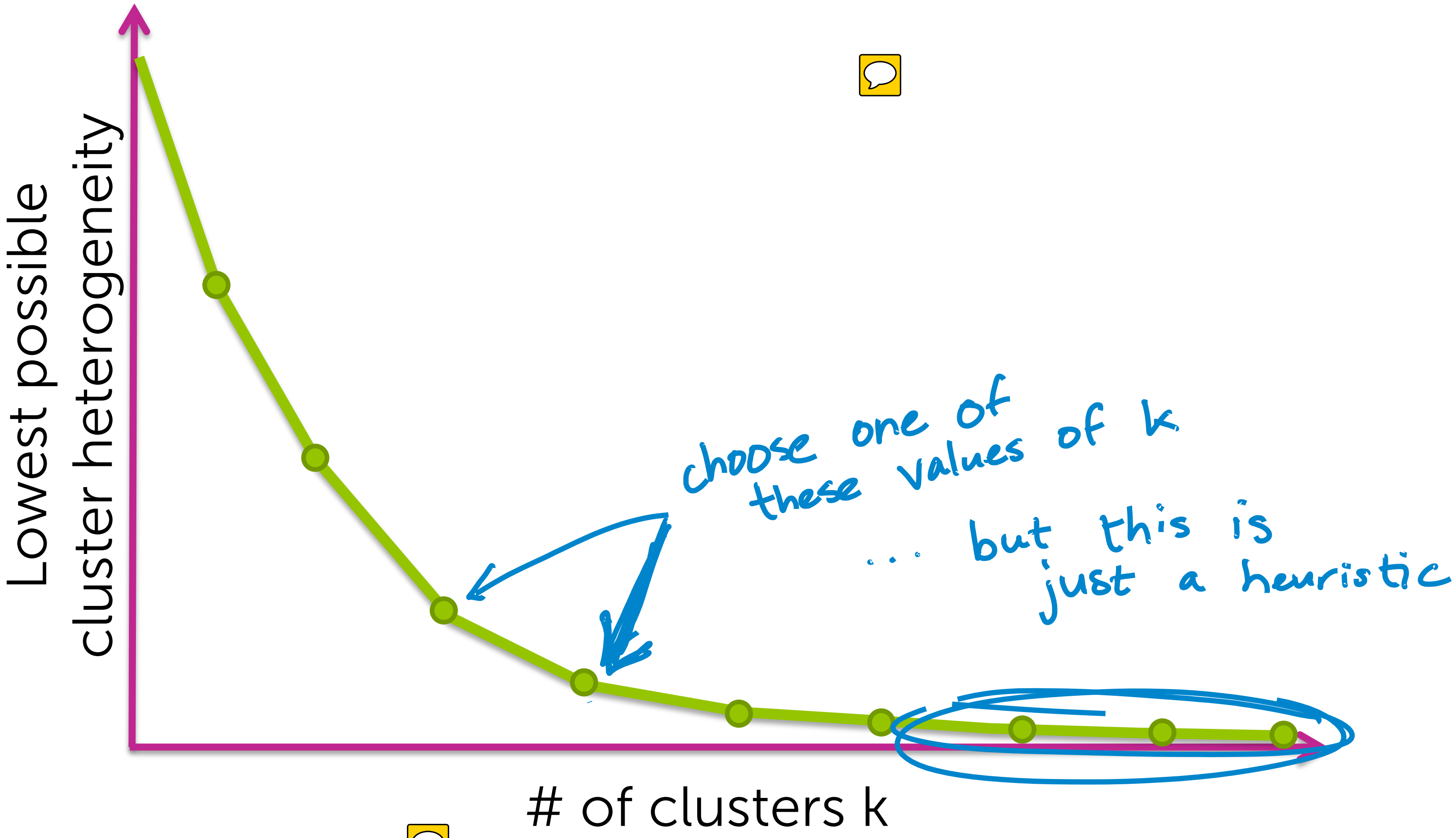
\# of observations

Extreme case of k=N:

– 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?



MapReduce

Counting words on a single processor

(The “Hello World!” of MapReduce)

Suppose you have 10B documents and 1 machine and want to count the # of occurrences of each word in the corpus

Code:

```
count[ ] ← init hash table
  for d in documents
    for word in d
      count[word] += 1
```

Naïve parallel word counting

- Word counts are independent across documents (data parallel)
- Count occurrences in sets of documents separately, then merge

evenly distribute data

Machine 1	Machine 2	…	Machine 1000
10M docs	10M docs	…	10M docs
$Count_1[\,]$	$Count_2[\,]$		$Count_{1000}[\,]$

add

combine local hash tables:

$$\text{Count[word]} = \sum_{i=1}^{1000} \text{Count}_i\text{[word]}$$

How do we do this for all words in vocab?

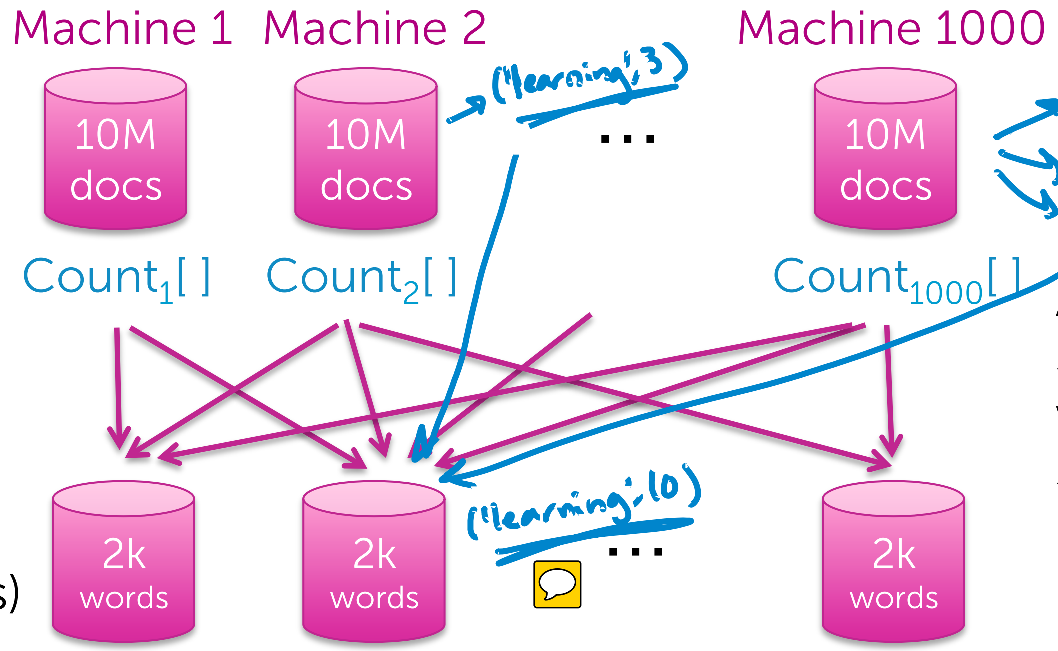
Back to sequential problem to merge counts…

have to cycle through all words in vocab… ugh.

Counting words in parallel & merging tables in parallel

1. Generate pairs (word,count) in parallel
2. Merge counts for each word in parallel

Phase 1: parallel (over docs) counting



All counts for a subset of words go to same machine

Phase 2: parallel (over words) sum

Which words go to machine i ?

$h: \underset{\text{vocab}}{V} \rightarrow [1,2,\ldots, \#\text{machines}]$

Send counts of 'learning' to machine h['learning']

How to map words to machines? Use a hash function!

h(word index) → machine index

MapReduce abstraction

Map:
- Data-parallel over elements, e.g., documents
- Generate (key,value) pairs
  - "value" can be any data type

('UW', 1)
('machine', 1)
('UW', 1)
('learning', 1) ...

Word count example:

```
map(doc)
    for word in doc
    emit(word,1)
```

key value

key list of values

```
reduce(word, counts_list)
    c = 0
    for i in counts_list
    c += counts_list[i]
    emit(word, c)
```

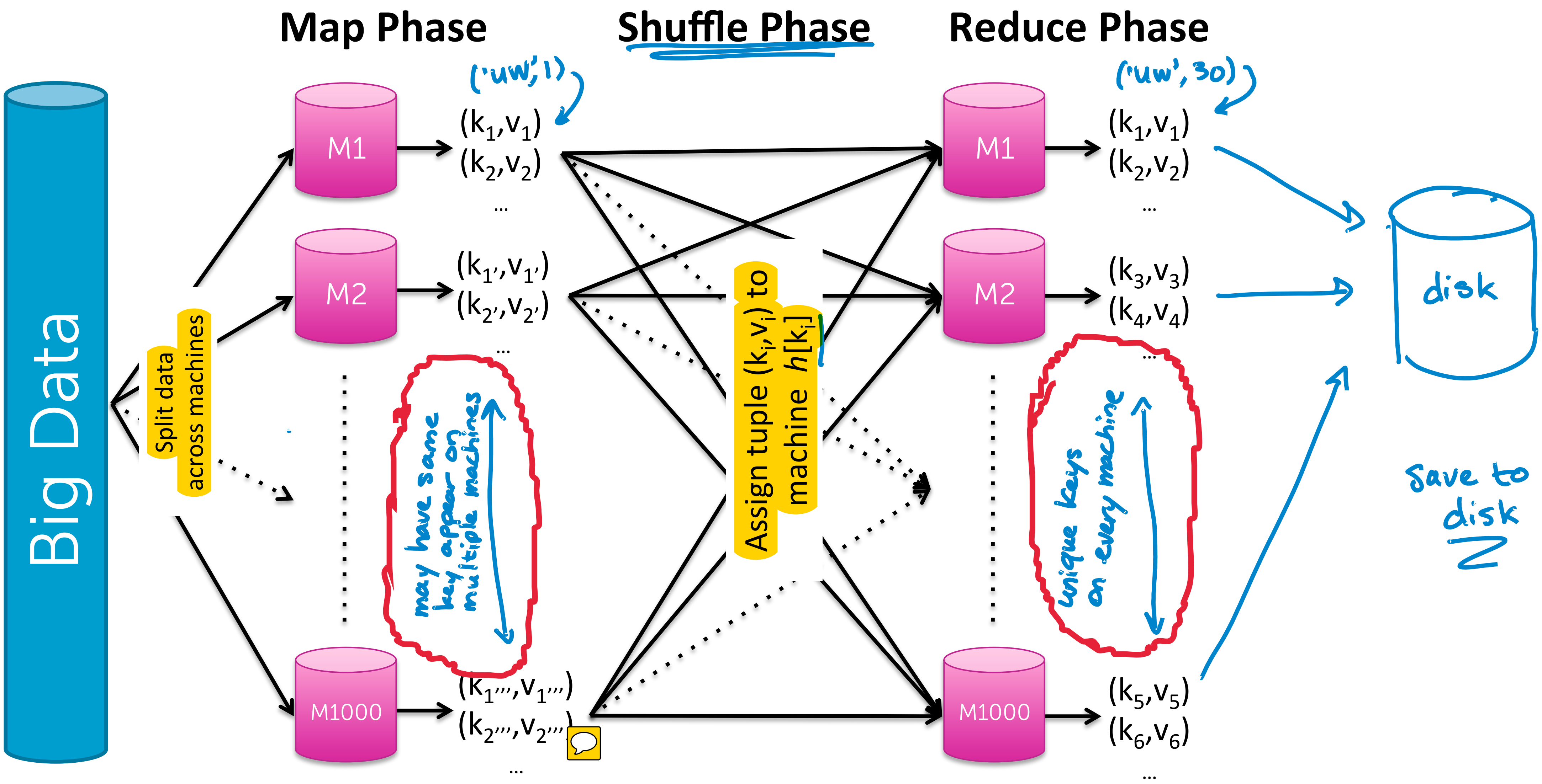
key value

Reduce:
- Aggregate values for each key
- Must be commutative-associative operation
  - a+b = b+a
  - (a+b)+c = a+(b+c)
- Data-parallel over keys
- Generate (key,value) pairs

reduce('UW', [1,17,0,0,12,2])
emit ('UW', 32)

MapReduce has long history in functional programming
- Popularized by Google, and subsequently by open-source Hadoop implementation from Yahoo!

MapReduce – Execution overview



Improving performance: Combiners

- Naïve implementation of MapReduce is very wasteful in communication during shuffle:

M1   ('uw',1) ('uw',1) ⋮ ('uw',1) $\Big\}$ 4002 instances of ('uw',1) on M1 $\xrightarrow{h(\text{'uw'})=7}$ M7

- Combiner: Simple solution…Perform reduce locally before communicating for global reduce
  - Works because reduce is commutative-associative

M1   ('uw',1) ⋮ ('uw',1) → reduce ('uw',4002) → M7

Scaling up k-means via MapReduce

MapReducing 1 iteration of k-means

Classify: Assign observations to closest cluster center

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

Map: For each data point, given ($\{\mu_j\}$,$\mathbf{x}_i$), emit(z_i,$\mathbf{x}_i$)

Recenter: Revise cluster centers as mean of assigned observations

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

Reduce: Average over all points in cluster j (z_i=k)

Classification step as Map

Classify: Assign observations to closest cluster center

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

set of cluster centers

a datapoint

map([$\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$,…, $\boldsymbol{\mu}_k$], $\mathbf{x}_i$)

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

emit(z_i,$\mathbf{x}_i$)

datapoint

cluster label

$z_i = 2$ (assigned to cluster 2)

datapoint x_i

e.g. emit (2, [17, 0, 1, 7, 0, 0, 5])

Recenter step as Reduce

Recenter: Revise cluster centers as mean of assigned observations

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

cluster label (key)

datapoints assigned to cluster j (have key j)

reduce(j, x_in_clusterj : [$\mathbf{x}_1$, $\mathbf{x}_3$,...,])
    sum = 0 ← total mass in cluster
    count = 0 ← total # of obs. in cluster
    for **x** in x_in_clusterj
    sum += **x**
    count += 1
    emit(j, sum/count)

cluster label

$\frac{\text{total mass}}{\text{total \# obs}}$

Some practical considerations

k-means needs an iterative version of MapReduce

- Not standard formulation

Mapper needs to get data point and all centers

- A lot of data!
- Better implementation:
  mapper gets many data points

Summary of parallel k-means using MapReduce

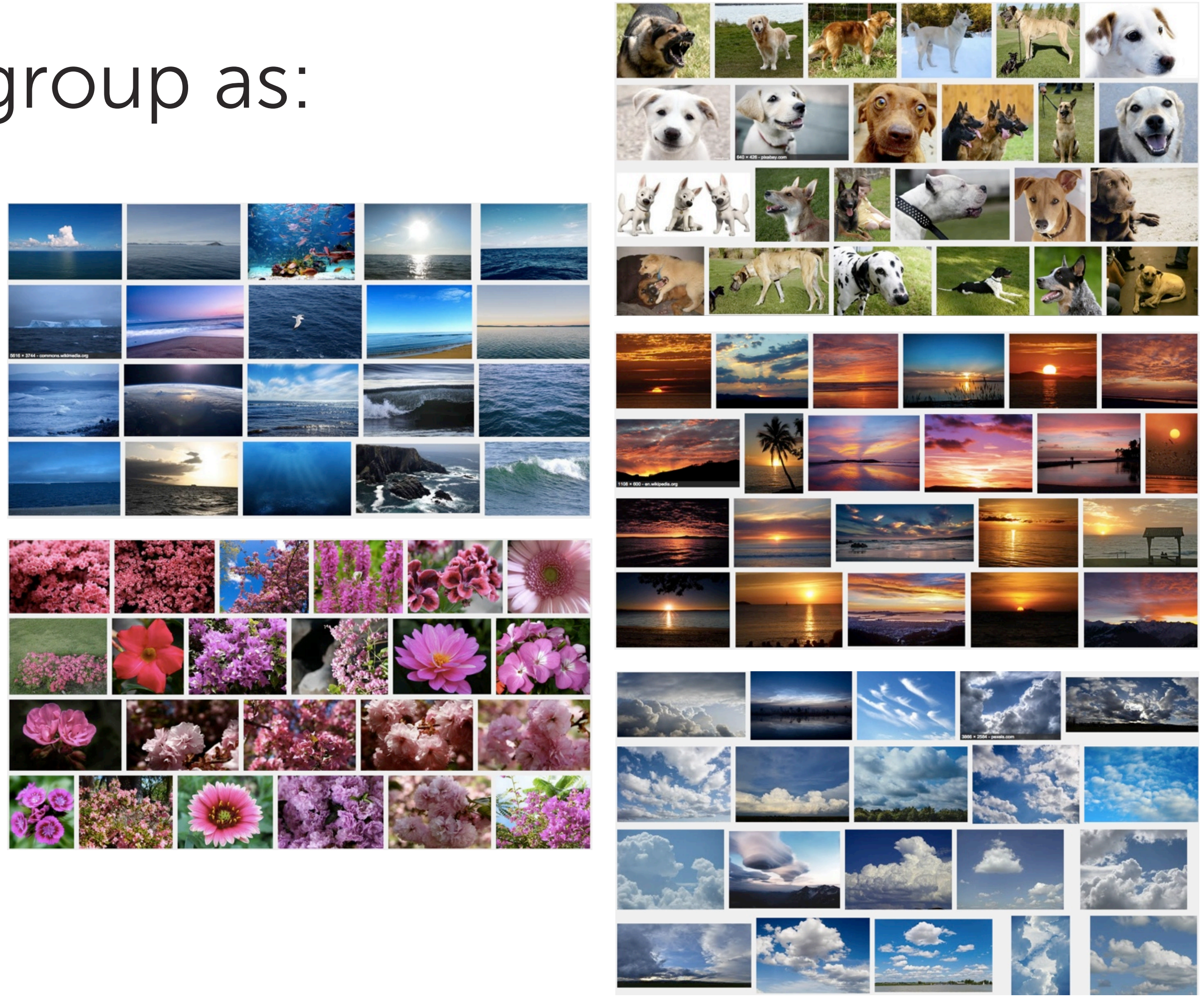
Map: classification step;
data parallel over data points

Reduce: recompute means;
data parallel over centers

Other examples

Clustering images

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



Structuring web search results

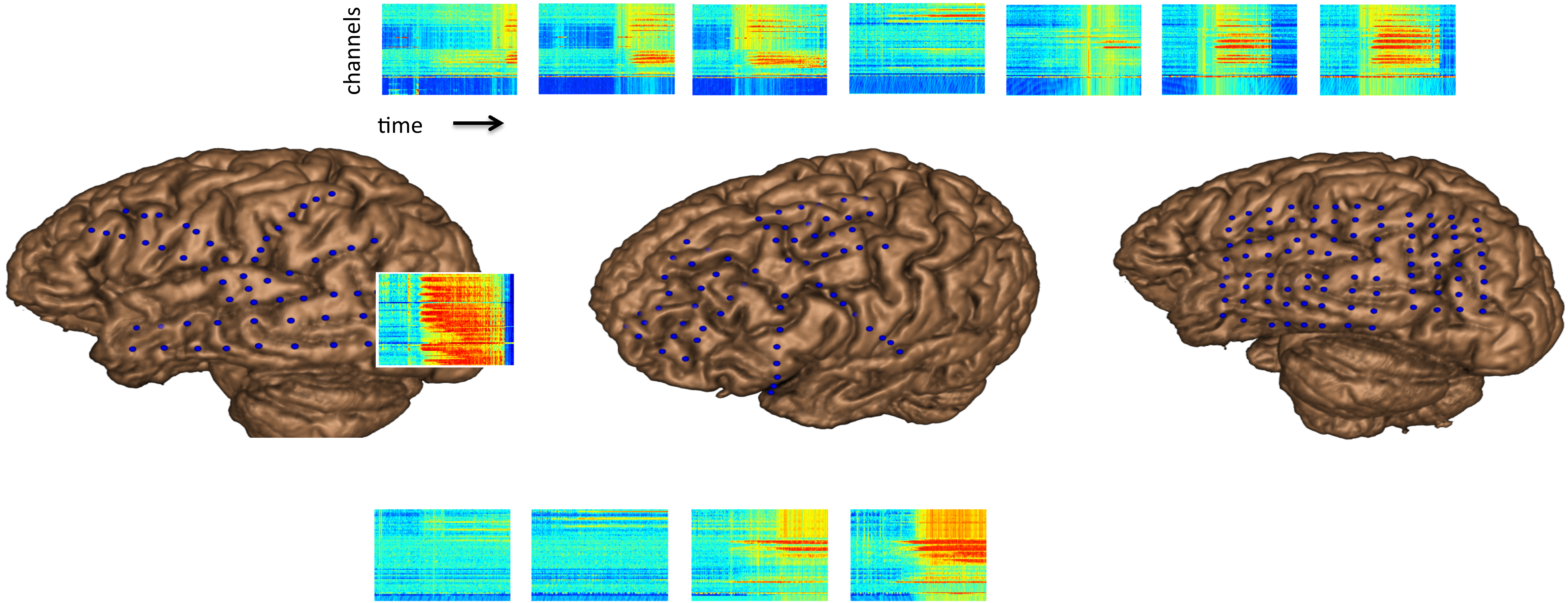
- Search terms can have multiple meanings
- Example: "cardinal"

- Use clustering to structure output

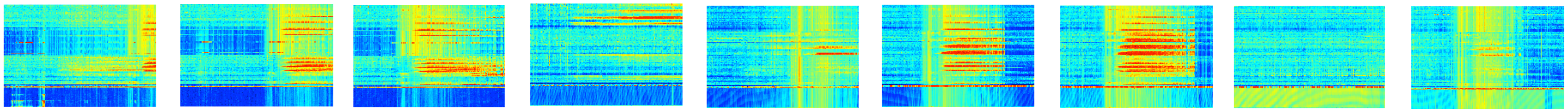
Grouping patients by medical condition

- Better characterize subpopulations and diseases

Example: Patients and seizures are diverse



Cluster seizures by observed time courses



Products on Amazon

- Discover product categories from purchase histories

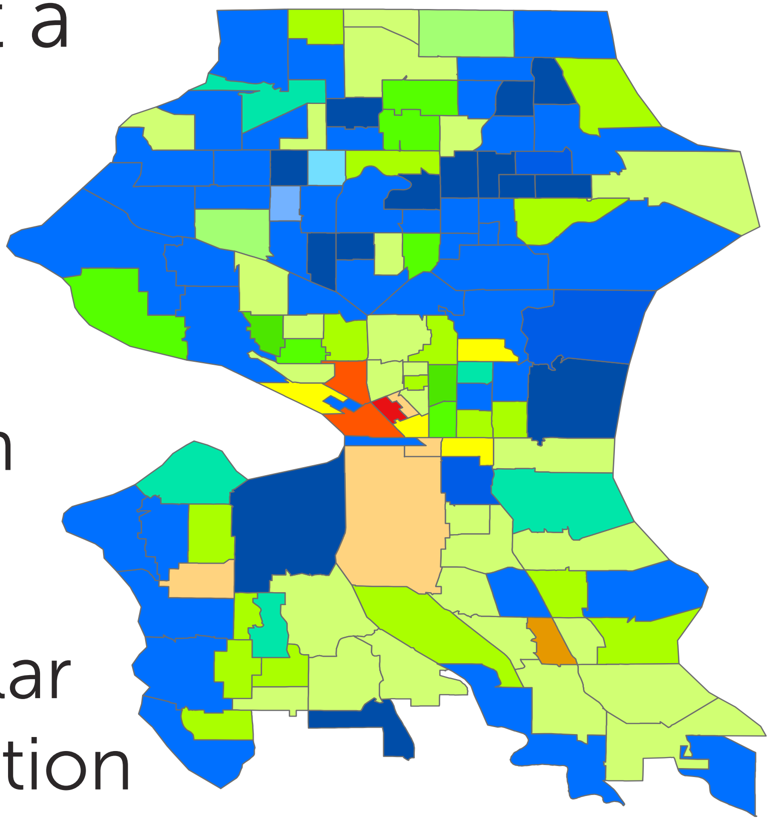
~~"furniture"~~

"baby"

- Or discovering groups of **users**

Discovering similar neighborhoods

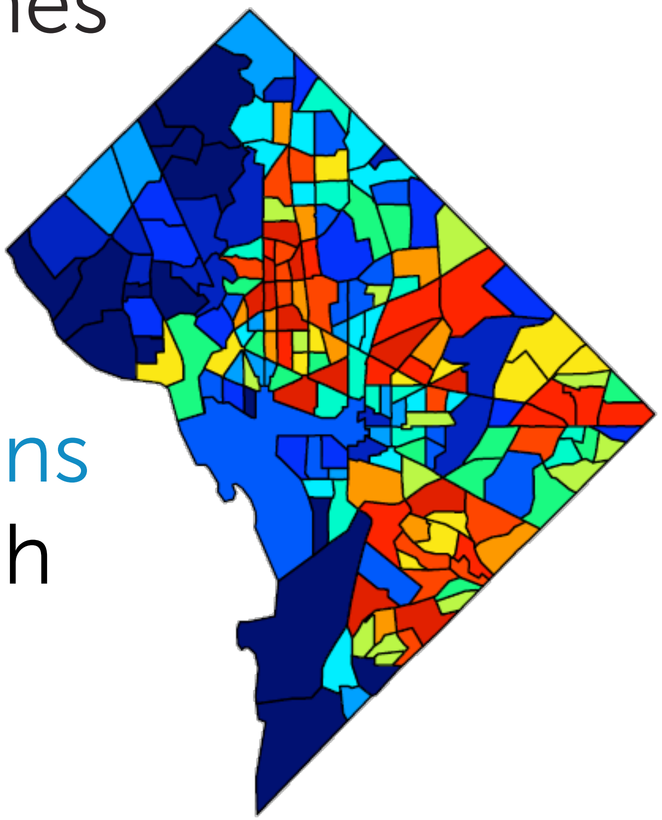
- **Task 1:** Estimate price at a small regional level
- Challenge:
  - Only a few (or no!) sales in each region per month
- Solution:
  - Cluster regions with similar trends and share information within a cluster



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 and MapReduce

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
- Define data parallel problems
- Explain Map and Reduce steps of MapReduce framework
- Use existing MapReduce implementations to parallelize k-means, understanding what's being done under the hood