Feature Preprocessing and Unsupervised Learning

Responsible:
Dr. Tegawendé F. BISSYANDE

tegawende.bissyande@uni.lu

Course Author:

Christopher Henard

Teacher Assistant:
Médéric Hurier
mederic.hurier@uni.lu

Course Features

- Sep. 20th - **Introduction to Big Data**

Part 1. Databases and Query Models for Big Data

- Sep. 27th - **Relational Databases: Reminders**
- Oct. 4th - **Relational Databases: Internals**
- Oct. 11th - **NoSQL & NewSQL Databases**
- Oct. 18th - **MapReduce Model**
- Oct. 25th - **Hadoop and Spark**
- Nov. 8th - **Datalog Model**

Part 2. Data Analysis and Machine Learning

- Nov. 15th - **Statistics and Data Analysis**
- Nov. 22th - **Communication and Visualization**
- Nov. 29th - **Feature Engineering and Supervised Learning**
- Dec. 5st - **Feature Preprocessing and Unsupervised Learning**
- Dec. 12th - **Homework Time**

Section Features

- Feature Preprocessing
- Unsupervised Learning
- Reinforcement Learning

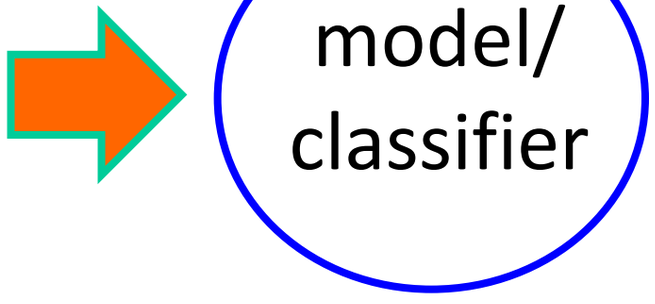
Feature Preprocessing

Our current learning model

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn



Expectation after preprocessing

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Feature preprocessing

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn

model/
classifier

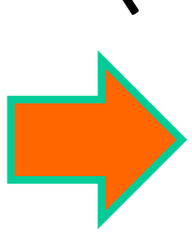
"better" training data

Expectation after preprocessing

training data
(labeled examples)

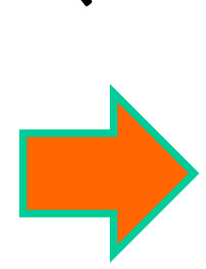
Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Feature preprocessing



Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn

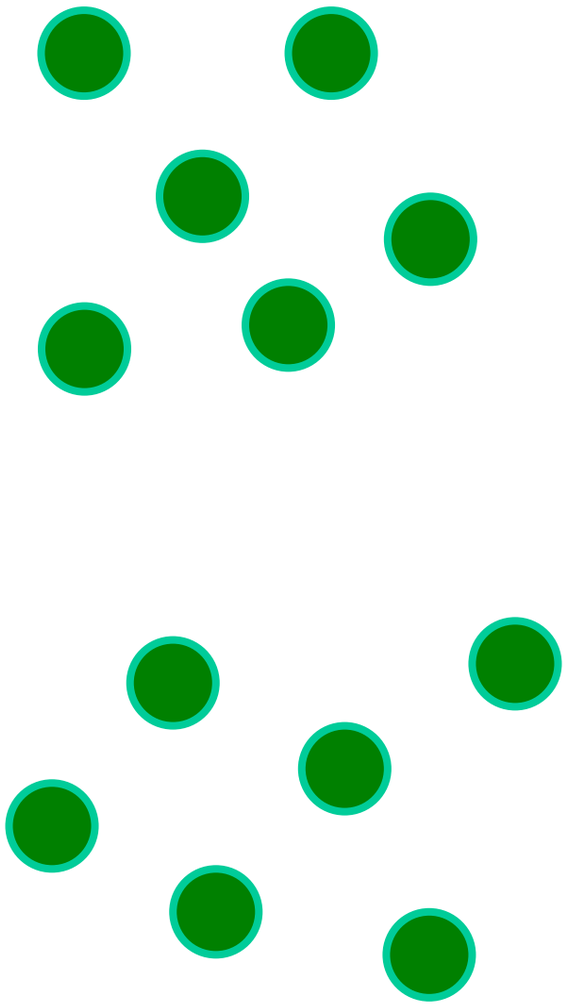


model/
classifier

“better” training data

What types of preprocessing might we want to do?

Outlier detection



What is an outlier?

An example that is inconsistent with the other examples

What types of inconsistencies?

Example:

- extremes values
- conflicts: same values, different labels

Removing extreme outliers

- Throw out examples that have extreme values in one dimension
- Throw out examples that are very far away from any other example
- Train a probabilistic model on the data and throw out "very unlikely" examples

This is an entire field of study by itself !

Often called outlier or anomaly detection

Removing conflicting examples

Identify examples that have the same features, but differing values

- can cause issues (for example, not converging)
- in general, unsatisfying from a learning perspective

Can be expensive computationally (examine all pairs), though faster approaches are available

Reminder: statistic measures

mean: a measure of centrality (μ)

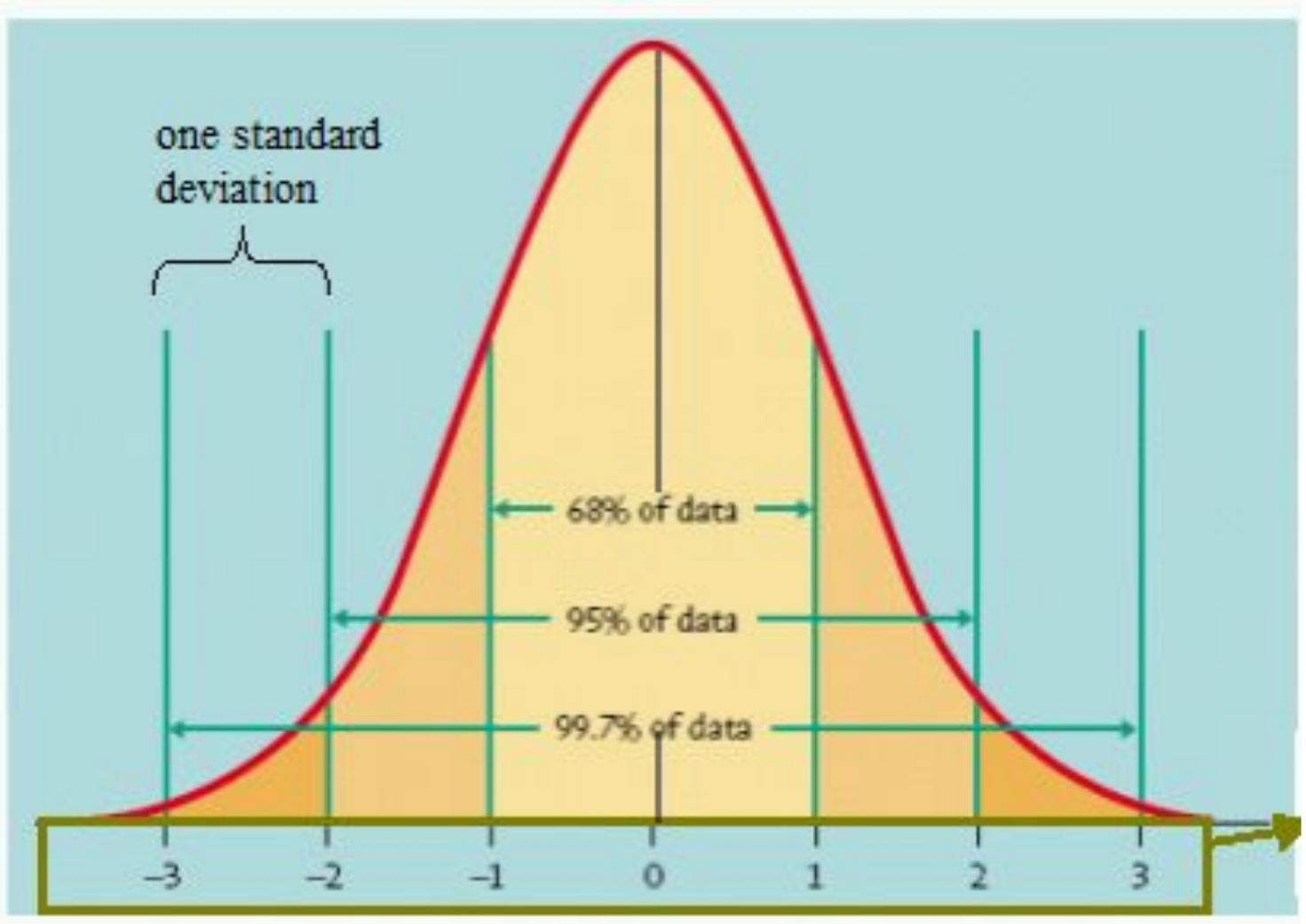
variance: a measure of variation, calculated as:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

standard deviation: square root of the variance (σ)

How can these help us with outliers?

Outlier detection

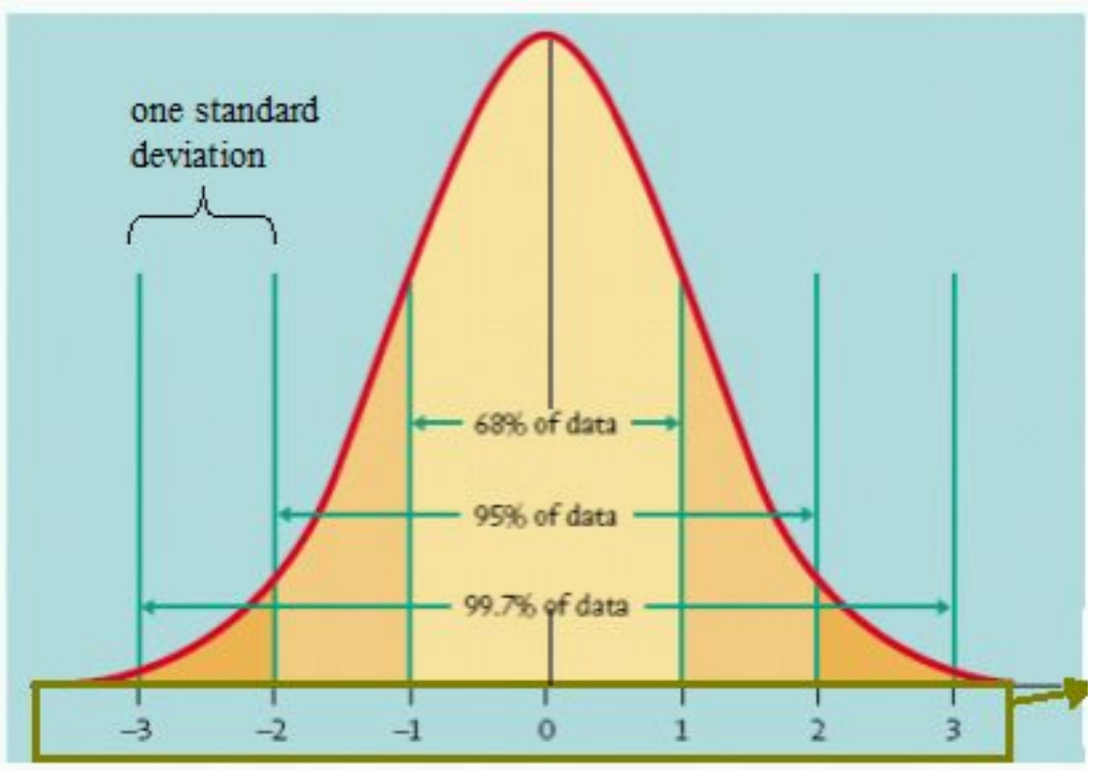


If we know the data is distributed normally
(i.e. via a normal/gaussian distribution)

Outliers in a single dimension

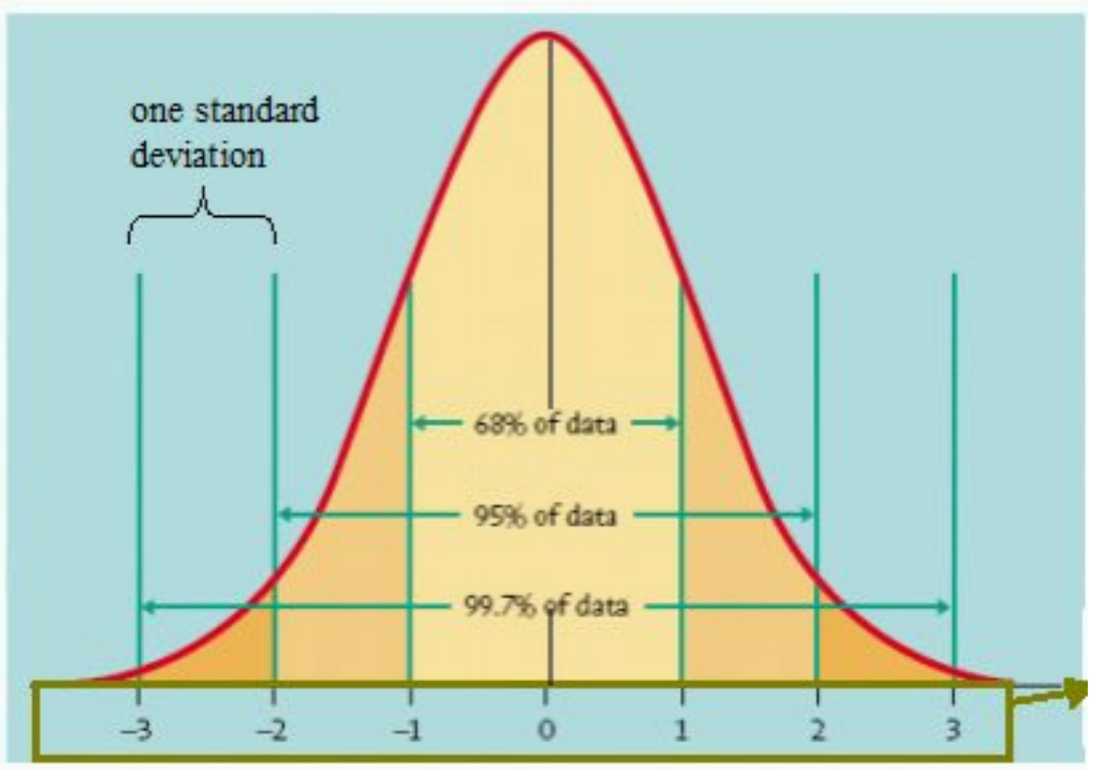
Examples in a single dimension that have values greater than $|k\sigma|$ can be discarded (for $k >> 3$)

Reasonable even if the data isn't actually distributed normally



Outliers in general

- Calculate the centroid/center of the data
- Calculate the average distance from center for all data
- Calculate standard deviation and discard points too far away



Outliers for Machine Learning

Some good practices:

- Throw out conflicting examples
- Throw out any examples with extreme values
- Check for erroneous feature values
  - negative values for a feature with only positive
- Let the learning algorithm/other pre-processing handle the rest

Feature pruning/selection

Good features provide us information that helps distinguish between labels. **However, not all features are good**

Feature pruning is the process of removing “bad” features

Feature selection is the process of selecting “good” features

What makes a bad feature ?

Why would we have them in our data?

Bad features

Pick 5 random binary numbers

f_1 f_2 … label

I've already labeled these examples and I have two features

Bad features

label

1

0

1

1

0

If we have a “random” feature, i.e. a feature with random binary values,

what is the probability that our feature perfectly predicts the label?

Noisy features

Adding features ***can*** give us more information, but not always

Determining if a feature is useful can be challenging

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Noisy features

Ideas for removing noisy/random features?

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Removing noisy features

The expensive way:

- Split training data into train/dev
- Train a model on all features
- For each feature f:
  - Train a model on all features – f
  - Compare performance of all vs. all-f on dev set
- Remove all features where decrease in performance between all and all-f is less than some constant

Feature ablation study

Removing noisy features

Binary features:

remove “rare” features, i.e. features that only occur (or don’t occur) a very small number of times

Real-valued features:

remove features that have low variance

In both cases, can either use thresholds, throw away lowest x%, use development data, etc.

Why ?

Some rules of thumb

Be very careful in domains where:

- the number of features > number of examples
- the number of features ≈ number of examples
- the features are generated automatically
- there is a chance of “random” features

Features should be removed based

on some domain knowledge (i.e. problem-specific)

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features

Feature selection

Let’s look at the problem from the other direction,
selecting good features

What are good features?

How can we pick/select them?

Good features

A good feature correlates well with the label

label				
1	1	0	1	
0	0	1	1	
1	1	0	1	…
1	1	0	1	
0	0	1	0	

How can we identify this?

- training error
- correlation model
- statistical test
- probabilistic test
- …

Training error feature selection

- For each feature f:
  - calculate the training error based only on feature f
- Rank each feature by this value
- Pick top k, top x%, etc.
  - can use a development set to help pick k or x

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Would our two classifiers (k-NN, DT) learn the same models on these two data sets?

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Decision trees don’t care about scale,
so they’d learn the same tree

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

k-NN: NO! The distances are biased based on feature magnitude

$$D(a,b)=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\ldots+(a_n-b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

Which of the two examples are closest to the first?

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D(a,b)=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\ldots+(a_n-b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

$$D = \sqrt{(7-4)^2 + (5-4)^2} = \sqrt{10}$$

$$D = \sqrt{(5-4)^2 + (8-4)^2} = \sqrt{17}$$

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D = \sqrt{(70-40)^2 + (5-4)^2} = \sqrt{901}$$

$$D = \sqrt{(50-40)^2 + (8-4)^2} = \sqrt{116}$$

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

How do we fix this?

Feature normalization

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Modify all values for a given feature

Normalize each feature

For each feature (over all the data):

Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

Rescale/adjust feature values to avoid magnitude bias:

- Variance scaling: divide each value by the standard dev.
- Absolute scaling: divide each value by the largest value

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data

Example normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
70	60	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Any problem with this?
Solutions?

Length normalization

Make all examples have length = 1

Divide each feature value by ||x||

- Prevents a single example from being too impactful
- Equivalent to projecting onto a unit sphere

$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

So far…

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data
5. Normalize example length
6. Finally, train your model!

What about testing?

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn

model/
classifier

"better" training data

Test data preprocessing

1. Throw out outliers
2. Remove irrelevant/noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data
5. Normalize example length

Whatever you do on training
you have to do the EXACT same on testing!

Feature pre-processing: Summary

Many preprocessing techniques

Which will work well will depend on the data and the classifier

Try them out and evaluate how they affect performance on data

Make sure to do **exact same** pre-processing on train and test

1. Throw out outlier examples
2. Remove noisy features
3. Pick “good” features
4. Normalize feature values
    1. center data
    2. scale data (either variance or absolute)
5. Normalize example length

Unsupervised Learning

Machine Learning

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Classification	Clustering	Decision Process
Regression	Segmentation	Reward System
Ranking	Dimension Reduction	Recommendation Systems
	Association Mining	

Reminder: Supervised learning

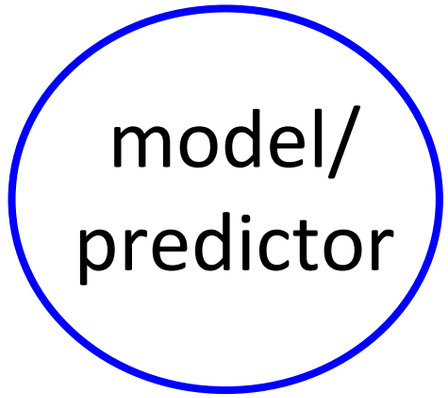
label

$label_1$

$label_3$

$label_4$

$label_5$



Supervised learning: given labeled data

Unsupervised learning





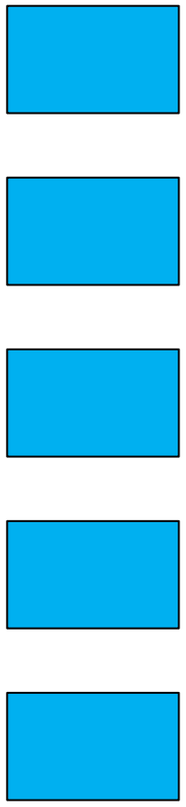
Unsupervised learning: given data without labels

In a nutshell

- The data has no labels!
- What can we still learn?
  - Salient groups in the data
  - Density in feature space
- Key approach: **clustering**
- … but also:
  - Association rules
  - Density estimation
  - Principal Components Analysis (PCA)

Unsupervised learning: clustering

Raw data



extract features

features

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

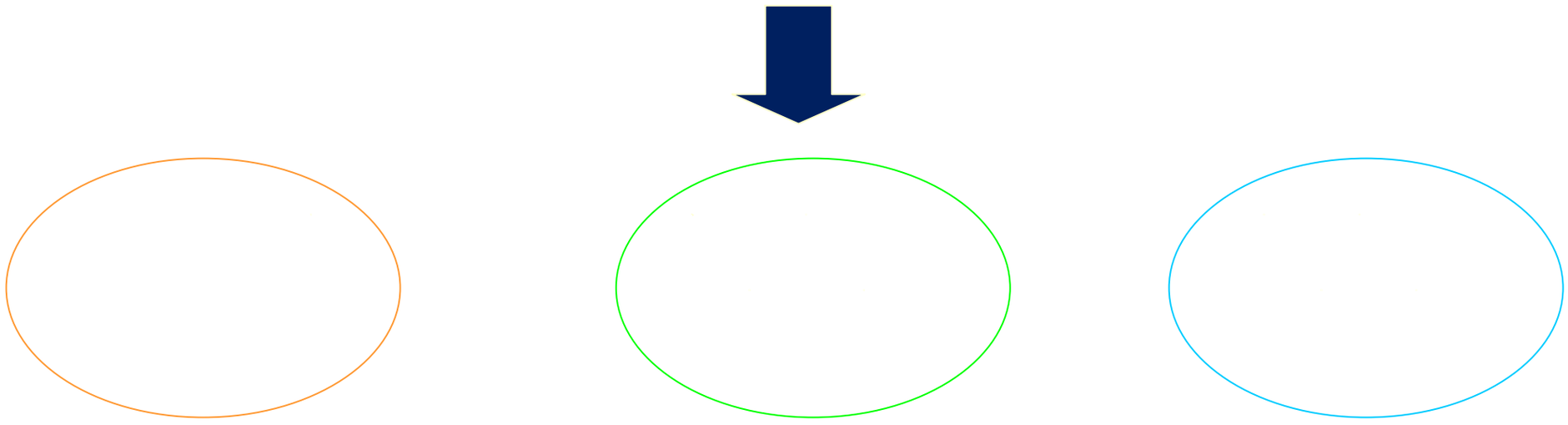
$f_1, f_2, f_3, \ldots, f_n$

group into classes/clusters

No “supervision”, we’re only given data and want to find groupings

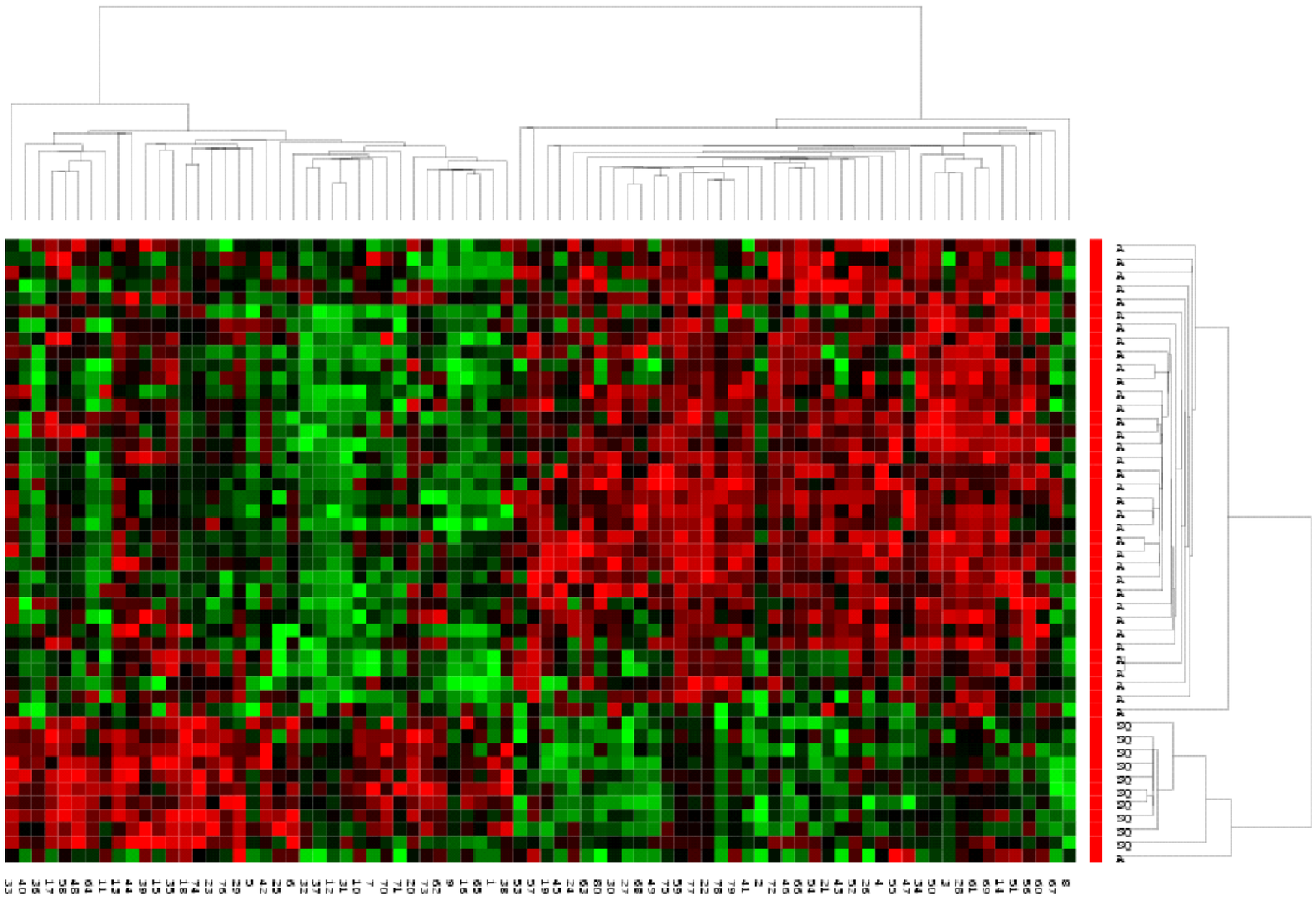
Unsupervised learning: clustering

Clustering is the process of grouping
a set of objects into classes of similar objects

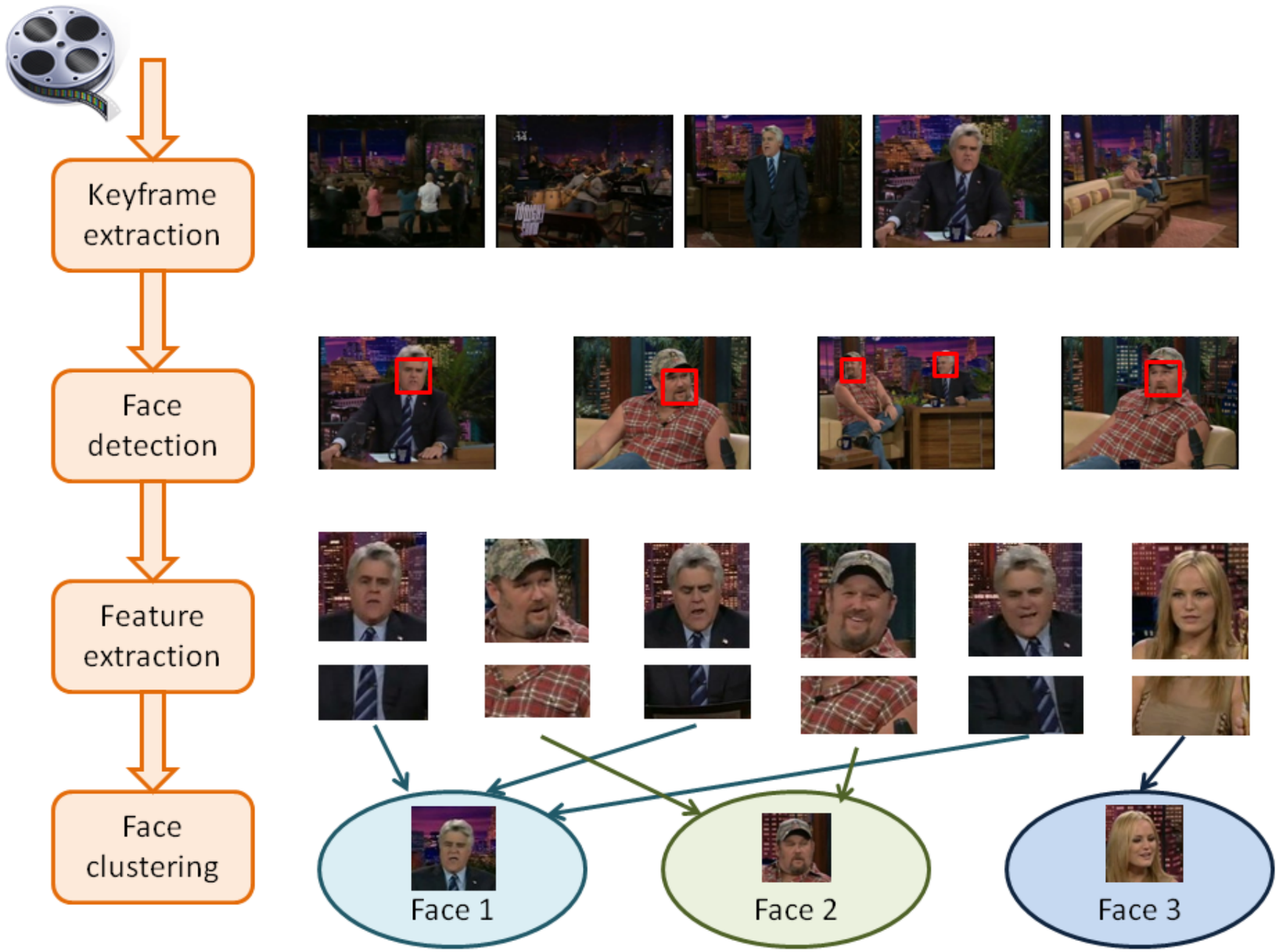


Applications of Clustering: image segmentation

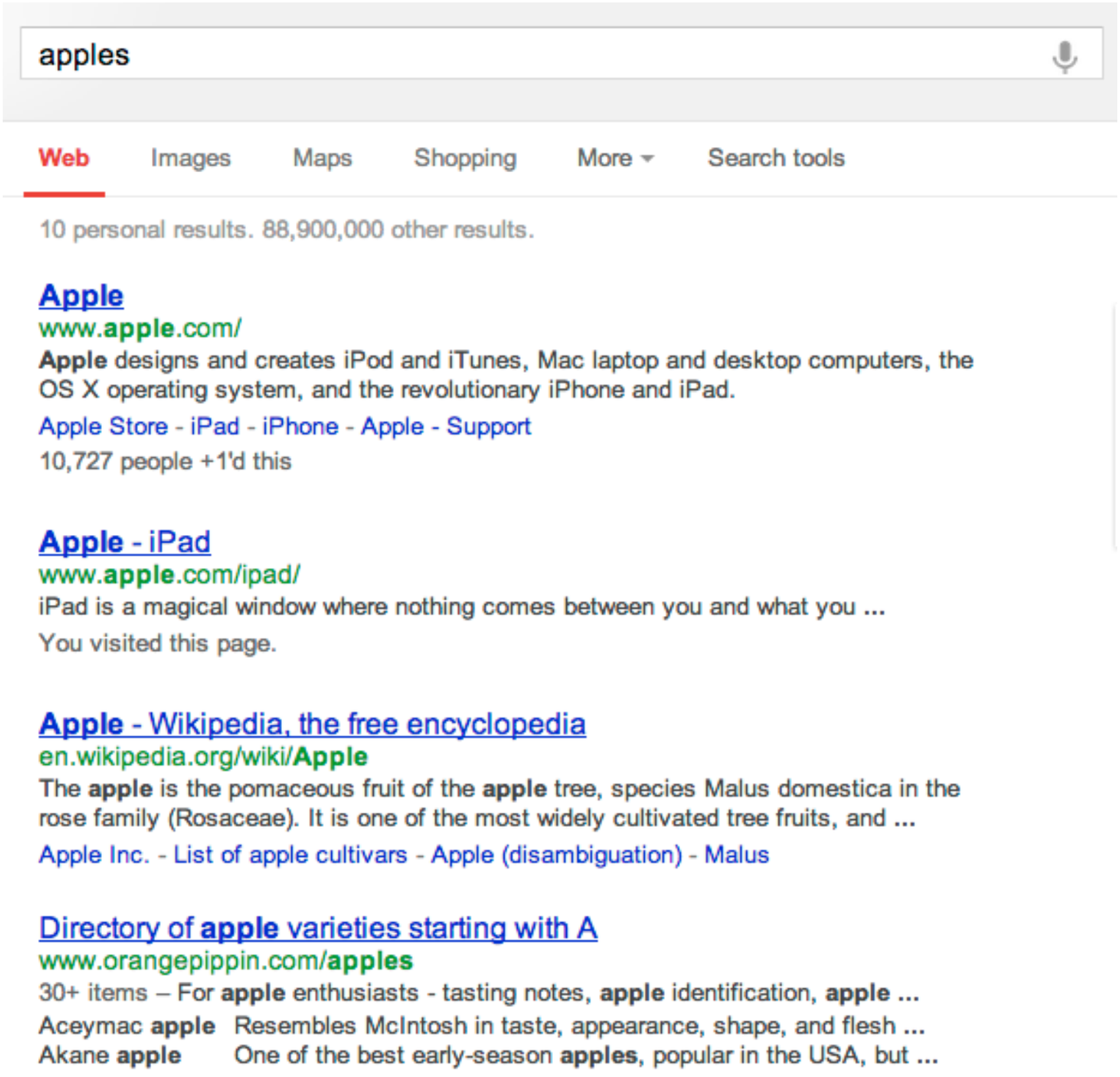
Applications of Clustering: genes expression data



Applications of Clustering: face grouping



Applications of Clustering: search results



Applications of Clustering: Google news

Google

News

Top Stories

- Iran
- Xbox One
- Tarun Tejpal
- Manny Pacquiáo
- Ukraine
- Kabul
- New England Patriots
- Latvia
- Derrick Rose
- Doctor Who

+ Xbox One

Console Wars 2013: Microsoft's Xbox One vs. Sony's PlayStation 4

E! Online - 1 hour ago

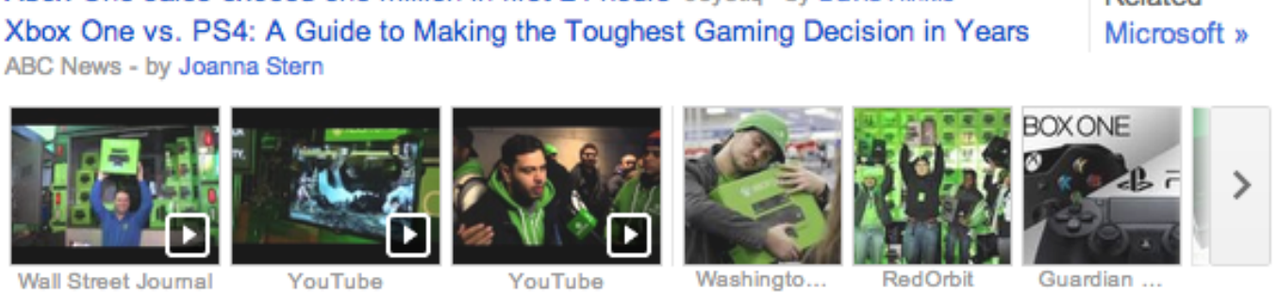
The future is now! Last week, Sony released its next generation console, PlayStation 4. This weekend, Microsoft drops the much touted all-in-one media device, Xbox One. We've been geeking out over the two new systems, and compiling a report on the new ...

Xbox One sales exceed one million in first 24 hours Joystiq - by David Hinkle

Xbox One vs. PS4: A Guide to Making the Toughest Gaming Decision in Years ABC News - by Joanna Stern

Related Microsoft »

See realtime coverage



Wall Street Journal | YouTube | YouTube | Washingto... | RedOrbit | Guardian ...

The Guardian

Xbox One and Microsoft websites marred by problems on launch day

The Guardian
9 hours ago

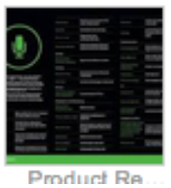
Written by Jemima Kiss

Microsoft's Xbox One launch was marred by problems with its online services early on Friday which took down the official website Xbox.

Consumers line up for Xbox One

USA TODAY - Nov 23, 2013

Eager video game players lined up at stores across the country awaiting the arrival of Microsoft's Xbox One, a week to the day after rival Sony introduced its PlayStation 4. The console, available for sale tonight at 12:01 a.m.



Product Re...

Here are all the Xbox One voice commands

Polygon
9 hours ago

Written by Megan Farokhmanesh

Microsoft posted a guide to Xbox One voice commands, including how to navigate menus, control volume and multitask, on its Tumblr.

Applications of Clustering: text analysis

- Noun phrase coreference

Input text

John Simon, Chief Financial Officer of Prime Corp. since 1986, saw his pay jump 20%, to $1.3 million, as the 37-year-old also became the financial-services company's president.

Cluster JS

John Simon

Chief Financial Officer

his

the 37-year-old

president

Cluster PC

Prime Corp.

the financial-services company

Singletons

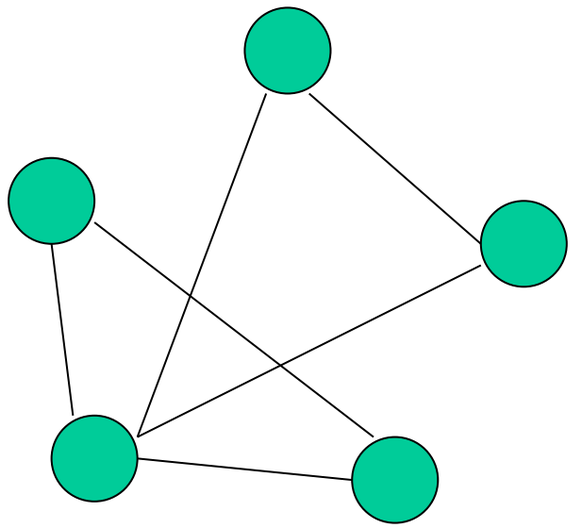
1986

pay

20%

$1.3 million

Applications of Clustering: users



Who-messages-who
IM/text/twitter graph

~100M nodes

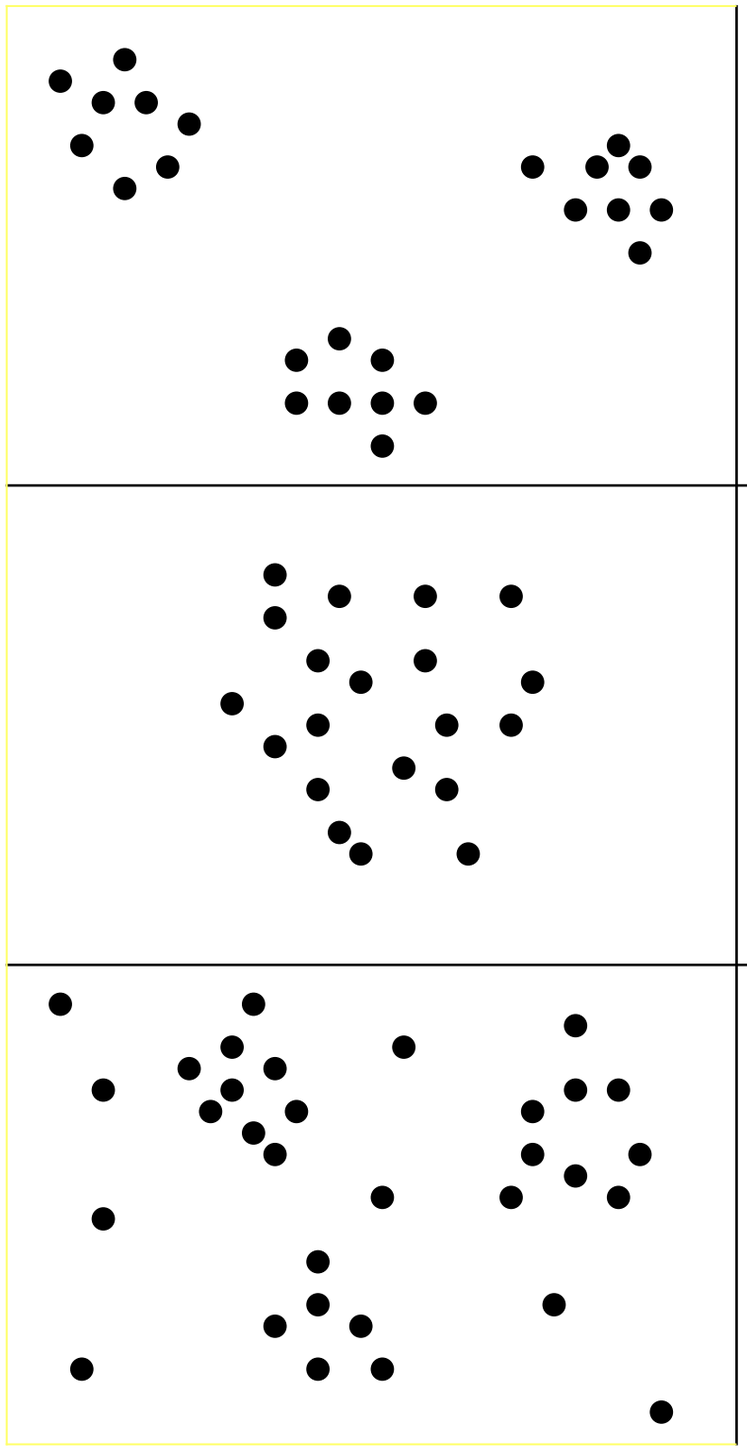
Find clusters of users

- Targeted advertising
- Exploratory analysis

Clusters of the Web Graph

- Distributed pagerank computation

Clustering



Sometimes easy

Sometimes impossible

and sometimes

in between

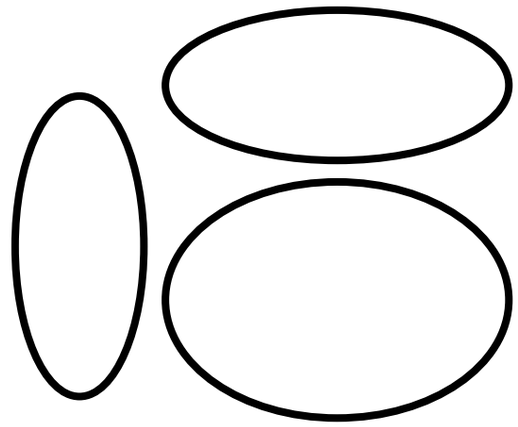
Which type ?

- Representation for clustering
  - How do we represent an example
  - Similarity/distance between examples
- Flat clustering or hierarchical
- Number of clusters
  - Fixed a priori
  - Data driven?

Clustering Algorithms

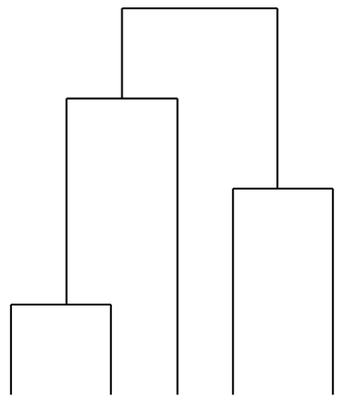
Flat algorithms

- Usually start with a random (partial) partitioning
- Refine it iteratively
  - *K*-means clustering
  - Model based clustering
- Spectral clustering



Hierarchical algorithms

- Bottom-up, agglomerative
- Top-down, divisive



Hard vs soft clustering

Hard clustering: Each example belongs to exactly one cluster

Soft clustering: An example can belong to more than one cluster

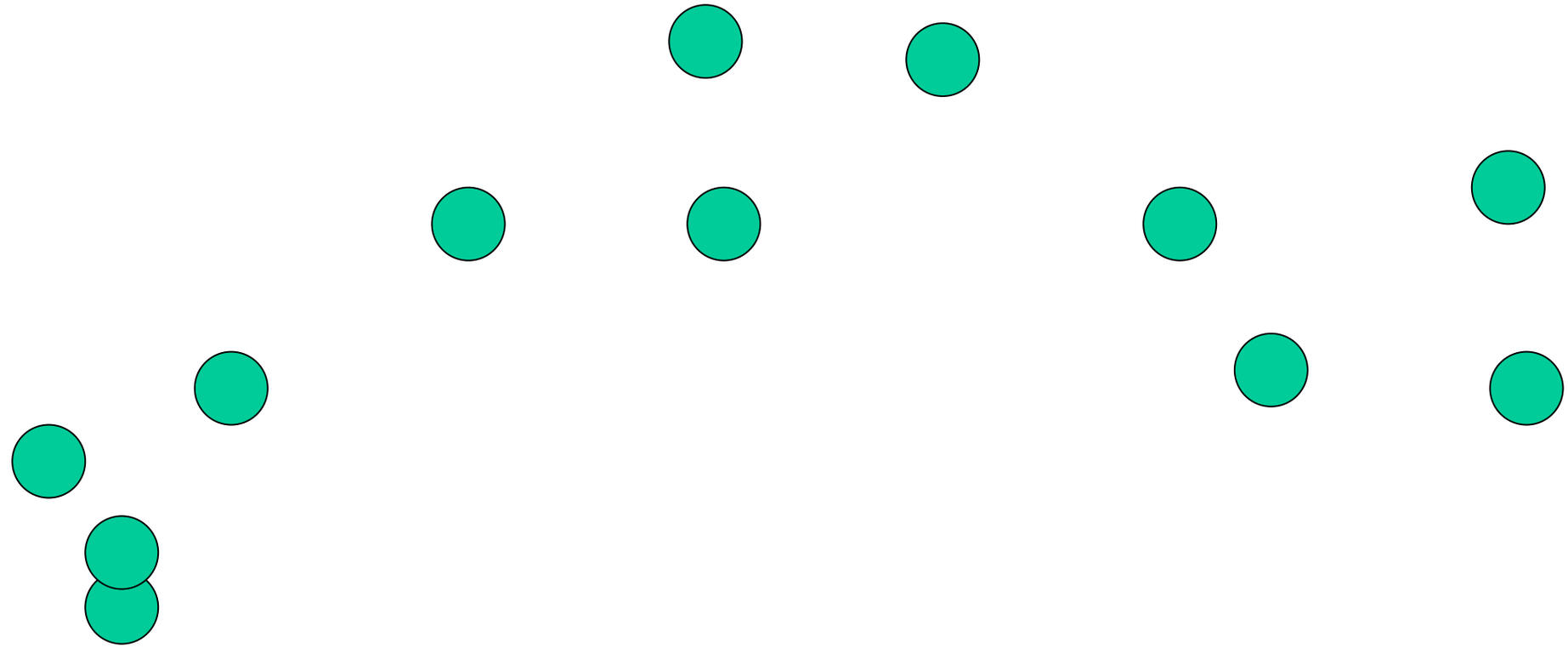
- Makes more sense for applications like creating browsable hierarchies
- You may want to put a pair of sneakers in two clusters:
  - (i) sports apparel
  - (ii) shoes
- often probabilistic

K-means clustering

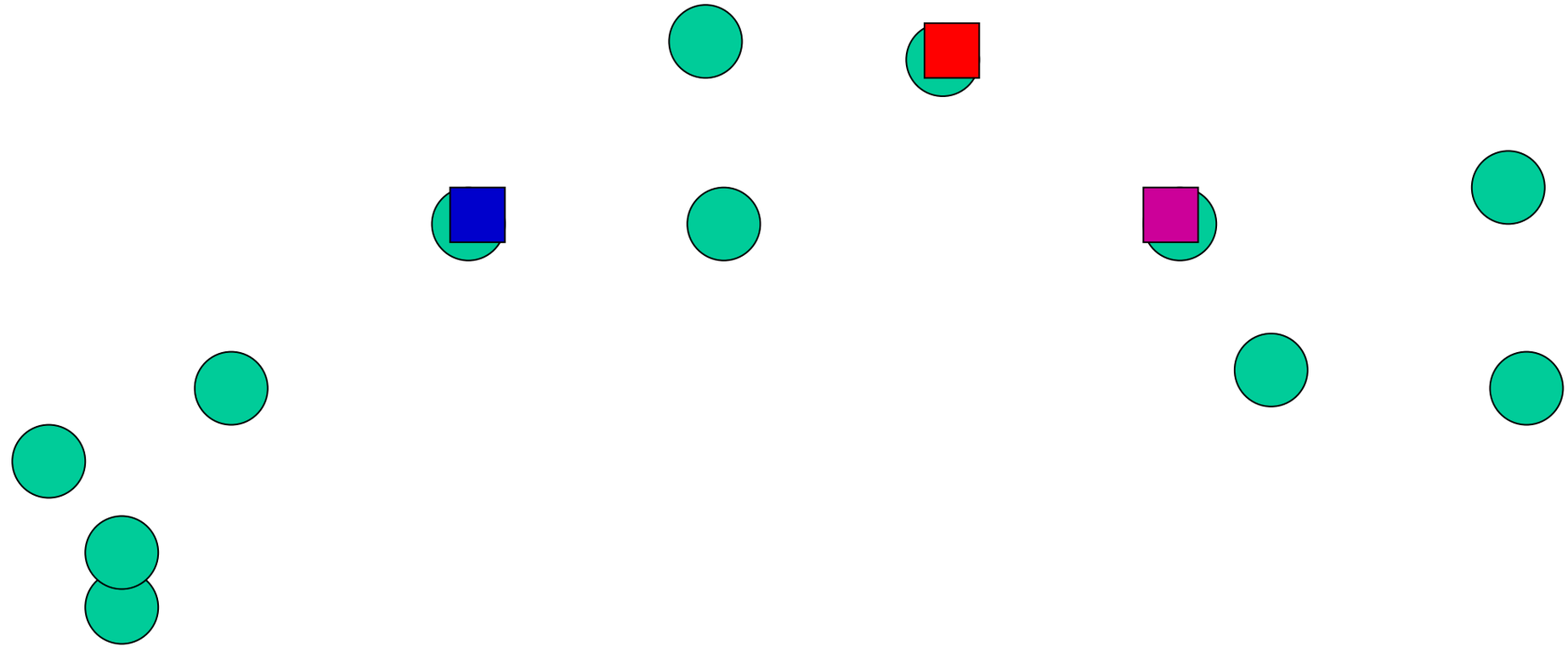
Most well-known and popular clustering algorithm

- Start with some initial cluster centers
- Iterate:
  - Assign/cluster each example to closest center
  - Recalculate centers as the mean of the points in a cluster

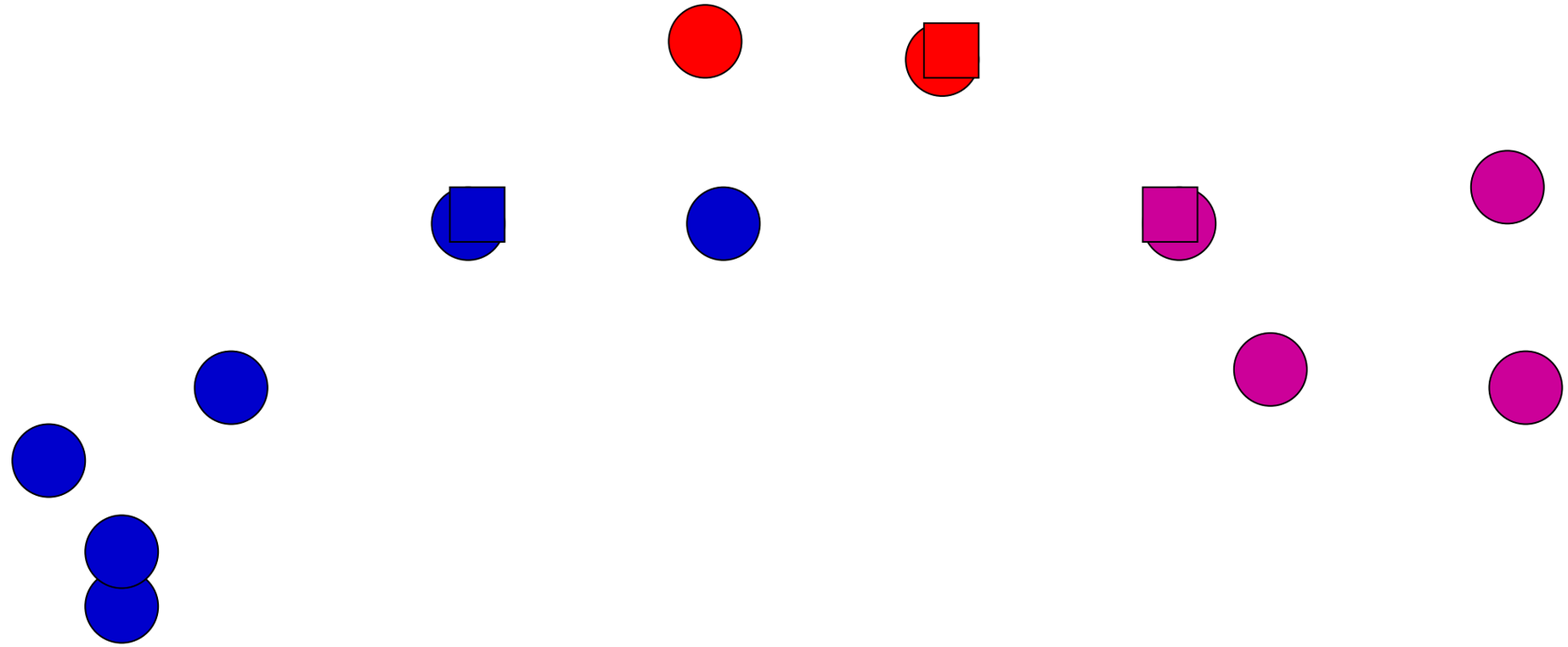
K-means: an example



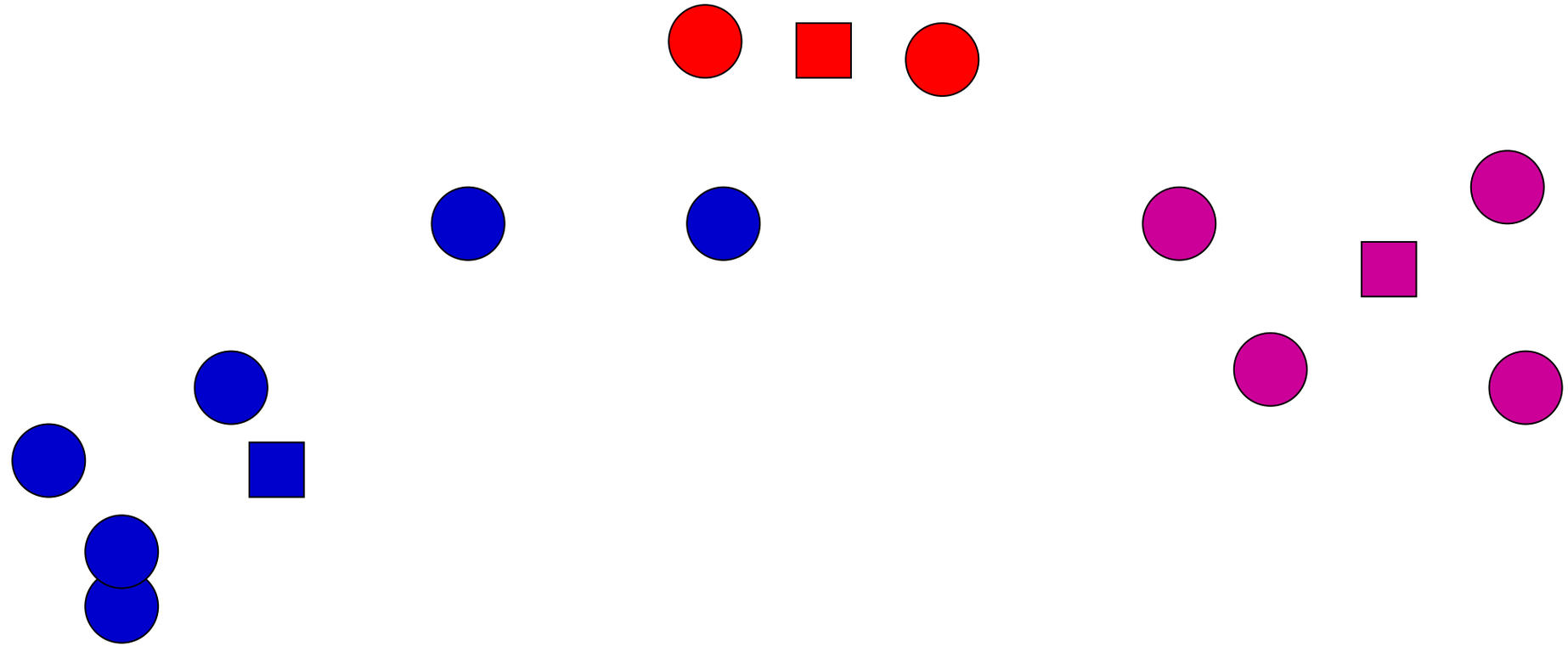
K-means: Initialize centers randomly



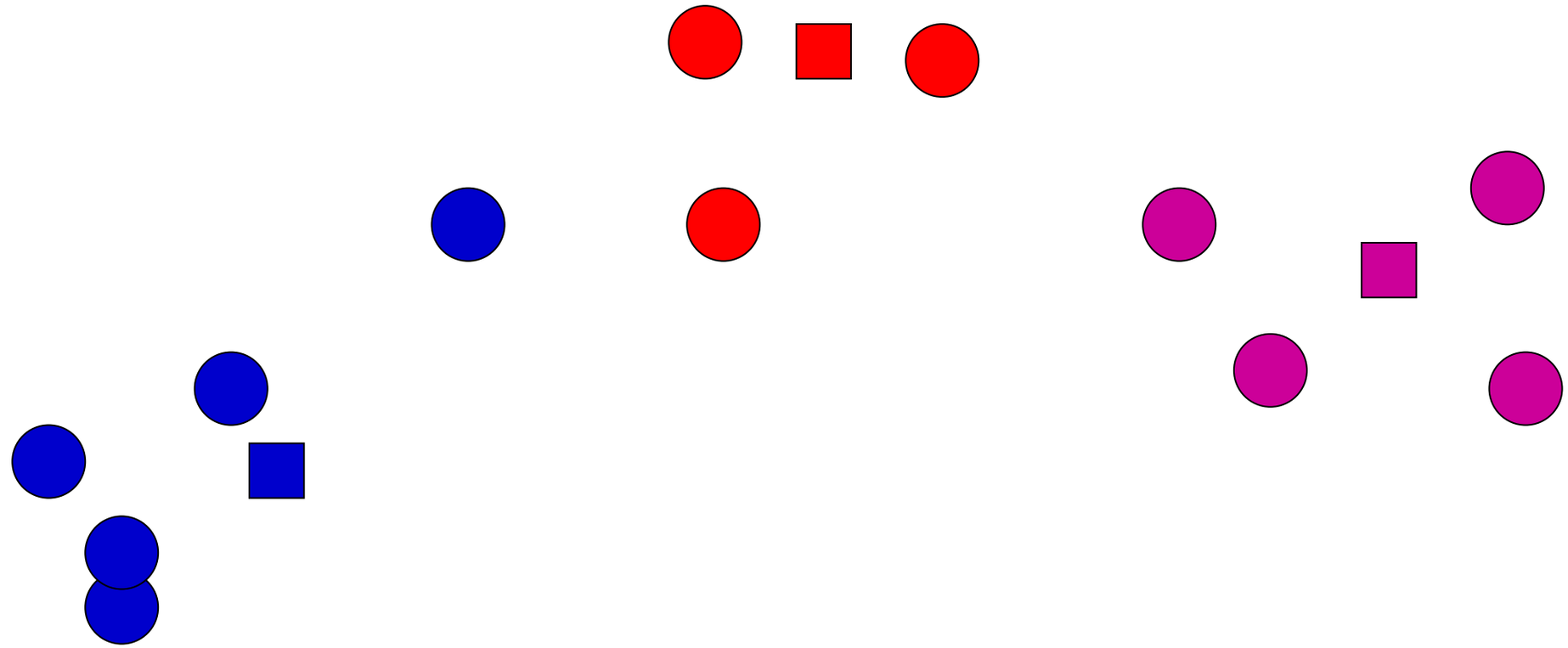
K-means: assign points to nearest center



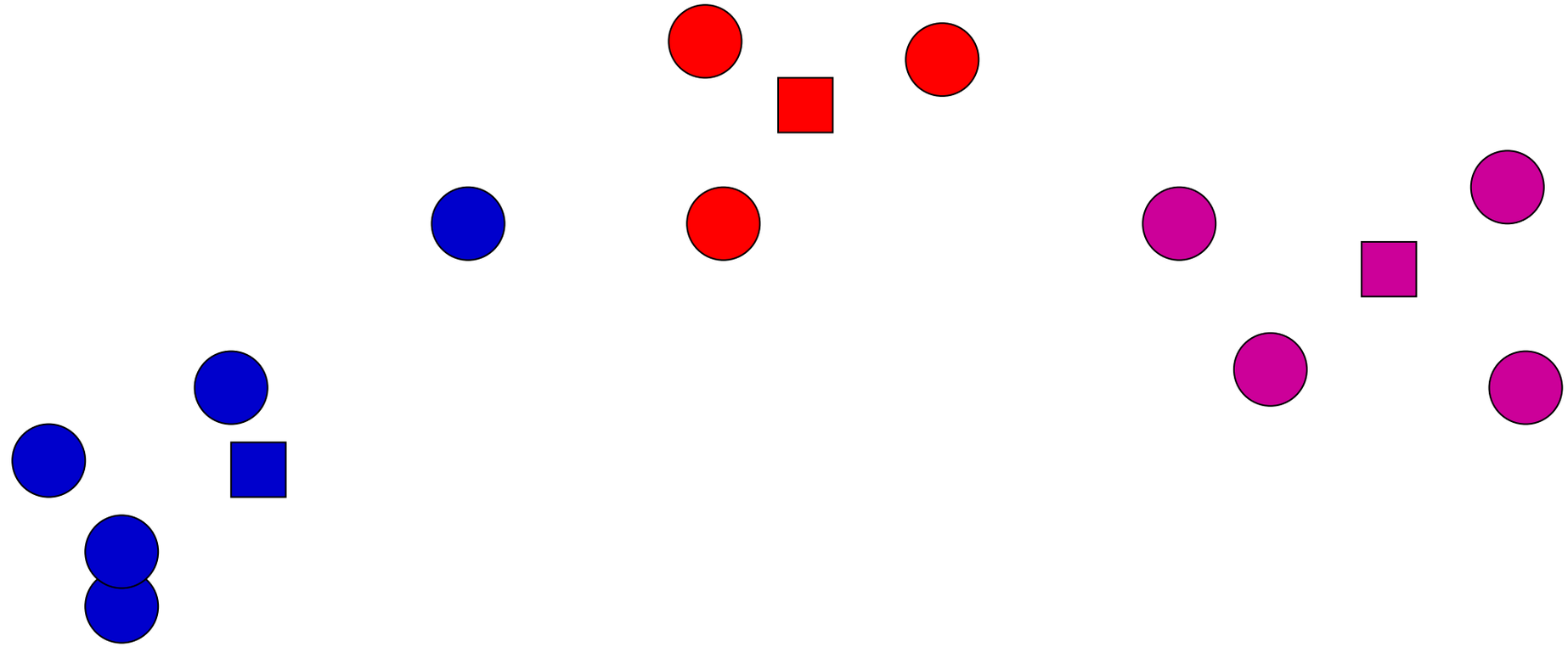
K-means: readjust centers



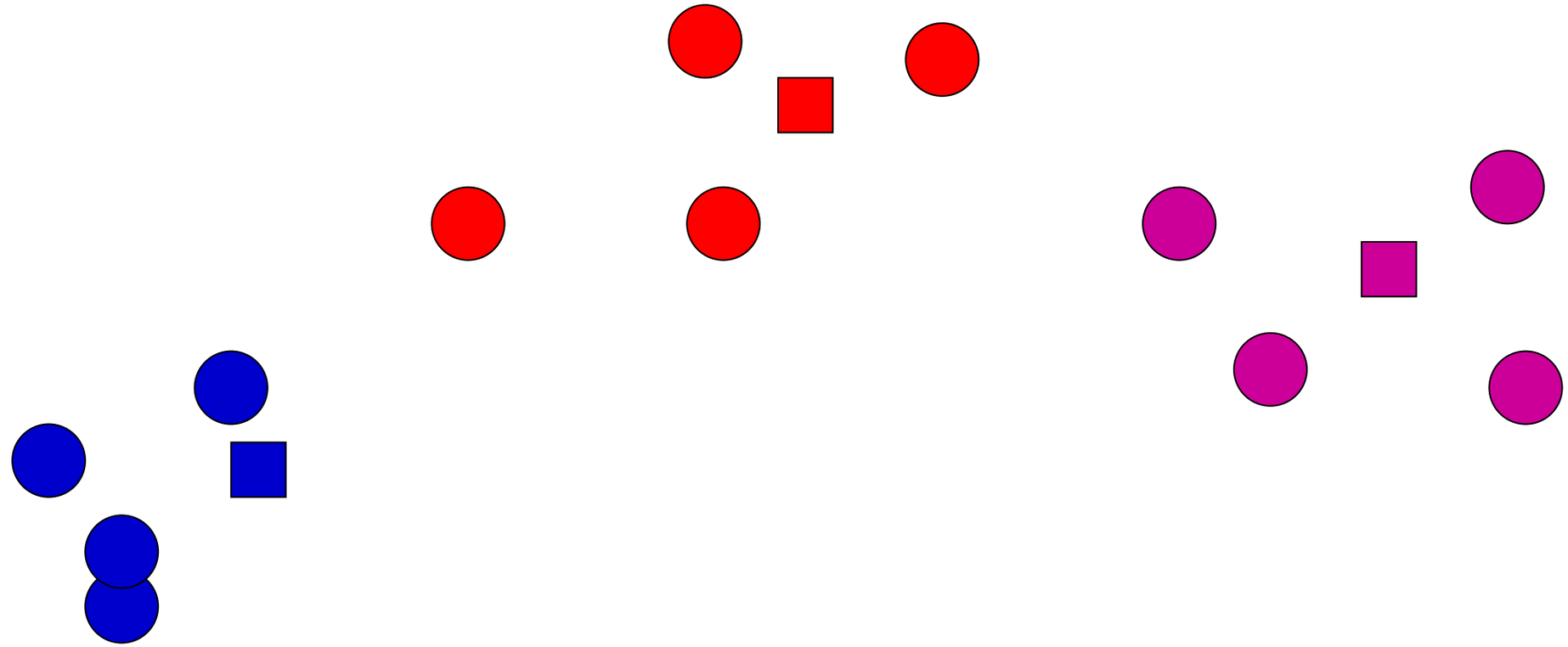
K-means: assign points to nearest center



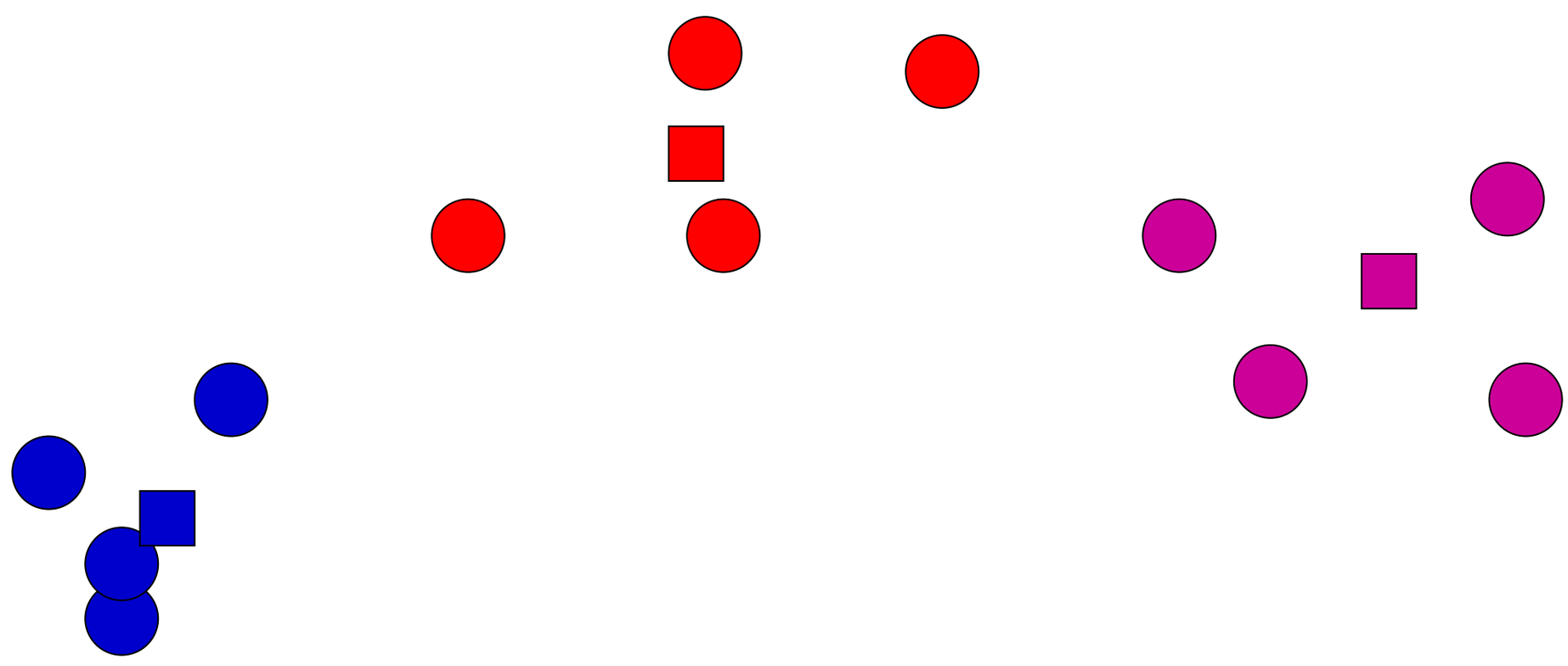
K-means: readjust centers



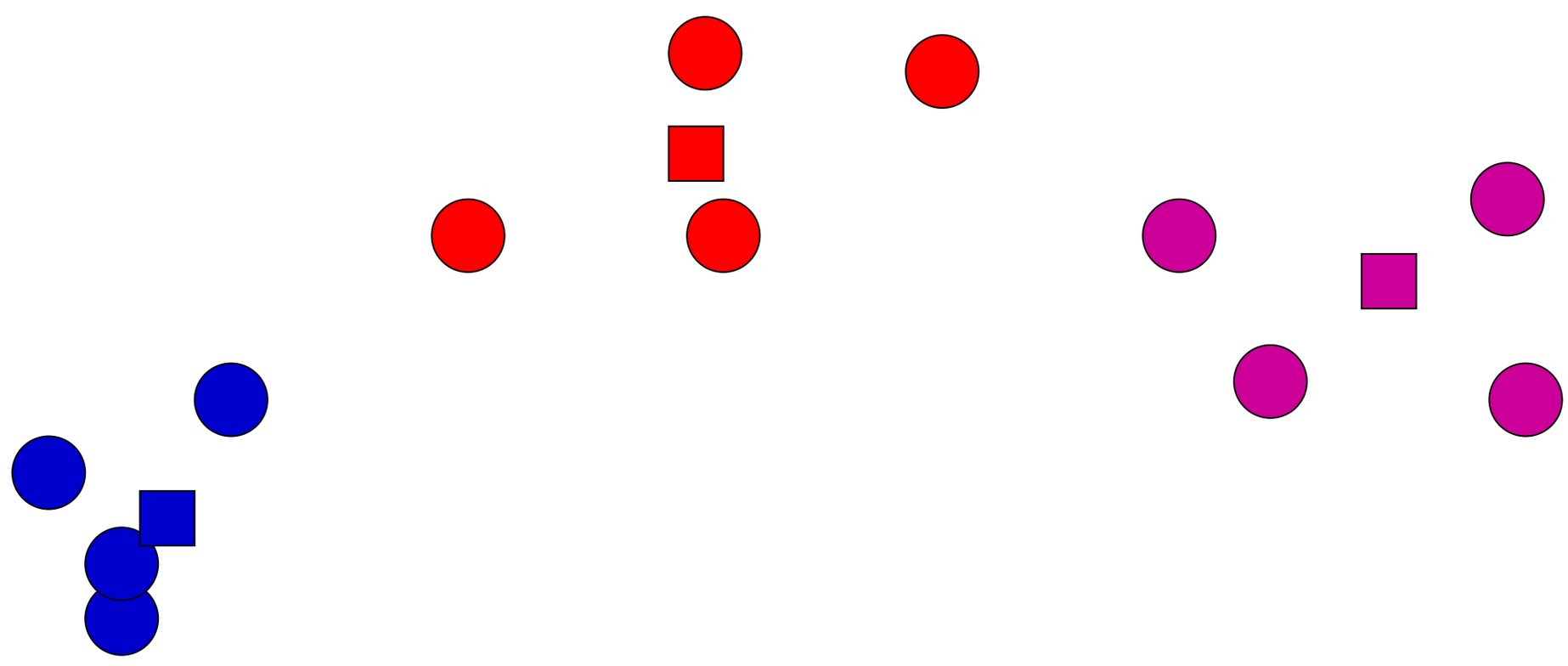
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center

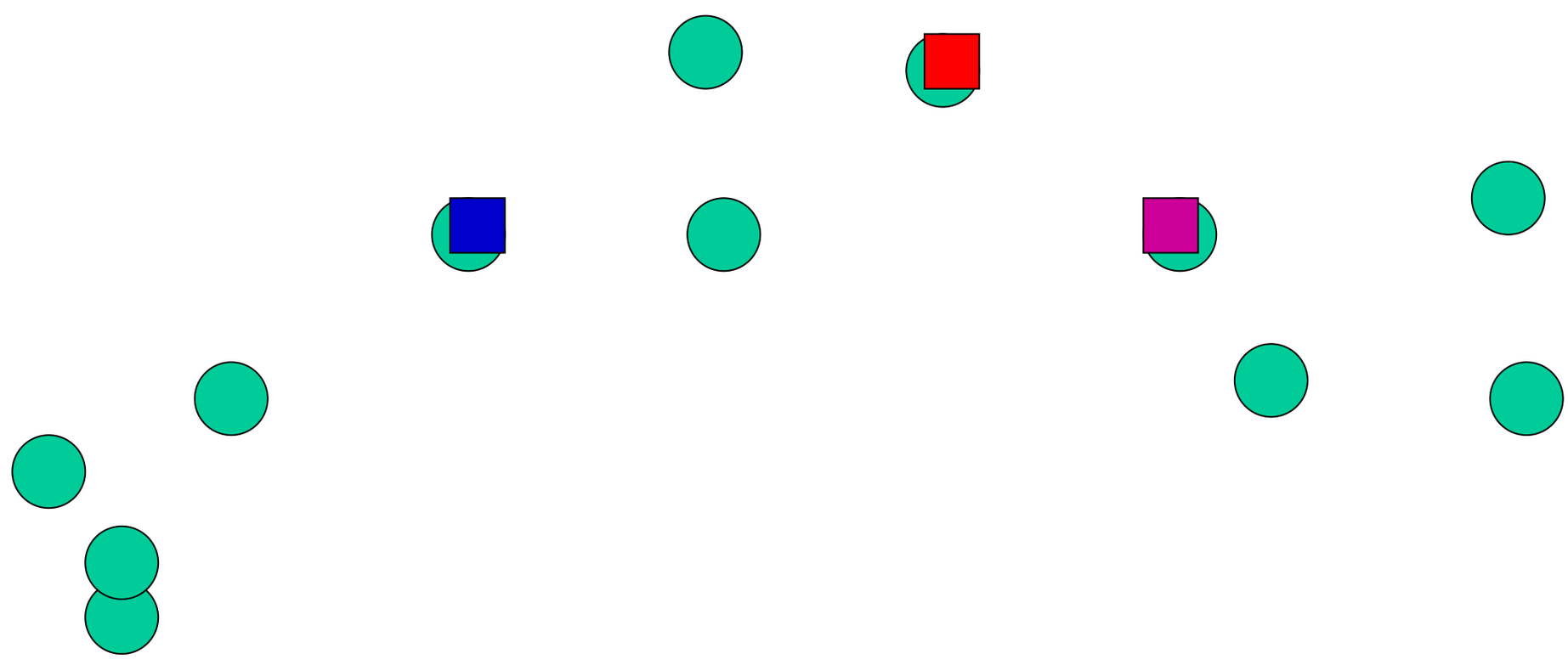


No changes: Done

K-means

Iterate:

- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



How do we do this?

K-means

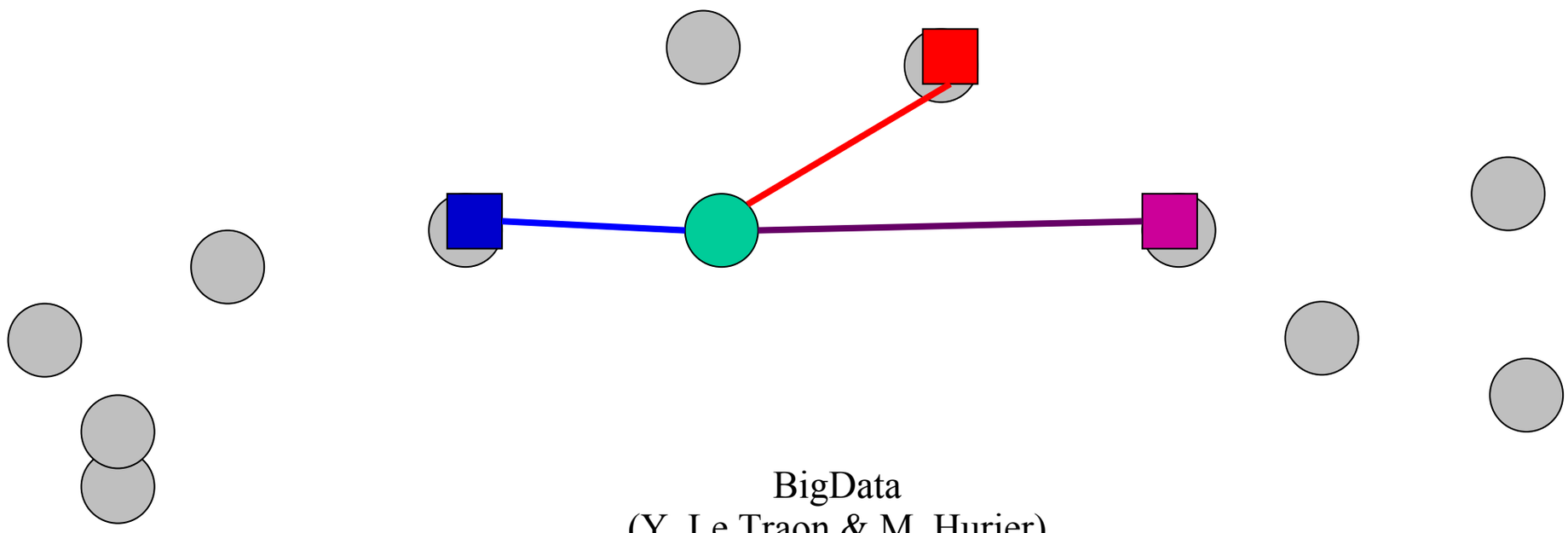
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get distance to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



K-means

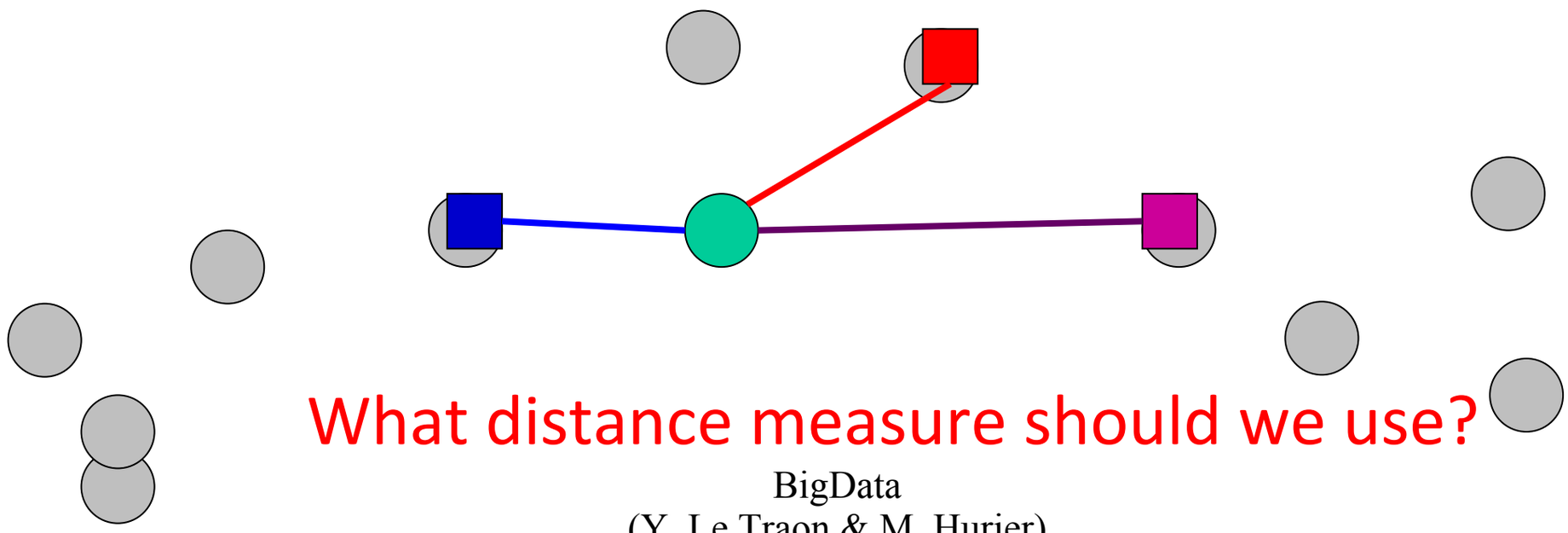
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get **distance** to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



What distance measure should we use?

Distance measures example

Euclidean:

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

good for spatial data

K-means

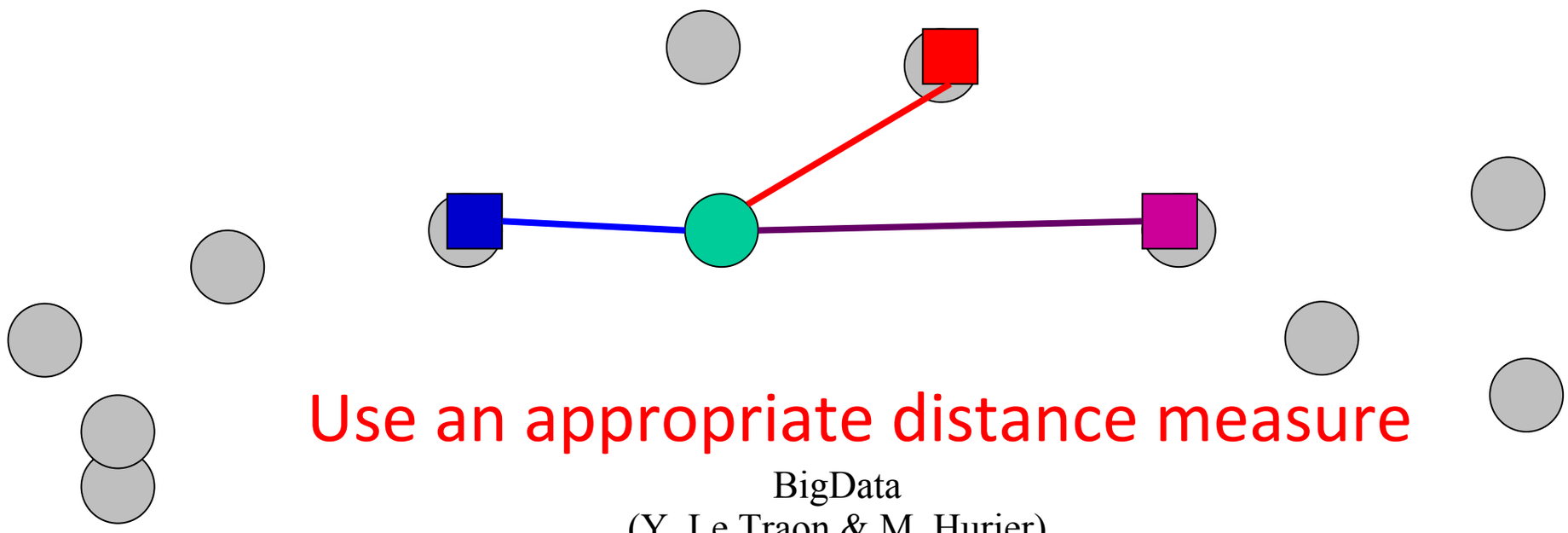
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get **distance** to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster

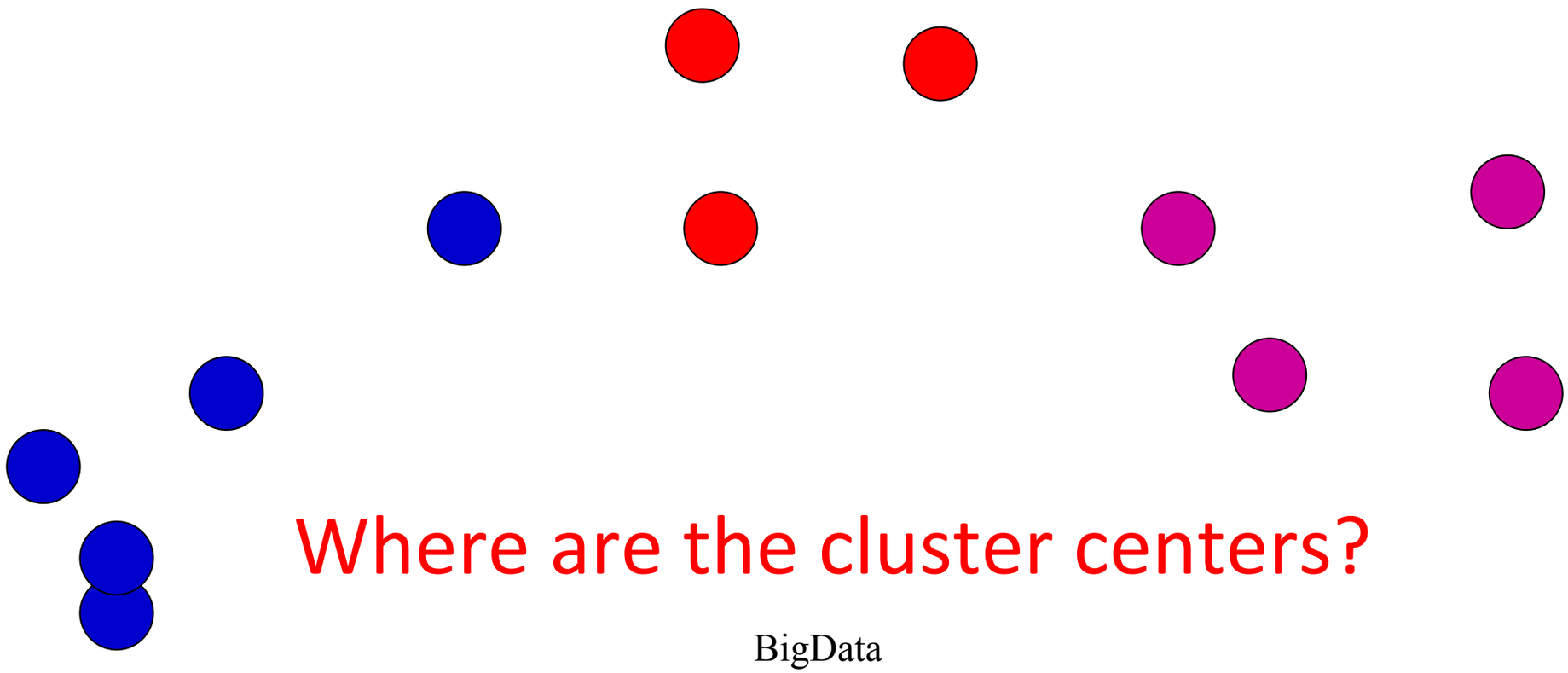


Use an appropriate distance measure

K-means

Iterate:

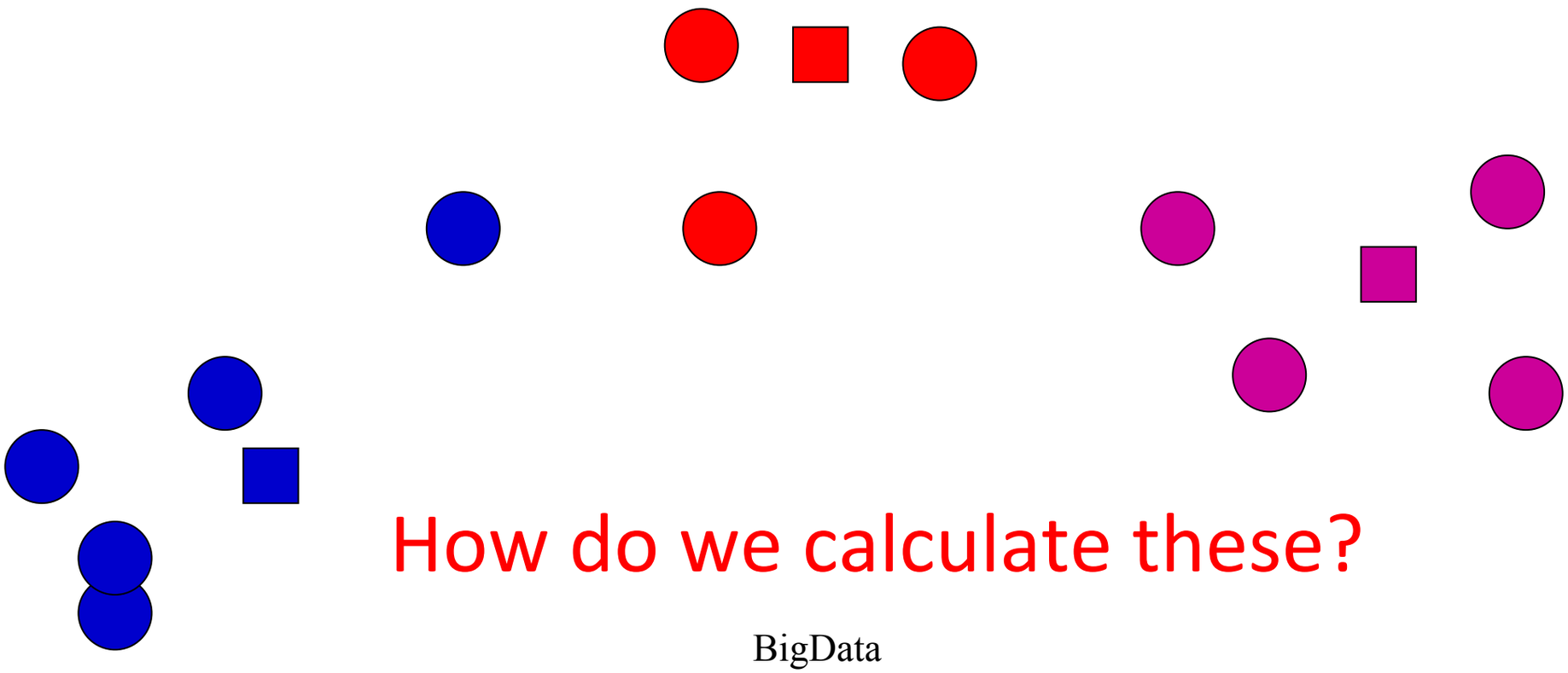
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

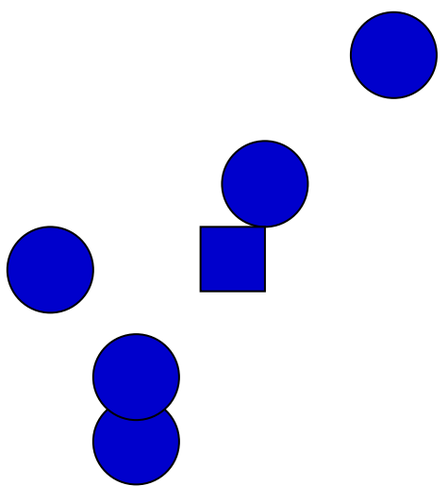


How do we calculate these?

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



Mean of the points in the cluster:

$$\mu(\mathrm{C}) = \frac{1}{|C|}\sum_{x \in C} x$$

where:

$$x + y = \sum_{i=1}^{n} x_i + y_i \qquad \frac{x}{|C|} = \sum_{i=1}^{n} \frac{x_i}{|C|}$$

K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does each step of k-means move towards reducing this loss function (or at least not increasing)?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

This isn't quite a complete proof/argument, but:

1. Any other assignment would end up in a larger loss

1. The mean of a set of values minimizes the squared error

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does this mean that k-means
will always find the minimum loss/clustering?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

---

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

NO! It will find *A minimum*.

Unfortunately, the k-means loss function is generally not convex and for most problems has many, many minima

We're only guaranteed to find one of them

K-means variations/parameters

Start with some initial cluster centers

Iterate:

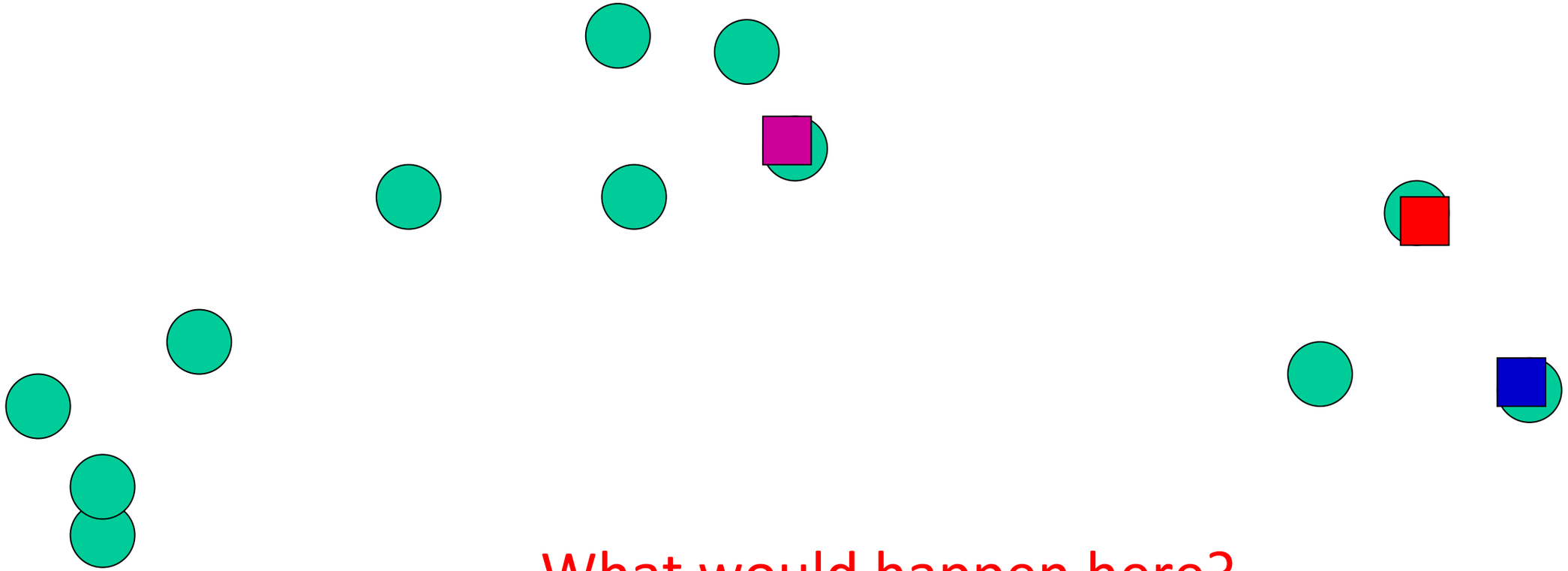
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

What are some other variations/parameters we haven't specified?

K-means variations/parameters

- Initial (seed) cluster centers
- Convergence
  - A fixed number of iterations
  - partitions unchanged
  - Cluster centers don’t change
- **K!**

K-means: Initialize centers randomly



What would happen here?

Seed selection ideas?

Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

Common heuristics:

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- **Try out multiple starting points**
- Initialize with the results of another clustering method

Furthest centers heuristic

μ_1 = pick random point

for i = 2 to K:

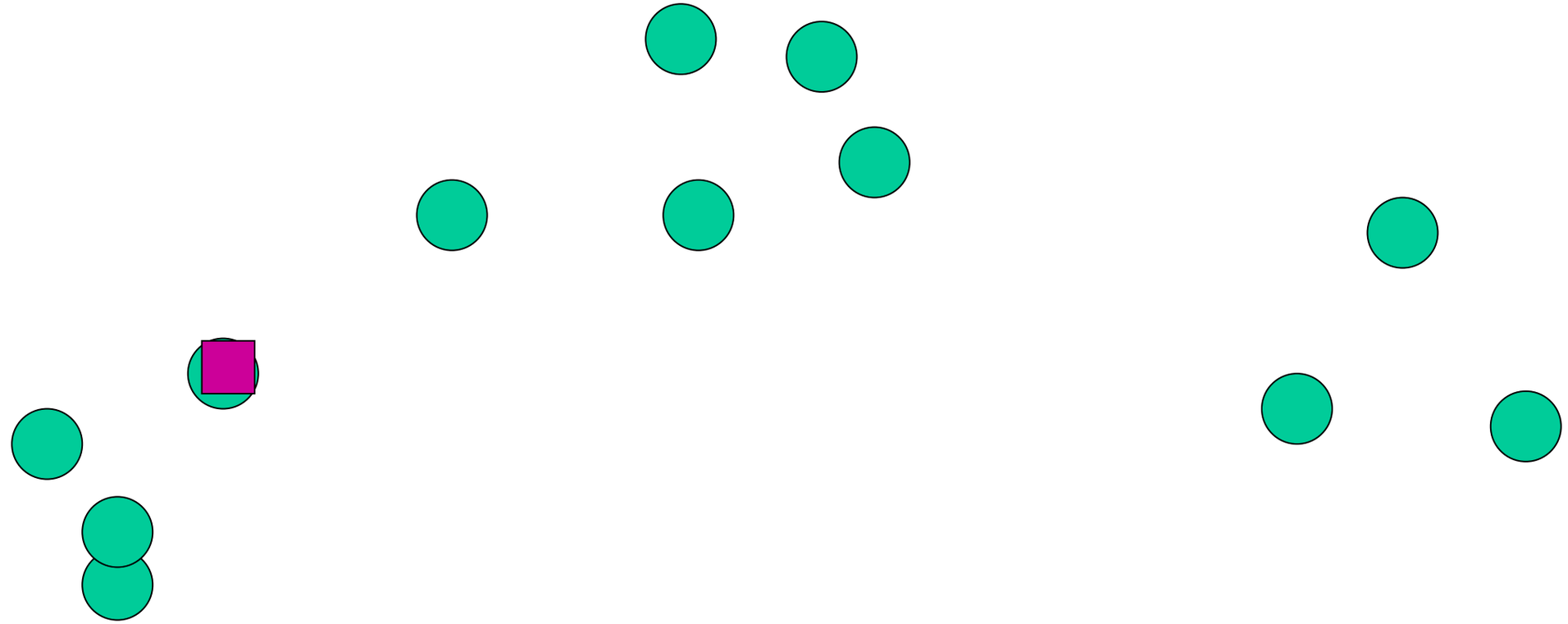
μ_i = point that is furthest from **any** previous centers

$$\mu_i = \underset{x}{\arg\max} \; \min_{\mu_j : 1 < j < i} d(x, \mu_j)$$

point with the largest distance to any previous center

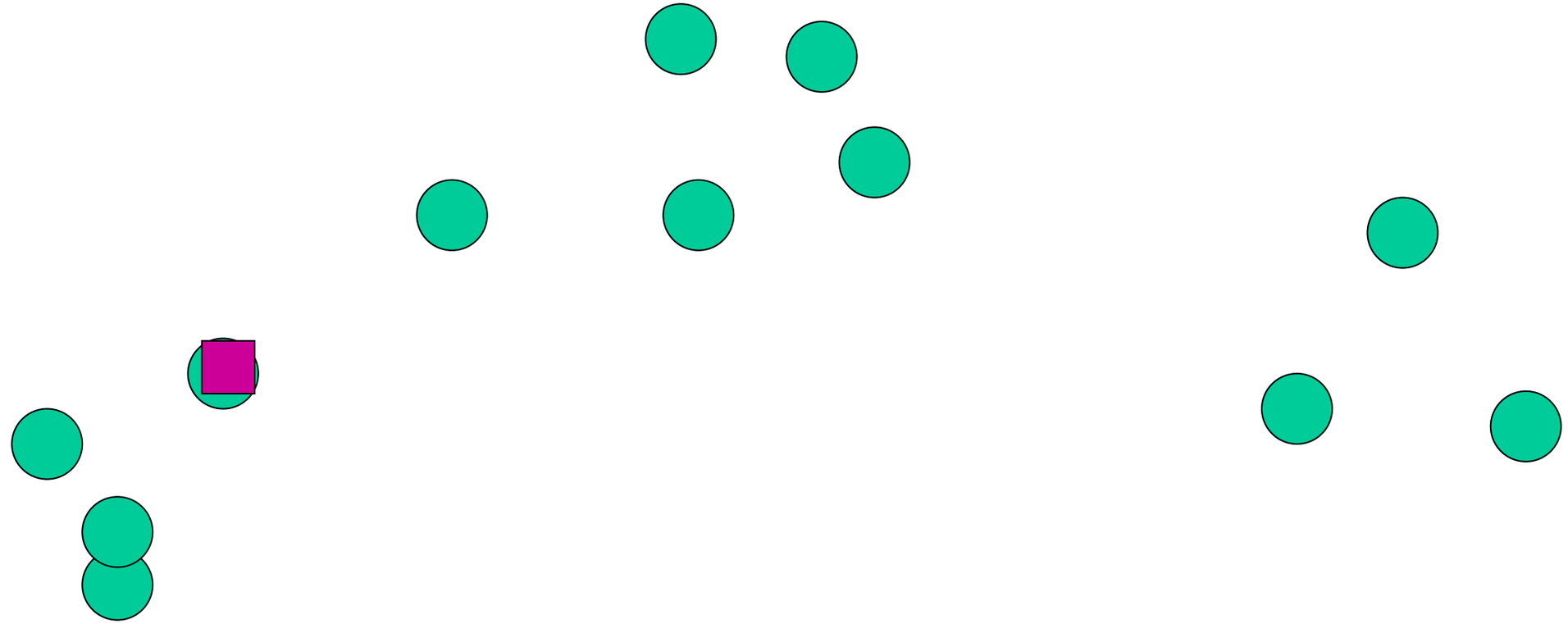
smallest distance from x to any previous center

K-means: Initialize furthest from centers



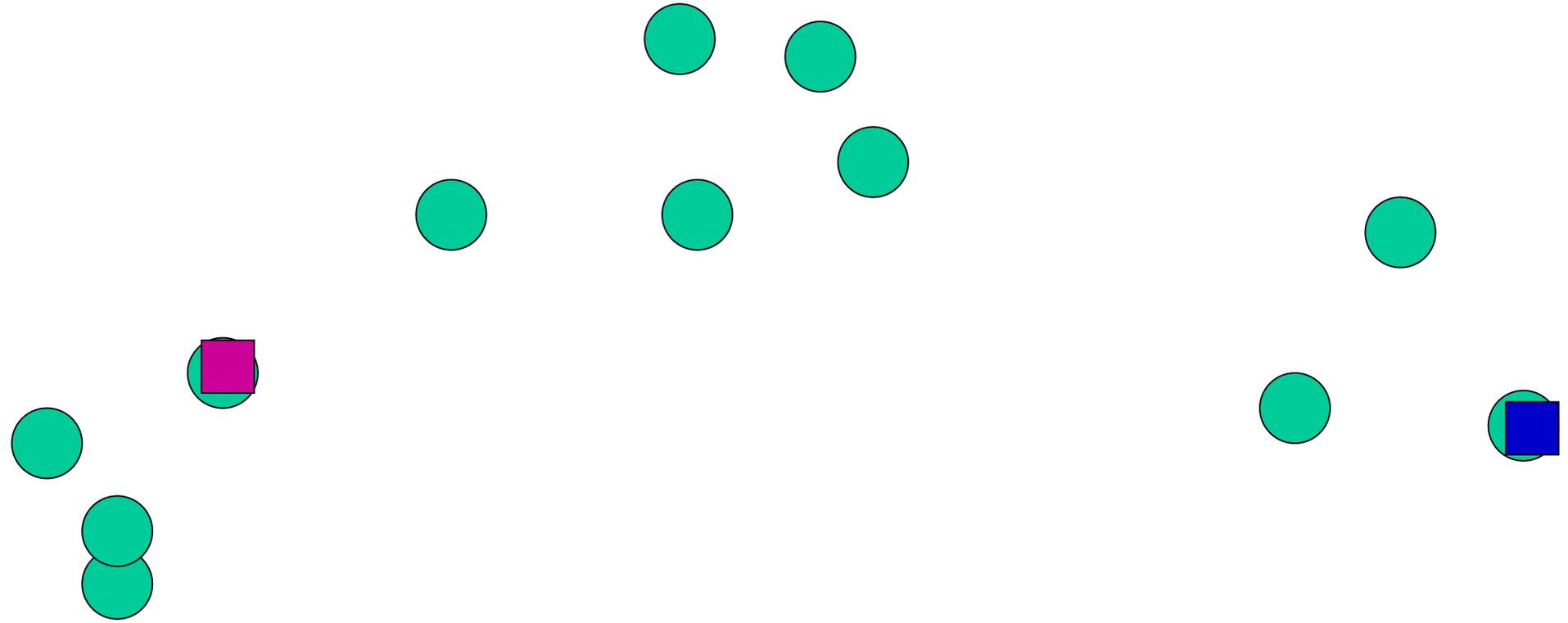
Pick a random point for the first center

K-means: Initialize furthest from centers



What point will be chosen next?

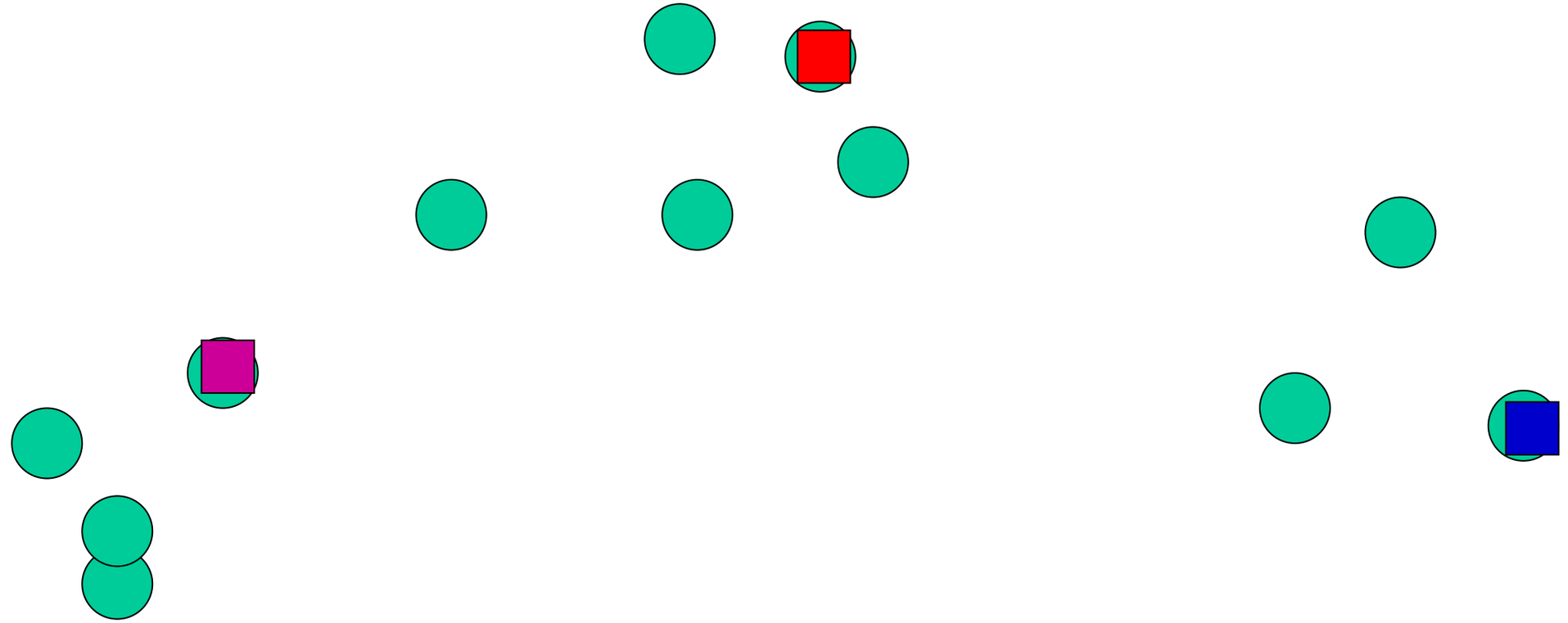
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

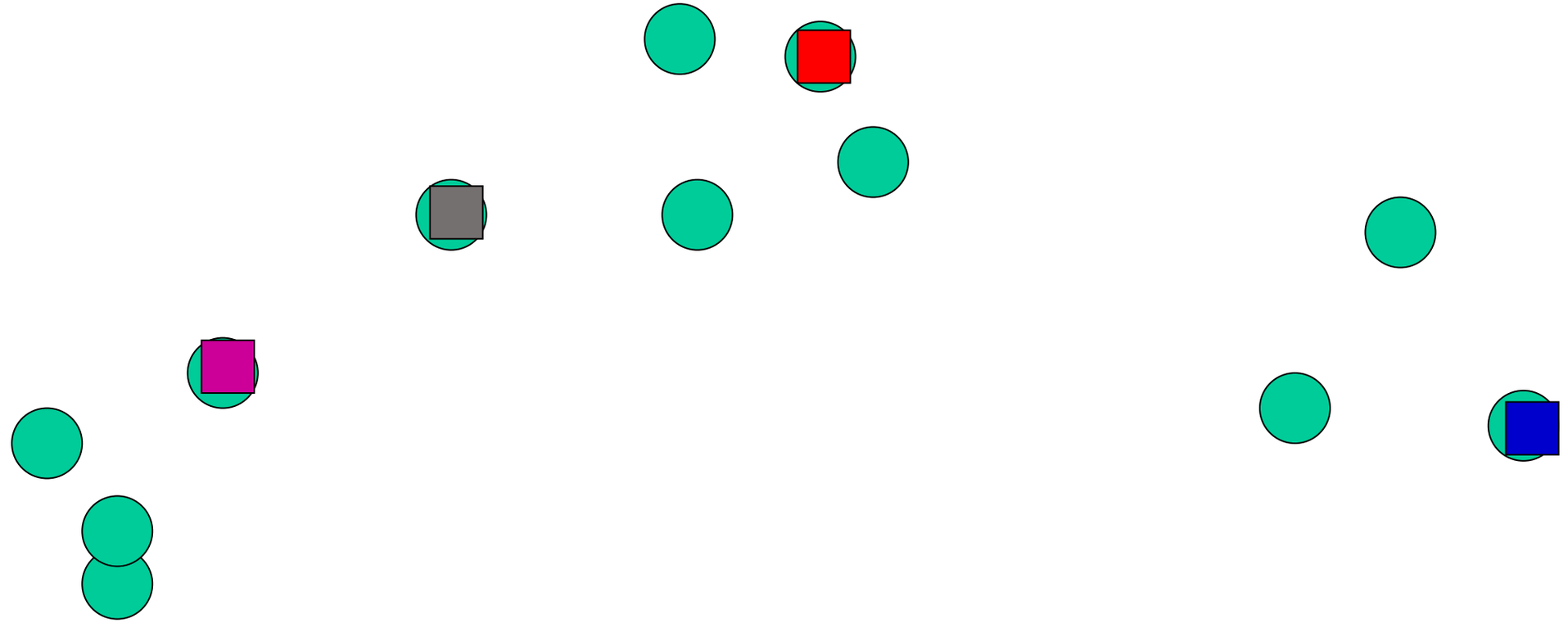
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

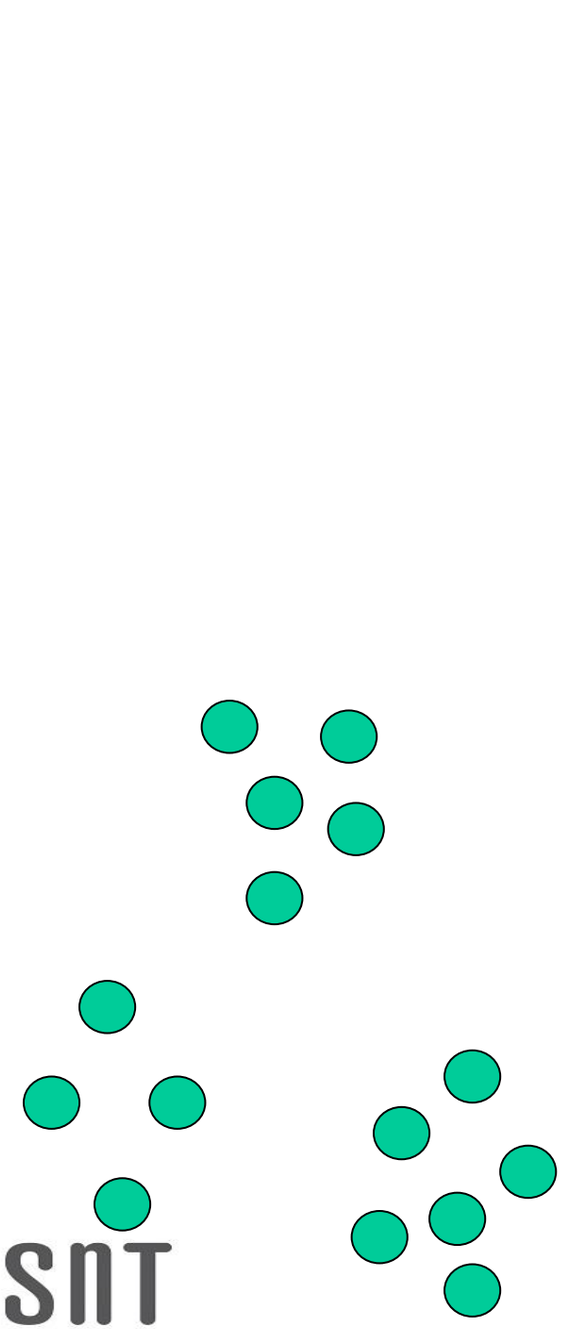
K-means: Initialize furthest from centers



Furthest point from center

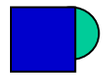
Any issues/concerns with this approach?

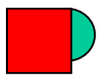
Furthest points concerns

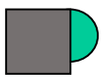


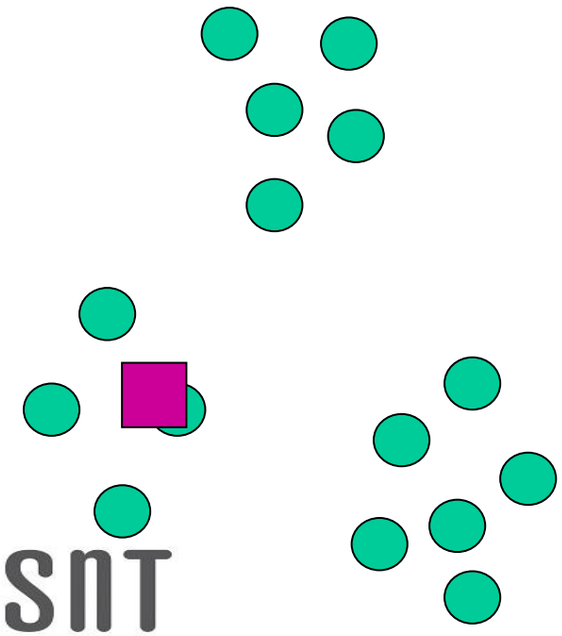
If k = 4, which points will get chosen?

Furthest points concerns



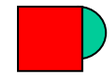


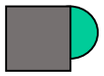


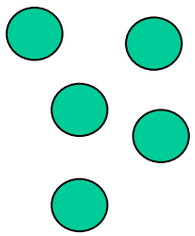


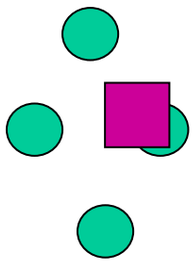
If we do a number of trials,
will we get different centers?

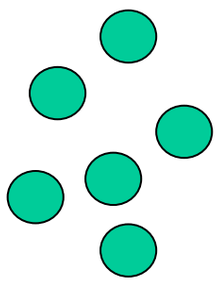
Furthest points concerns











Doesn’t deal well with outliers

K-means++

μ_1 = pick random point

for k = 2 to **K**:

    for i = 1 to **N**:

$s_i = \min d(x_i, \mu_{1\ldots k-1})$ // smallest distance to any center

μ_k = randomly pick point proportionate to s

How does this help?

K-means++

μ_1 = pick random point

for k = 2 to **K**:

    for i = 1 to **N**:

$s_i = \min d(x_i, \mu_{1\ldots k-1})$ // smallest distance to any center

μ_k = randomly pick point proportionate to s

- Makes it possible to select other points
  - if #points >> #outliers, we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees!

K-means for clustering

Original image

159 KB

Clustered, k=4

53 KB

Issue 1 with k-means

- K-means is greedy!
- Converging to a non-global optimum:

Issue 1 with k-means

- K-means is greedy!
- Converging to a non-global optimum:

Issue 2 with k-means

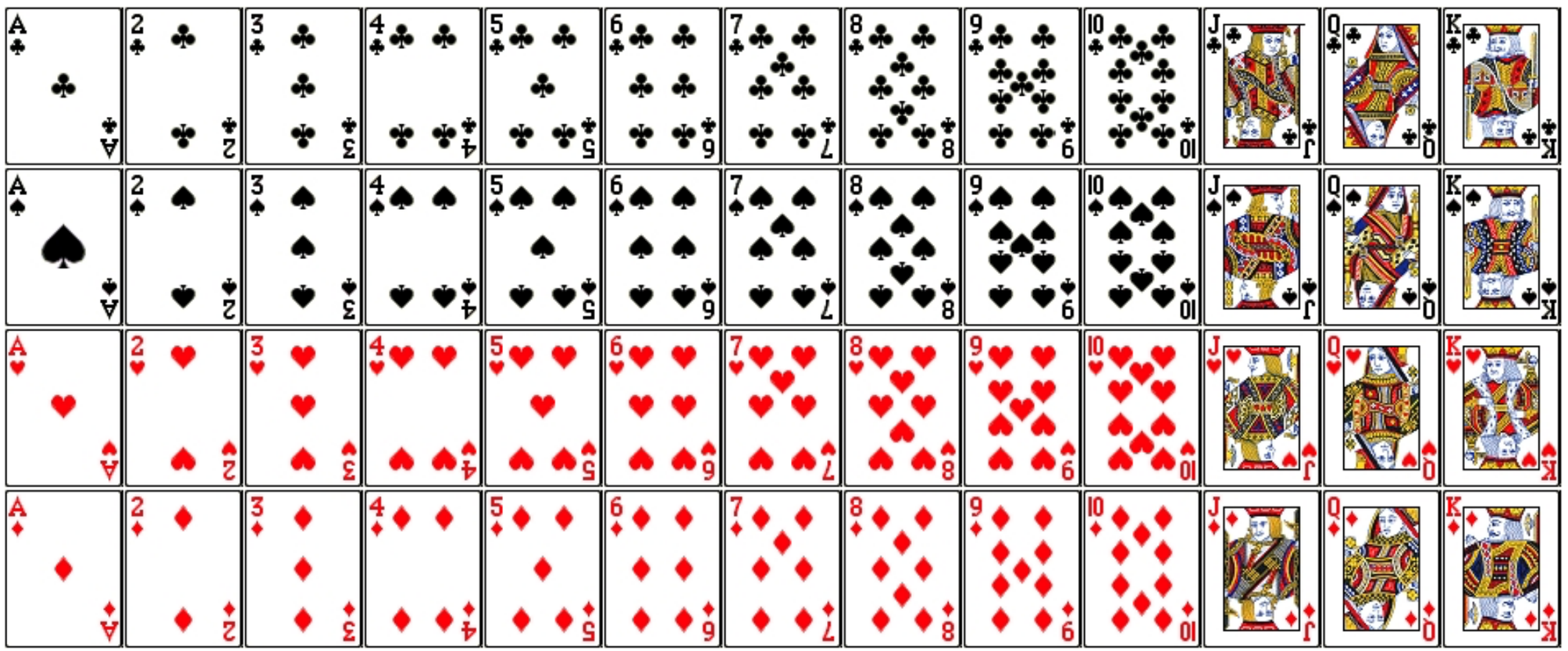
How long will it take? We don't know!

- K-means is O(nkdI)
  - with d = # features (dimensionality), I =# iterations
- # iterations depends on random initialization
  - "Good" init: few iterations
  - "Bad" init: lots of iterations
- How can we tell the difference, before clustering?
  - we can't !
  - use heuristics to guess "good" init

Issue 3 with k-means

How many clusters?

The “Holy Grail” of clustering



Unsupervised learning: modeling

When people think of unsupervised learning, they frequently think about clustering

Another category: learning probabilities/parameters for models without supervision

- Learn a translation dictionary
- Learn a grammar for a language
- Learn the social graph

Reinforcement Learning

Machine Learning

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Classification	Clustering	Decision Process
Regression	Segmentation	Reward System
Ranking	Dimension Reduction	Recommendation Systems
	Association Mining	

Reinforcement learning

Automatically determine the ideal behavior within a specific context

- Current environment/state matters
- Reward feedback = reinforcement signal
- Closer to the fields of Artificial Intelligence

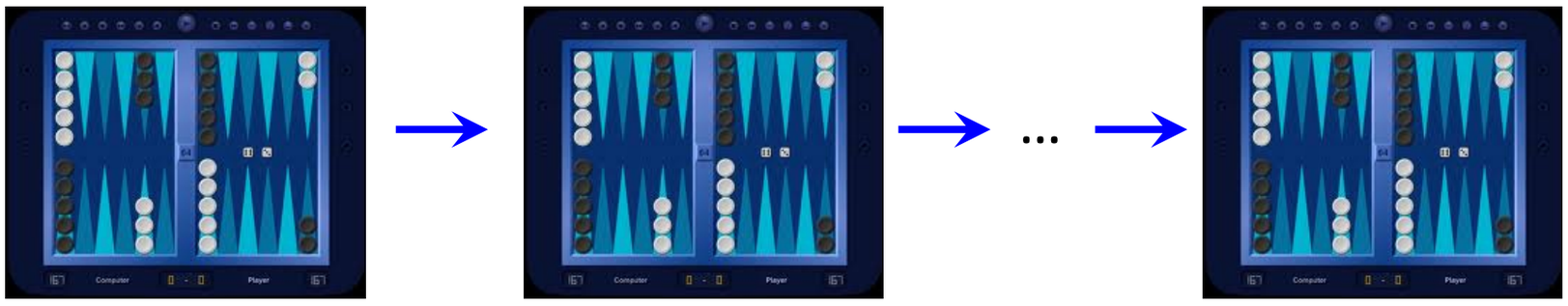
Reinforcement learning

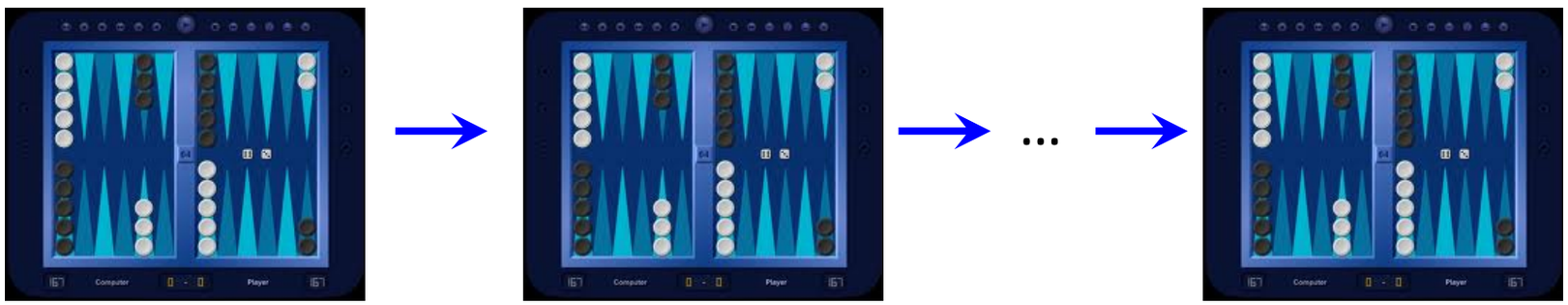
left, right, straight, left, left, left, straight	**GOOD**
left, straight, straight, left, right, straight, straight	**BAD**
left, right, straight, left, left, left, straight	**18.5**
left, straight, straight, left, right, straight, straight	**-3**

Given a sequence of examples and a reward
learn to predict the action to take in for an individual example

Example: Reinforcement learning

Backgammon





Given sequences of moves and whether or not the player won at the end, learn to make good moves

Other learning variations

What data is available:

- Supervised, unsupervised, reinforcement learning
- Semi-supervised, active learning, …

How are we getting the data:

- online vs. offline learning

Type of model:

- generative vs. discriminative
- parametric vs. non-parametric

Thank you for your attention !

Questions ?