

CSE419 – Artificial Intelligence and Machine Learning 2021

PhD Furkan Gözükkara, Toros University

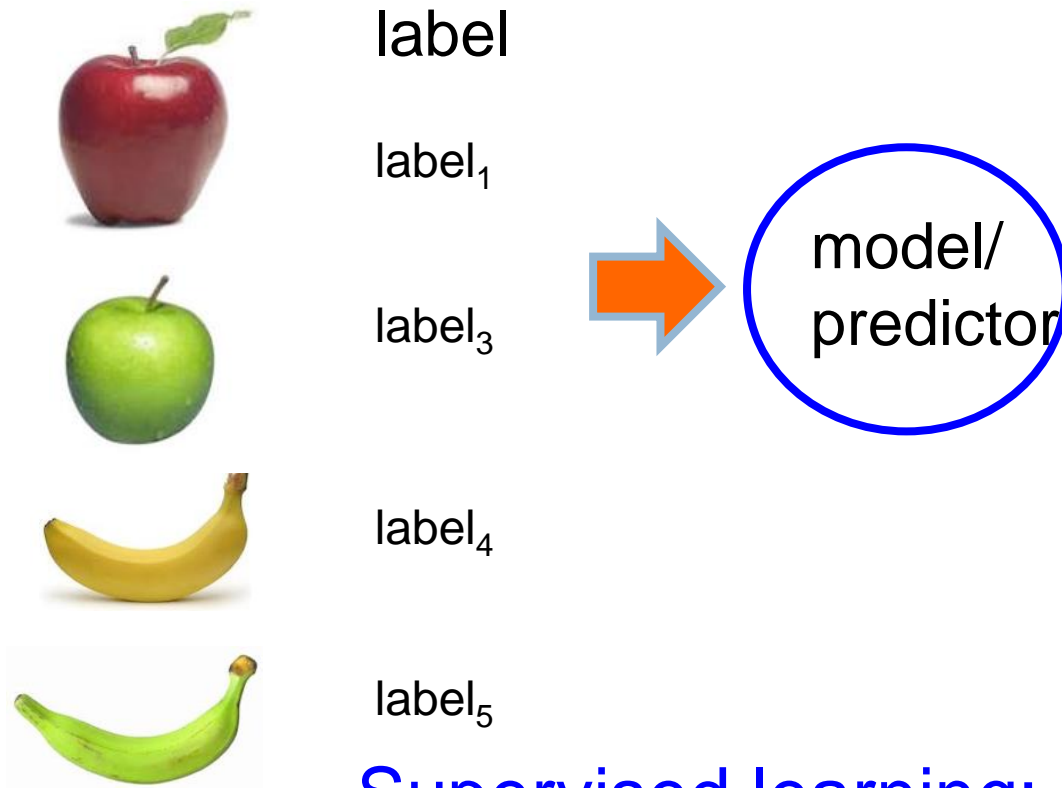
https://github.com/FurkanGozukara/CSE419_2021

Lecture 15

Unsupervised Learning

Based on Asst. Prof. Dr. David Kauchak (Pomona College) Lecture Slides

Supervised learning



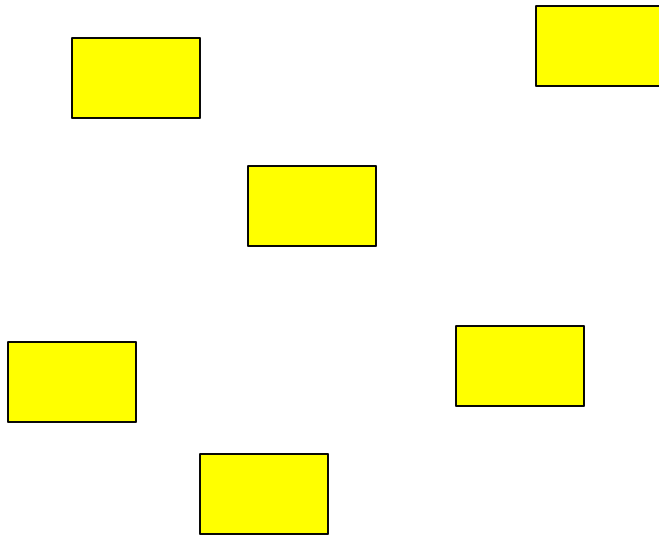
Supervised learning: given labeled examples

Unsupervised learning



Unsupervised learning: given data, i.e. examples, but no labels

Unsupervised learning



Given some example without labels, do something!

Unsupervised learning applications



learn clusters/groups without any label

customer segmentation (i.e. grouping)

image compression

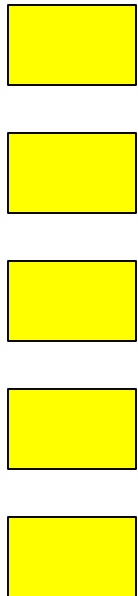
bioinformatics: learn motifs

find important features

...

Unsupervised learning: clustering

Raw data



extract
features

features

$f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$
 $f_1, f_2, f_3, \dots, f_n$

group into
classes/clu
sters

Clusters

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

Unsupervised learning: modeling

Most frequently, when people think of unsupervised learning they think clustering

Another category: learning probabilities/parameters for models without supervision

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

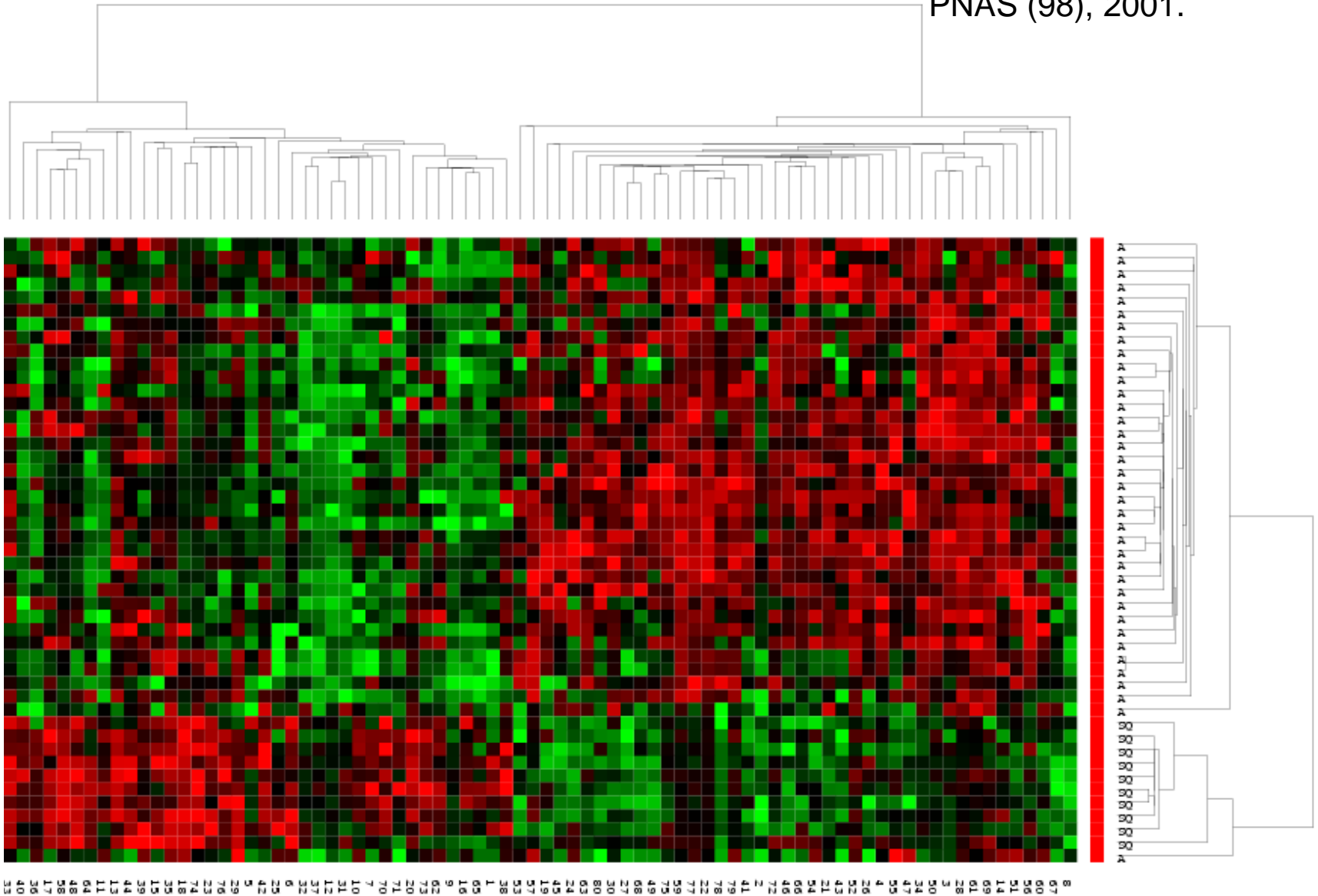
Clustering

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

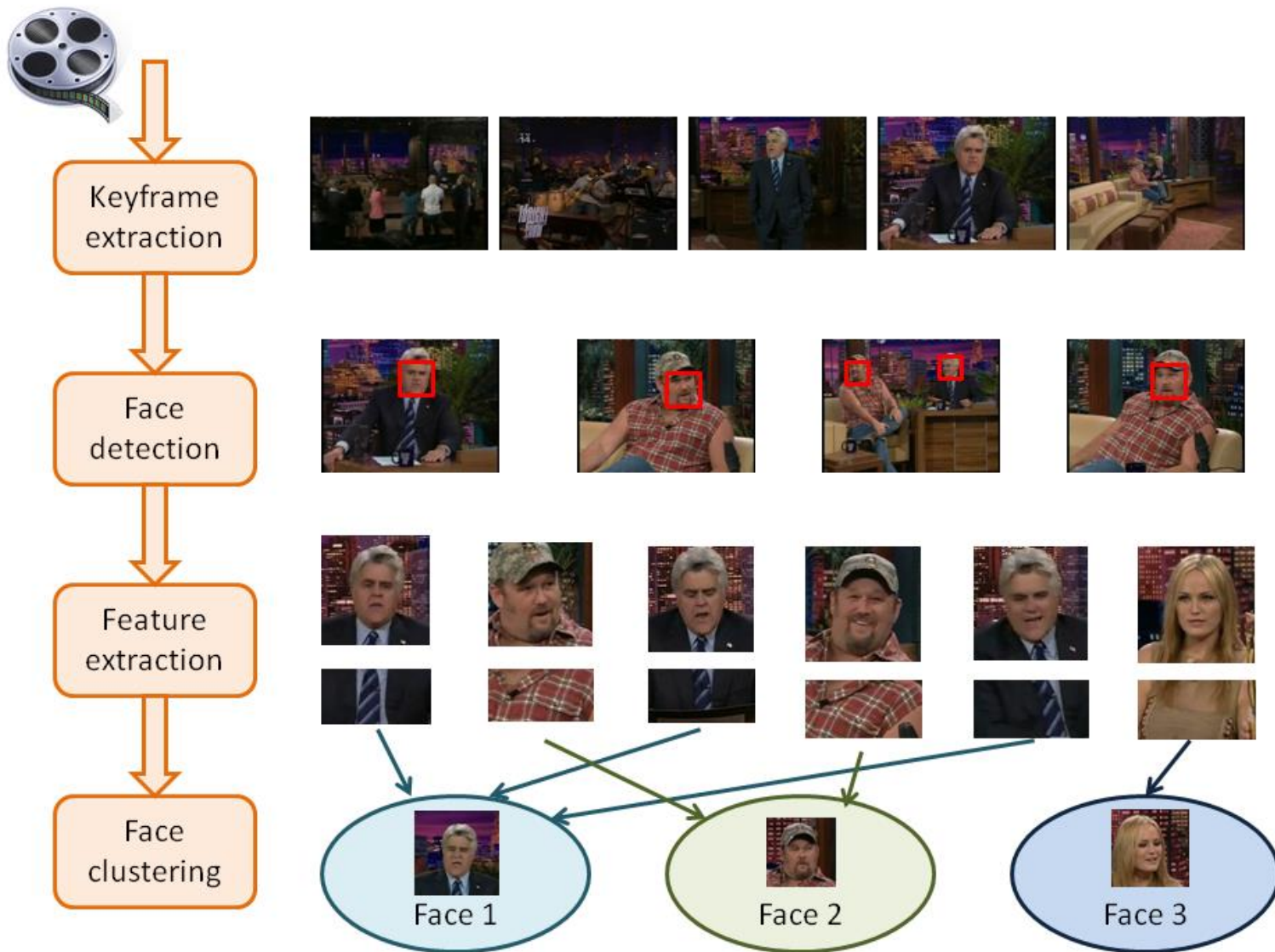
Applications?

Gene expression data

Data from Garber et al.
PNAS (98), 2001.



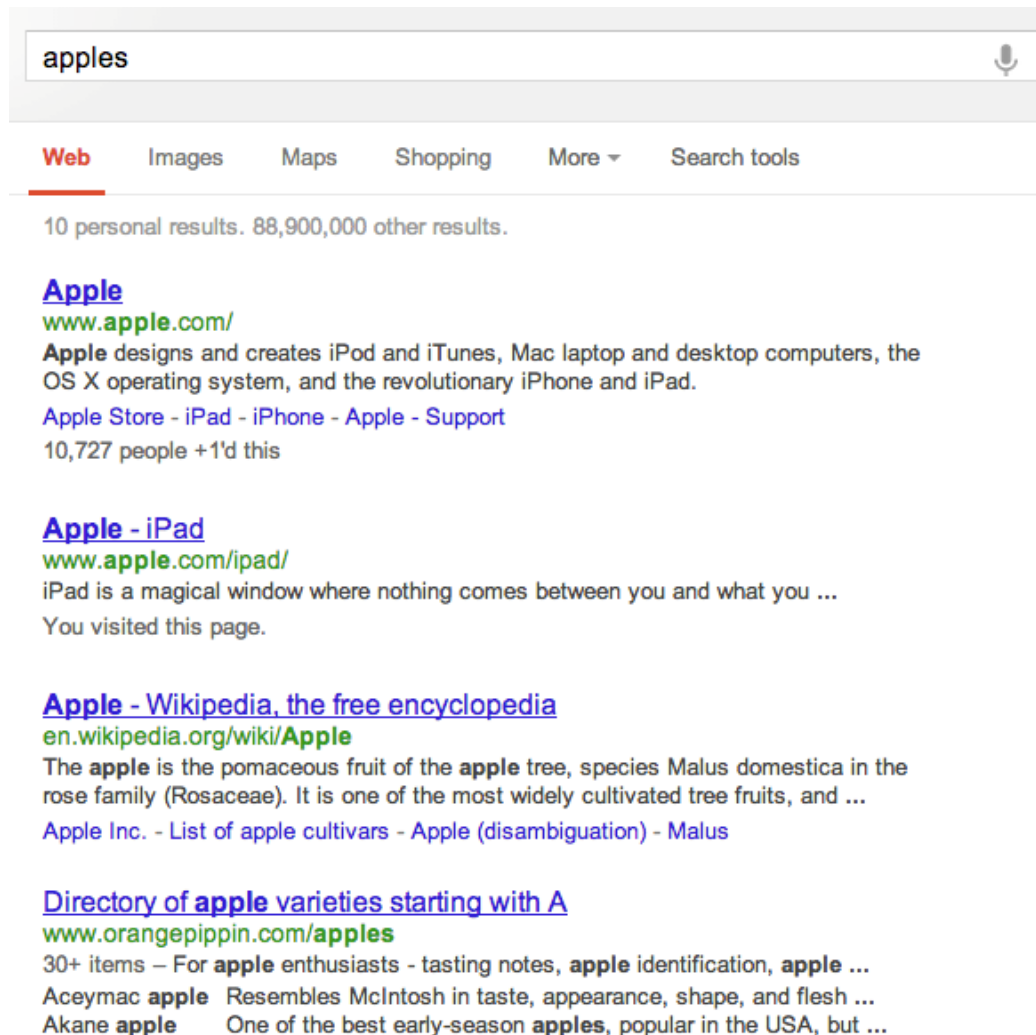
Face Clustering



Face clustering



Search result clustering



apples

Web Images Maps Shopping More Search tools

10 personal results. 88,900,000 other results.

Apple
www.apple.com/
Apple designs and creates iPod and iTunes, Mac laptop and desktop computers, the OS X operating system, and the revolutionary iPhone and iPad.
[Apple Store](#) - [iPad](#) - [iPhone](#) - [Apple](#) - [Support](#)
10,727 people +1'd this

Apple - iPad
www.apple.com/ipad/
iPad is a magical window where nothing comes between you and what you ...
You visited this page.

Apple - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Apple
The **apple** is the pomaceous fruit of the **apple** tree, species *Malus domestica* in the rose family (Rosaceae). It is one of the most widely cultivated tree fruits, and ...
[Apple Inc.](#) - [List of apple cultivars](#) - [Apple \(disambiguation\)](#) - [Malus](#)

Directory of apple varieties starting with A
www.orangeippin.com/apples
30+ items – For **apple** enthusiasts - tasting notes, **apple** identification, **apple** ...
Aceymac apple Resembles McIntosh in taste, appearance, shape, and flesh ...
Akane apple One of the best early-season **apples**, popular in the USA, but ...

Google News

Google

News

Top Stories

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

+ Xbox One



E! Online

See realtime coverage

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 ...

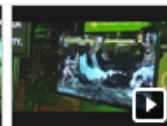
[Xbox One sales exceed one million in first 24 hours](#) Joystiq - by David Hinkle
[Xbox One vs. PS4: A Guide to Making the Toughest Gaming Decision in Years](#)

ABC News - by Joanna Stern

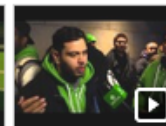
Related
[Microsoft »](#)



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...

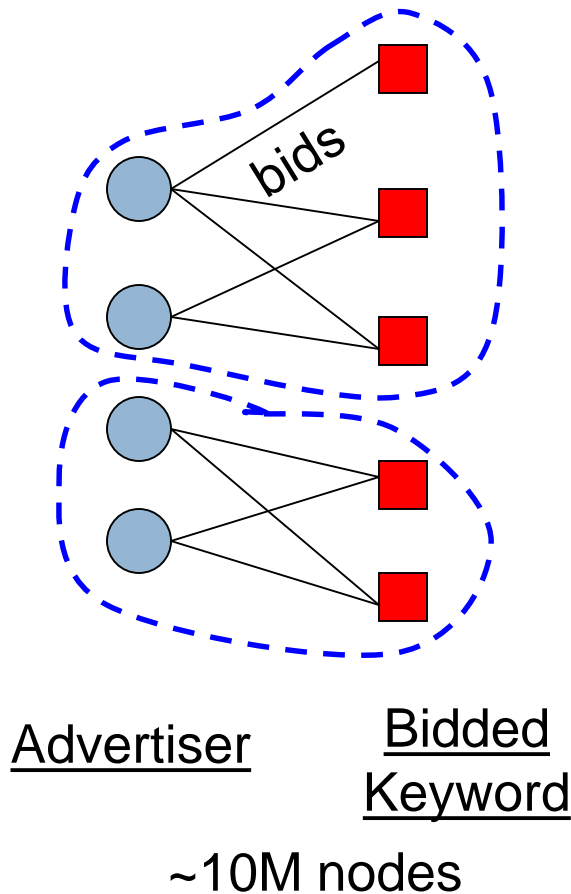
Here are all the Xbox One voice commands

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.

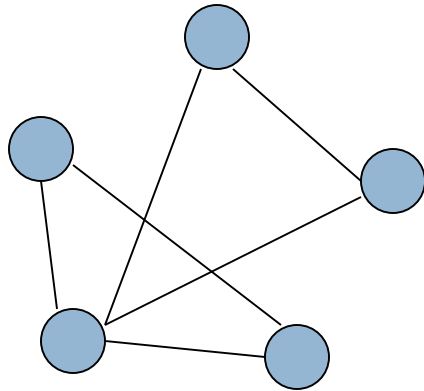
Clustering in search advertising



Find clusters of advertisers and keywords

- ▣ Keyword suggestion
- ▣ Performance estimation

Clustering applications



Who-messages-who
IM/text/twitter graph

~100M nodes

Find clusters of users

- ▣ Targeted advertising
- ▣ Exploratory analysis

Clusters of the Web Graph

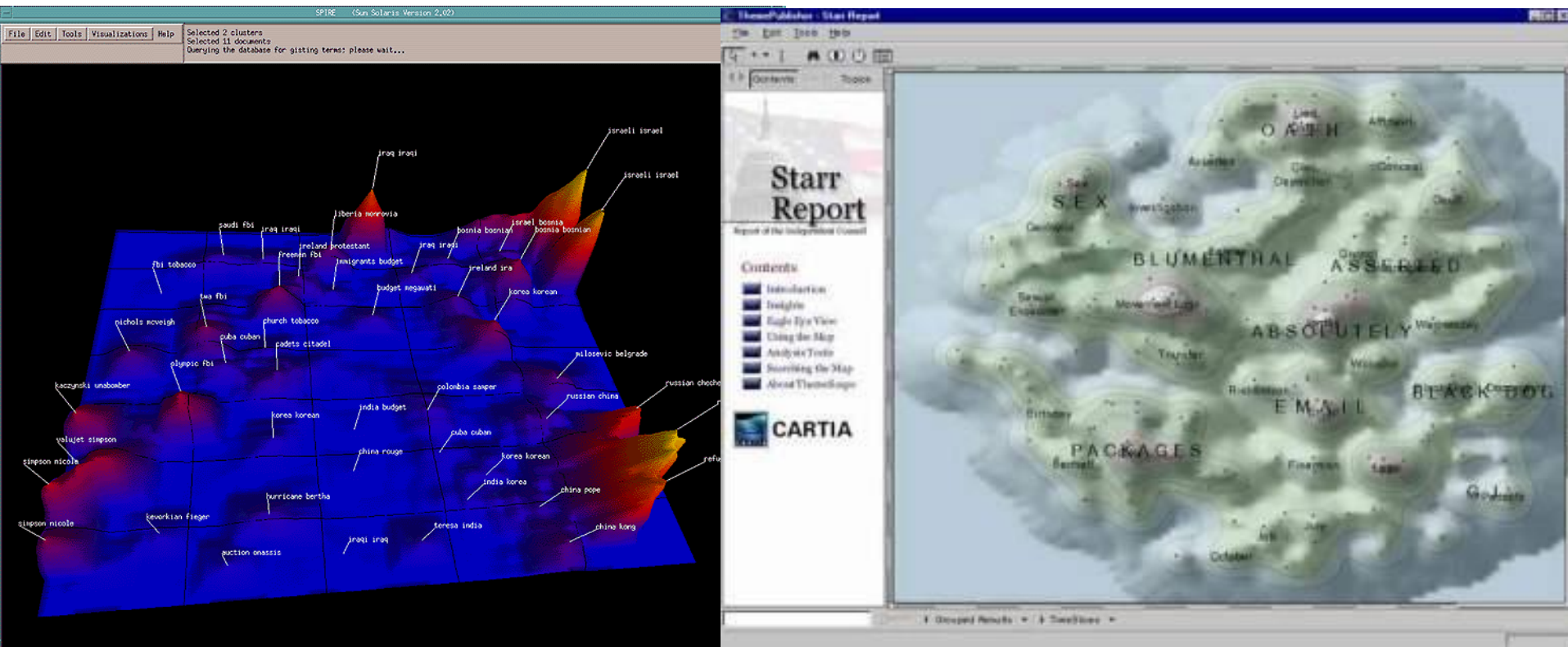
- ▣ Distributed pagerank computation

Data visualization

Wise et al, “Visualizing the non-visual” PNNL

ThemeScapes, Cartia

- [Mountain height = cluster size]



Issues for clustering

Representation for clustering

- ▣ How do we represent an example
 - features, etc.
- ▣ Similarity/distance between examples

Flat clustering or hierarchical

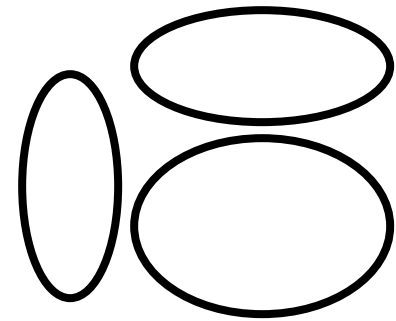
Number of clusters

- ▣ Fixed a priori
- ▣ Data driven?

Clustering Algorithms

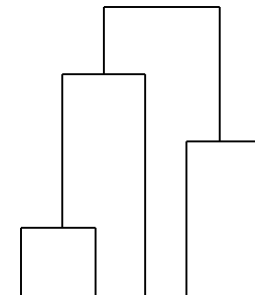
Flat algorithms

- ▣ 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 (probabilistic)

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

K-means



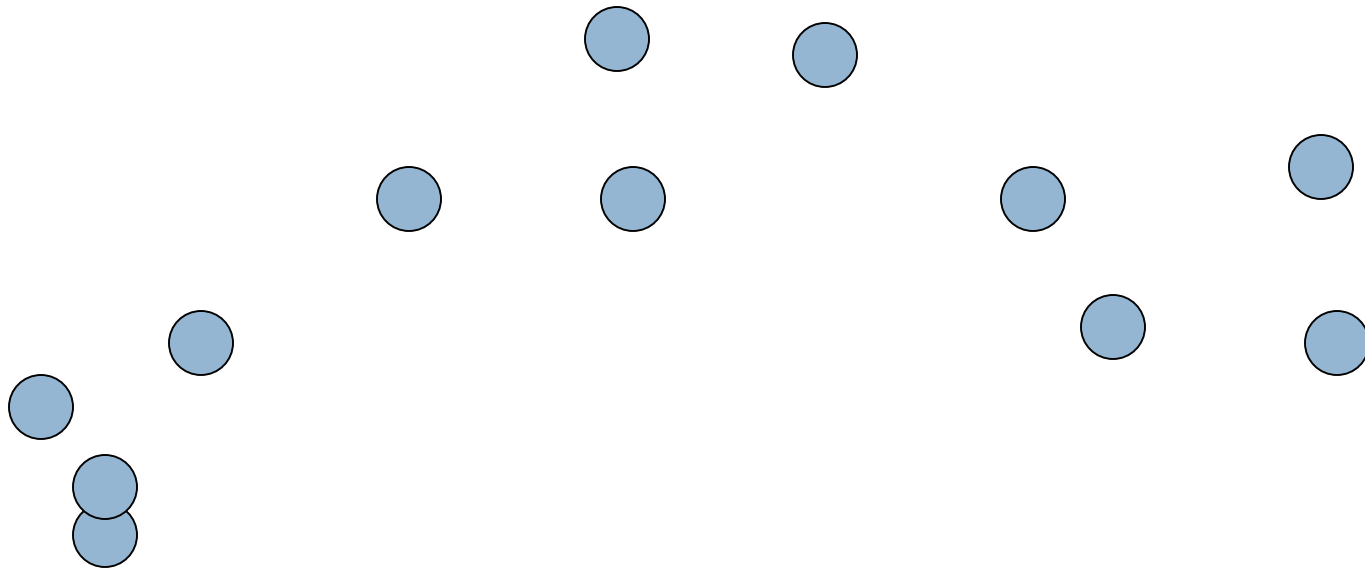
Most well-known and popular clustering algorithm:

Start with some initial cluster centers

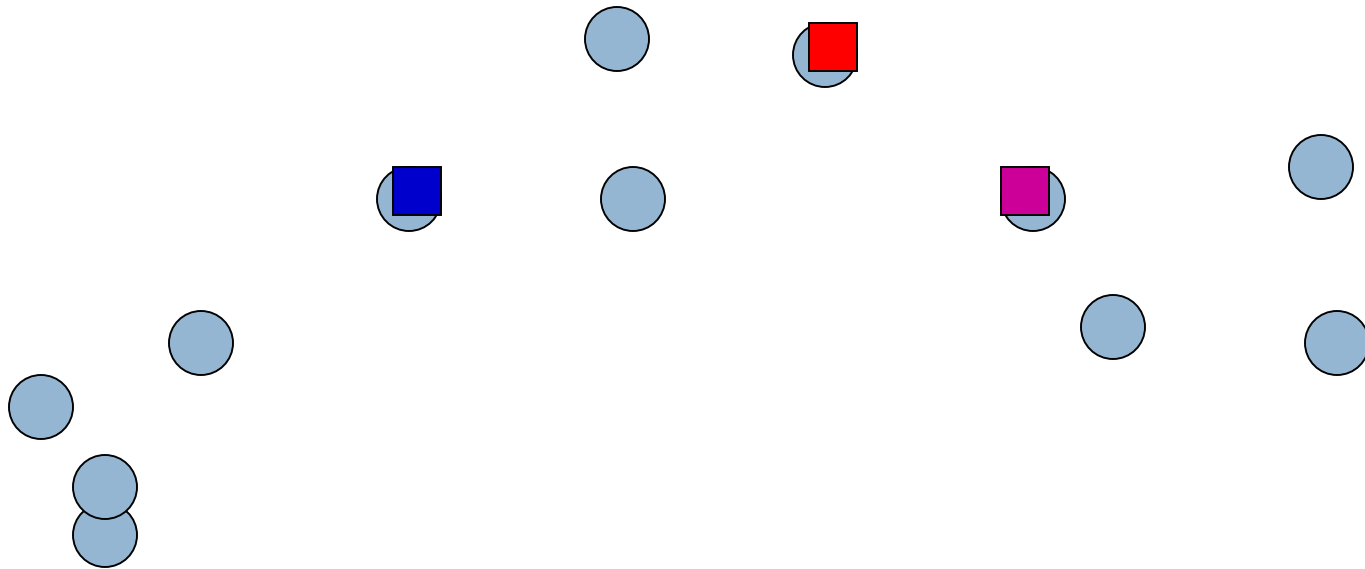
Iterate:

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

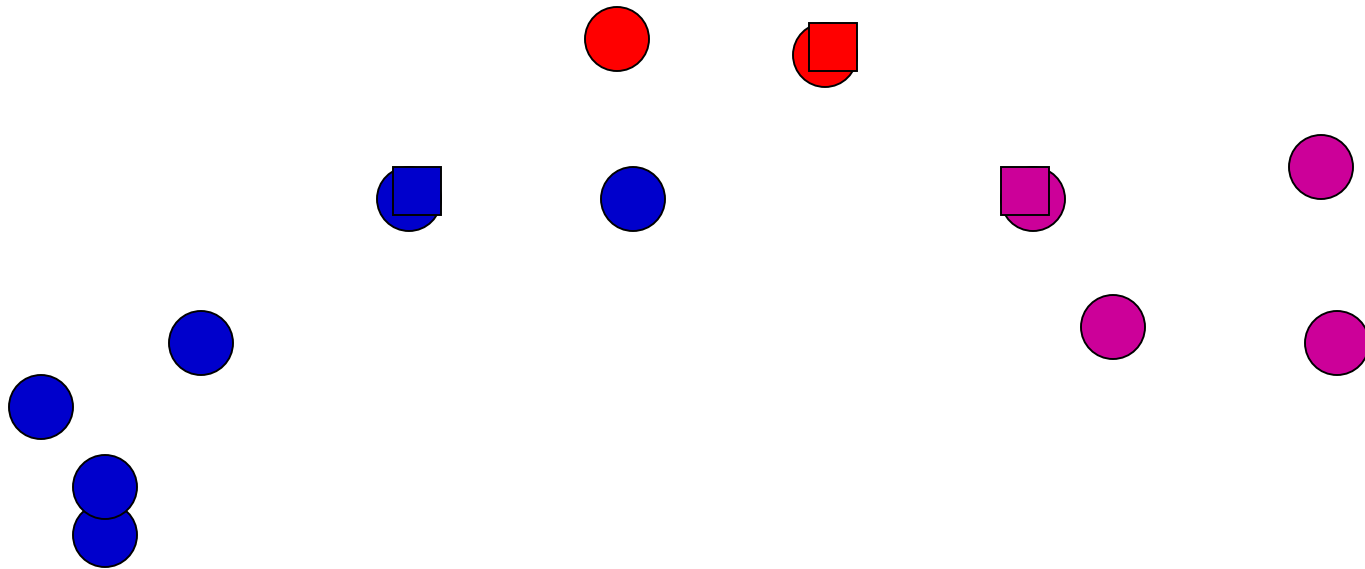
K-means: an example



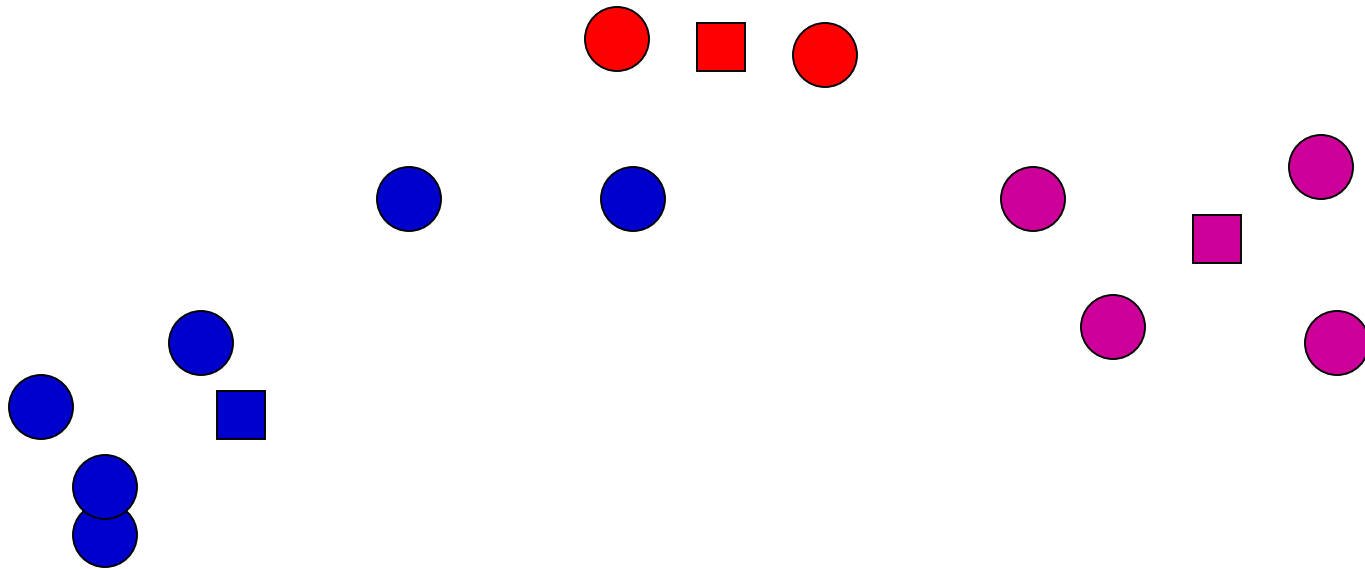
K-means: Initialize centers randomly



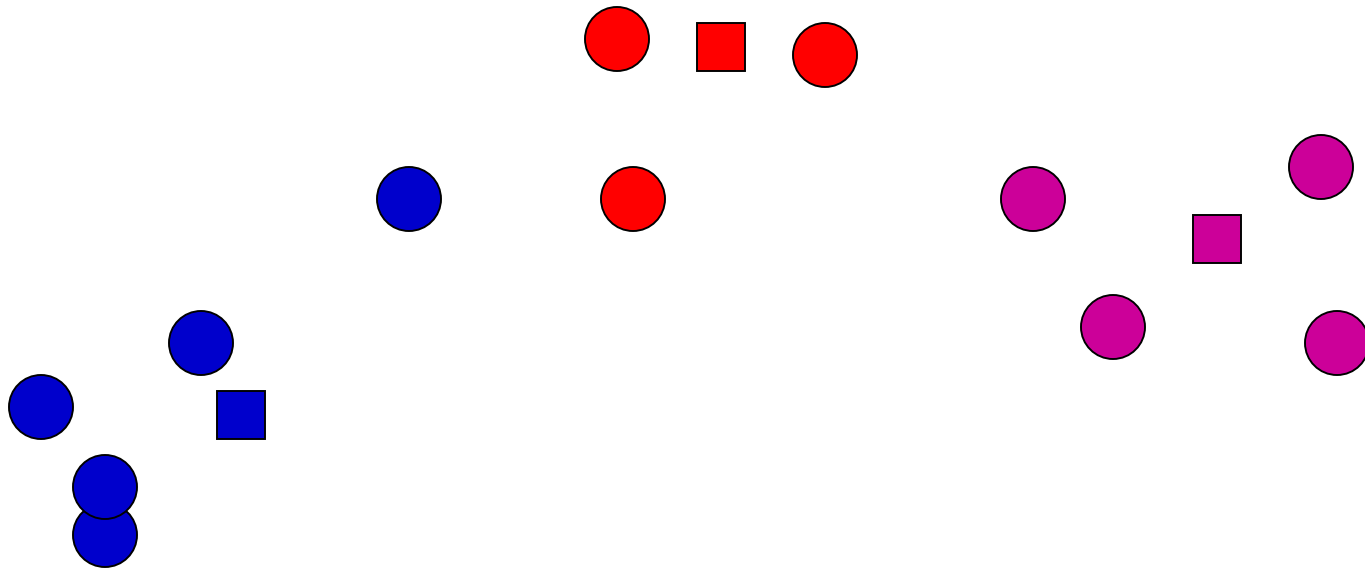
K-means: assign points to nearest center



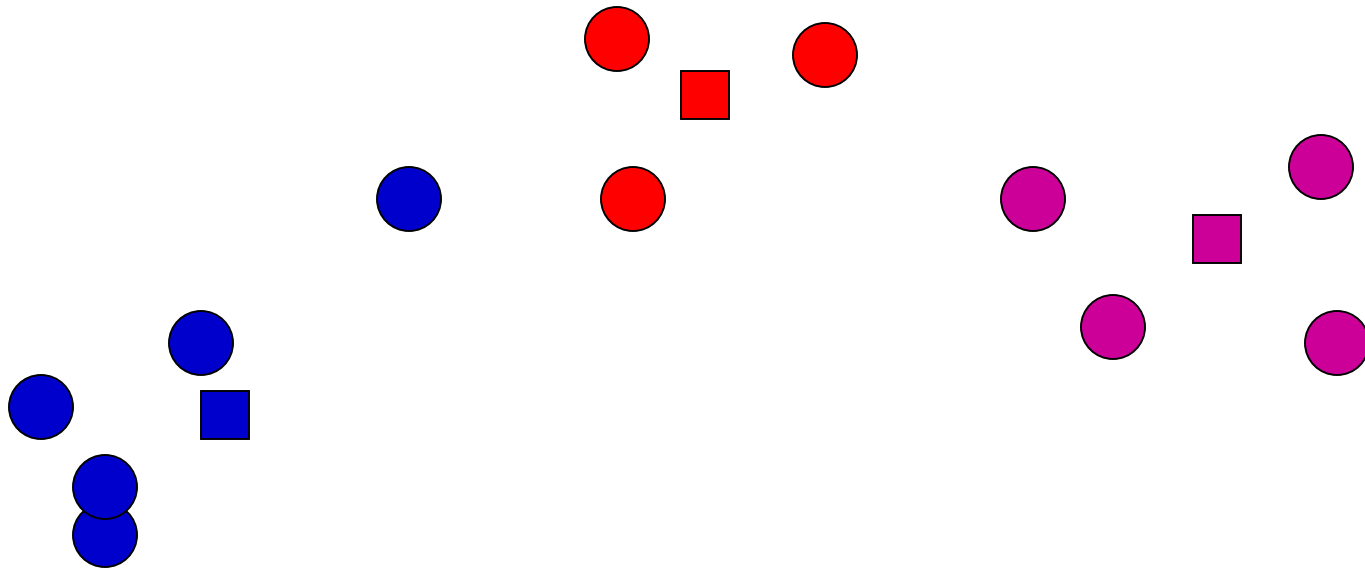
K-means: readjust centers



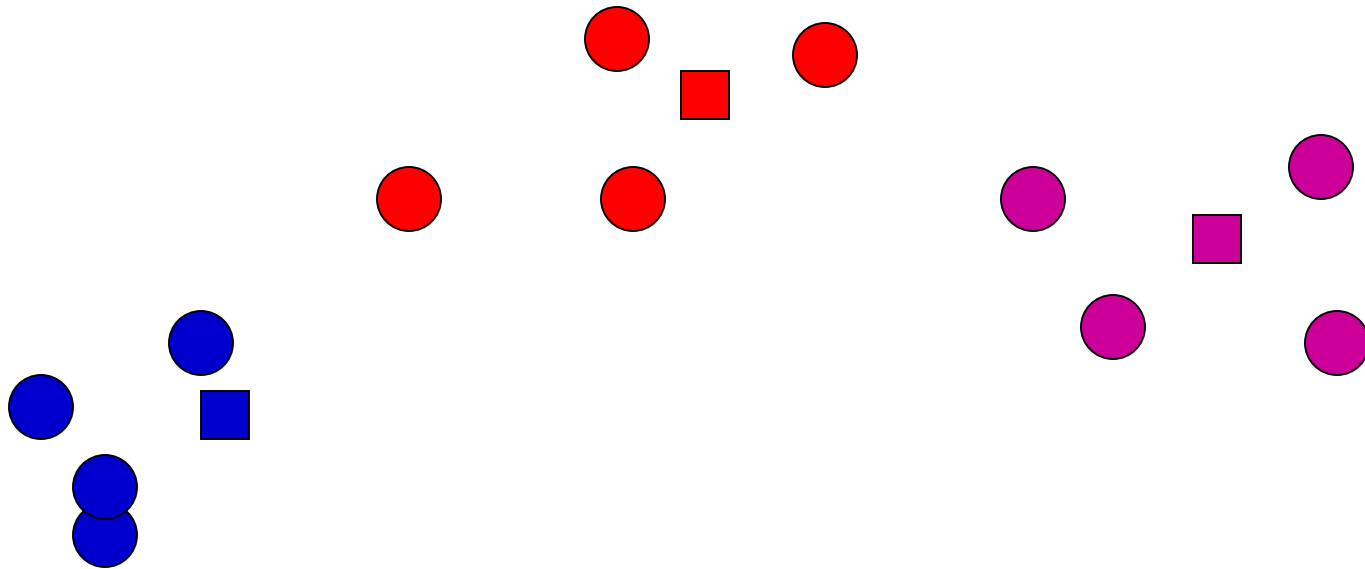
K-means: assign points to nearest center



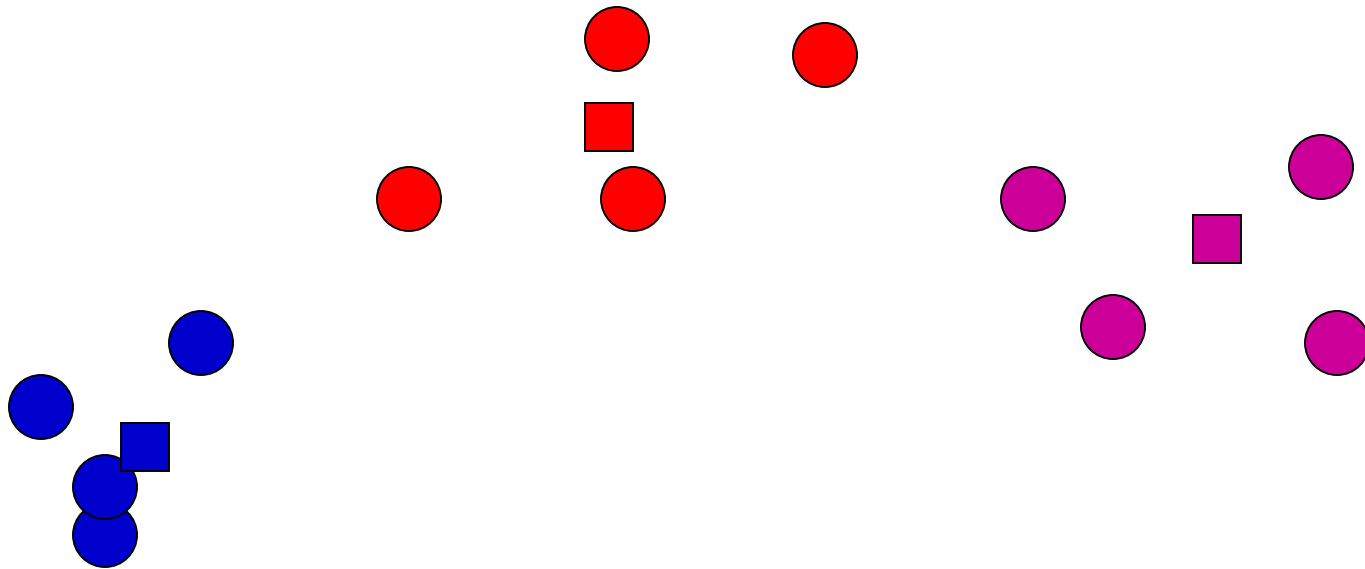
K-means: readjust centers



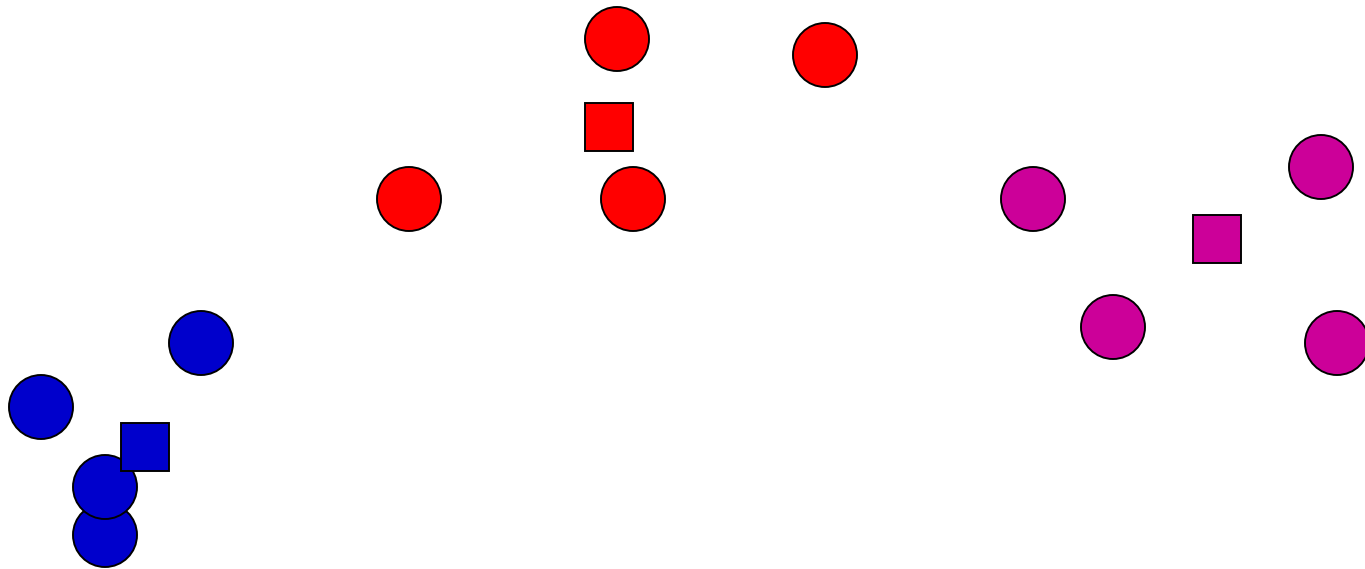
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center

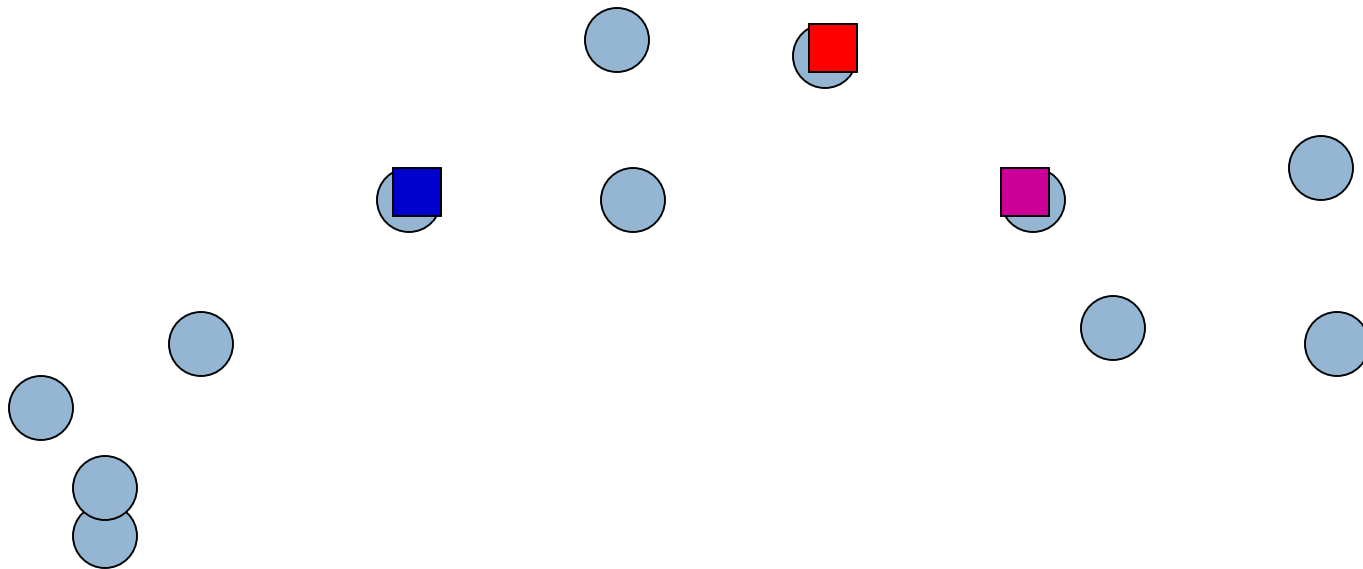


No changes: Done

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

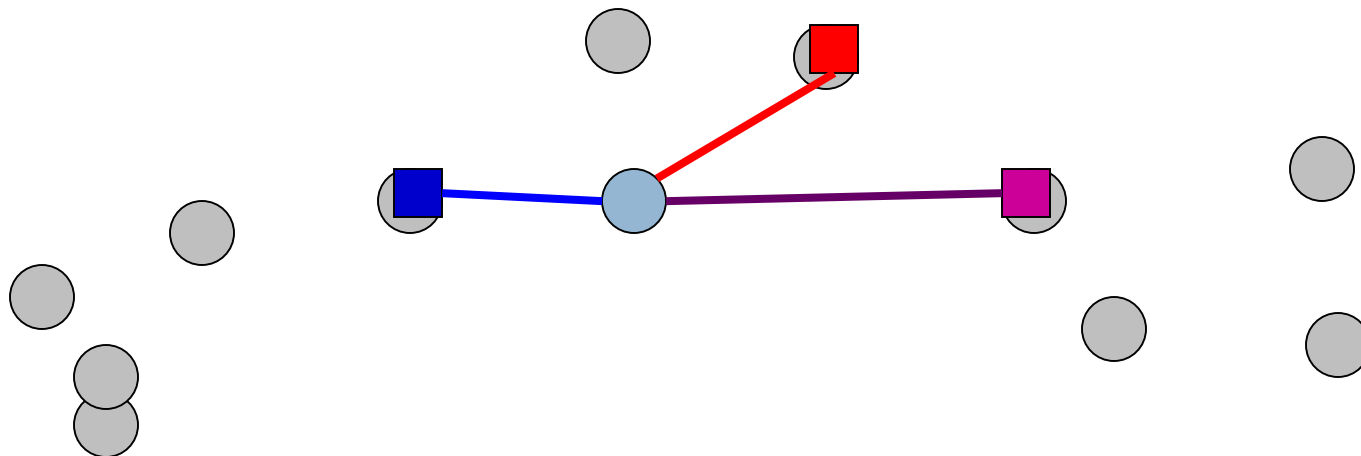
Iterate:

- **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

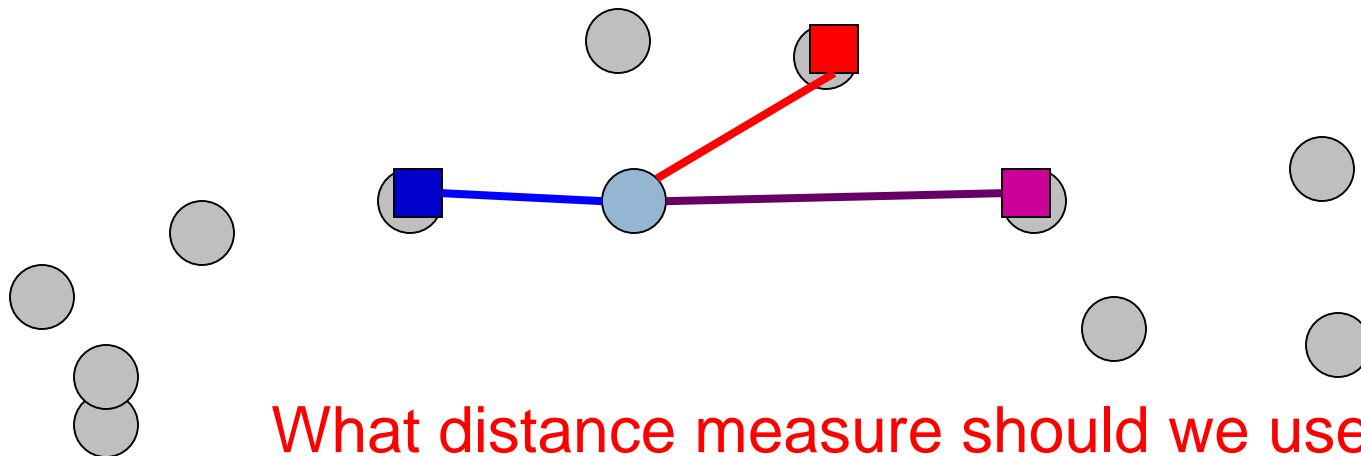
Iterate:

- **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



Distance measures

Euclidean:

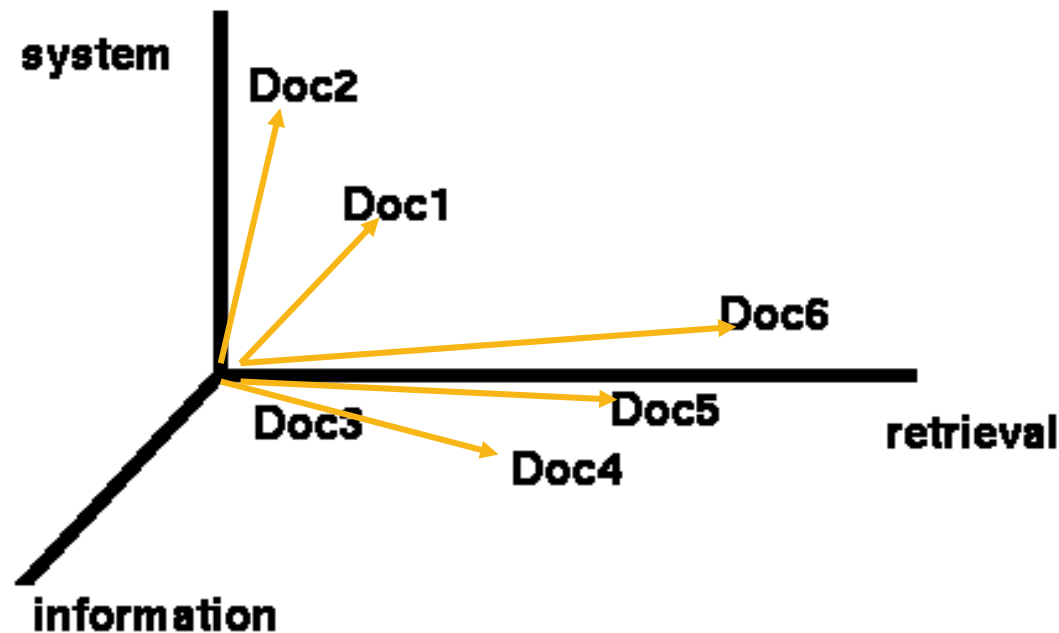
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

good for spatial data

Clustering documents (e.g. news data)

One feature for each word. The value is the number of times that word occurs.

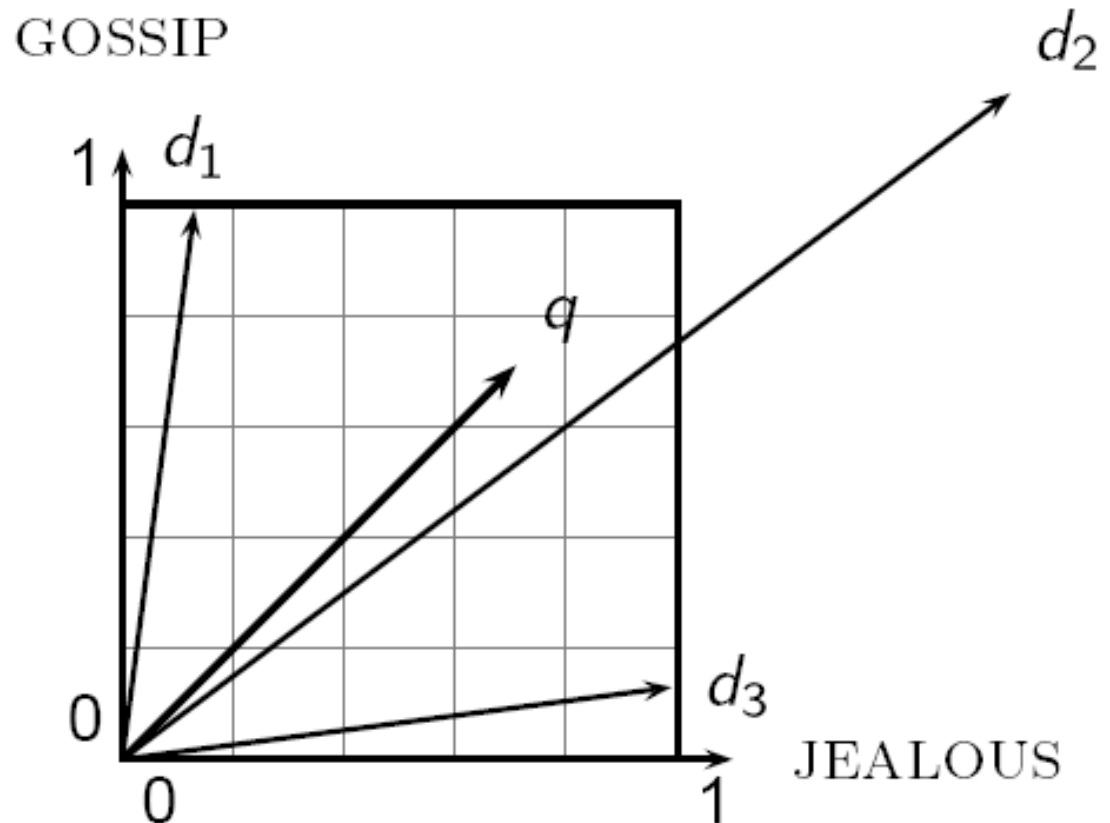
Documents are points or vectors in this space



When Euclidean distance doesn't work

Which document is closest to q using Euclidean distance?

Which do you think should be closer?

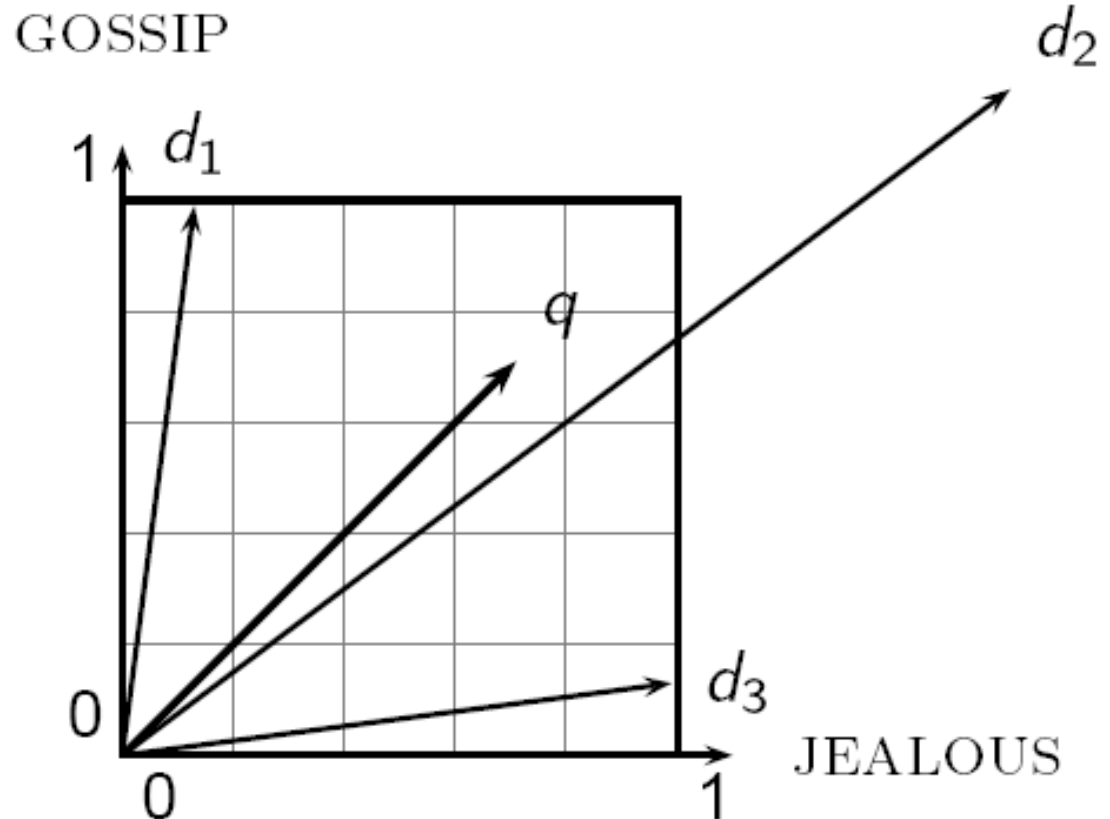


Issues with Euclidian distance

the Euclidean distance between q and d_2 is large

but, the distribution of terms in the query q and the distribution of terms in the document d_2 are very similar

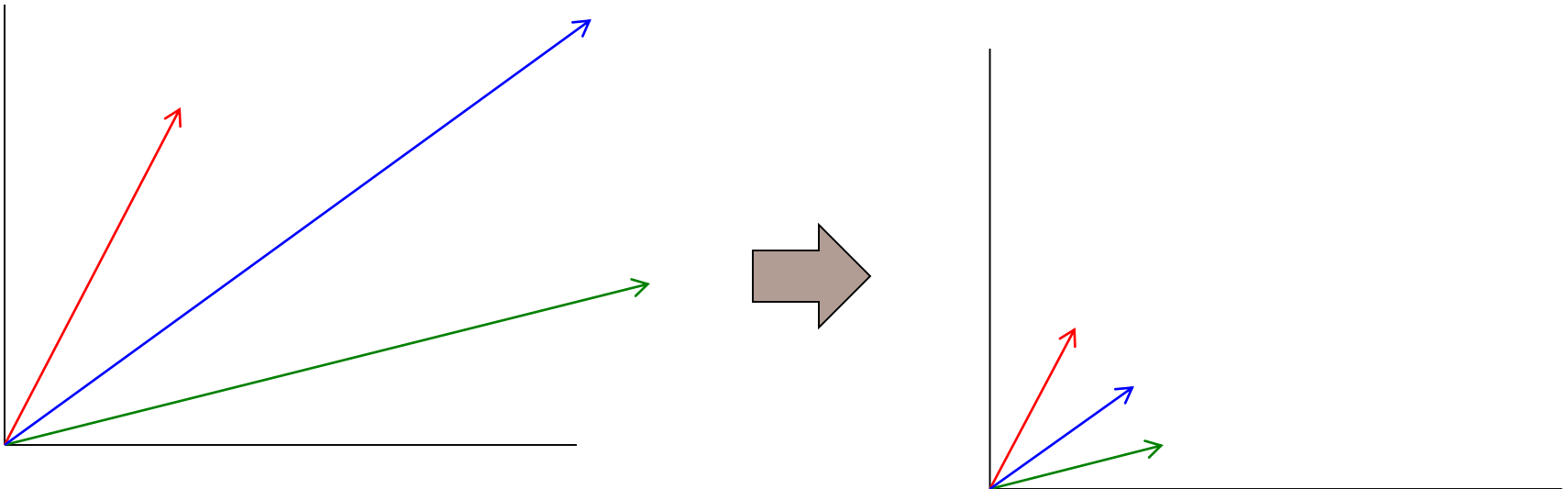
This is not what we want!



cosine similarity

$$\text{sim}(x, y) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{x}{\|x\|} \cdot \frac{y}{\|y\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

correlated with the
angle between two vectors



cosine distance

cosine similarity is a similarity between 0 and 1, with things that are similar 1 and not 0

We want a distance measure, cosine distance:

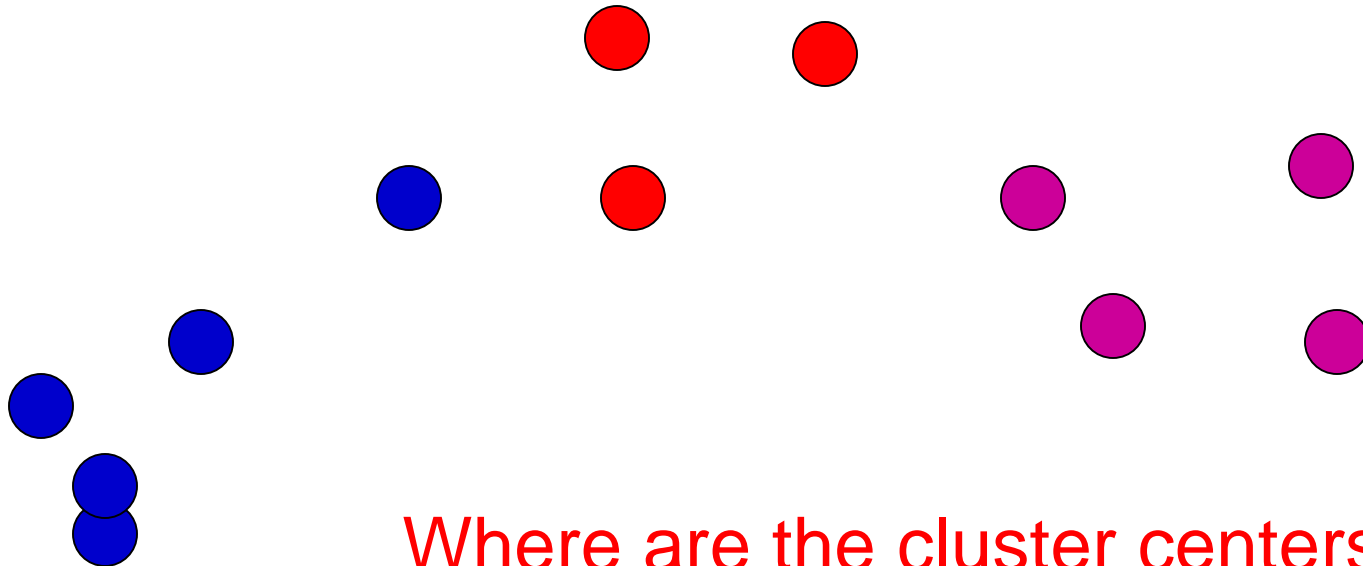
$$d(x, y) = 1 - \text{sim}(x, y)$$

- good for text data and many other “real world” data sets
- is computationally friendly since we only need to consider features that have non-zero values **both** examples

K-means

Iterate:

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

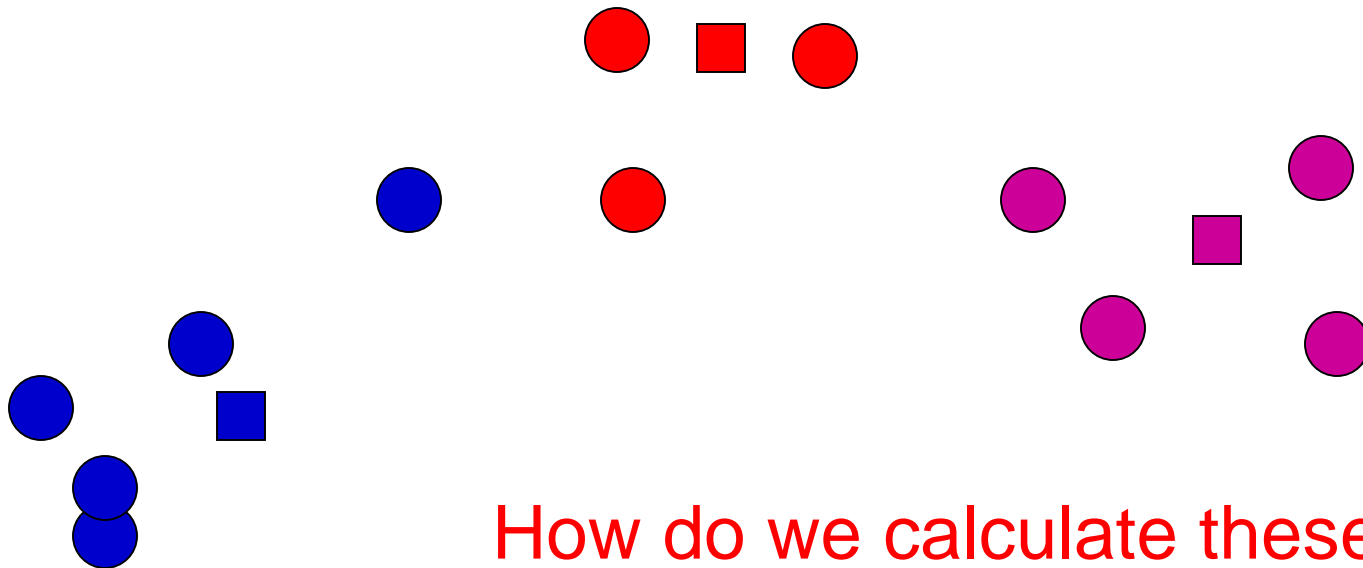


Where are the cluster centers?

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:

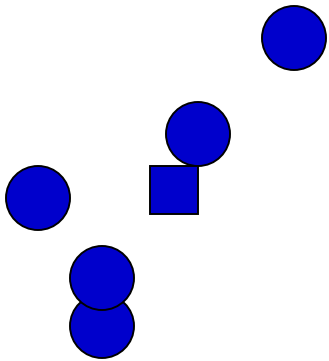
- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Mean of the points in the cluster:

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

where:

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



K-means loss function

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

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

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
-

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

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
-

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

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
-

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

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
-

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

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

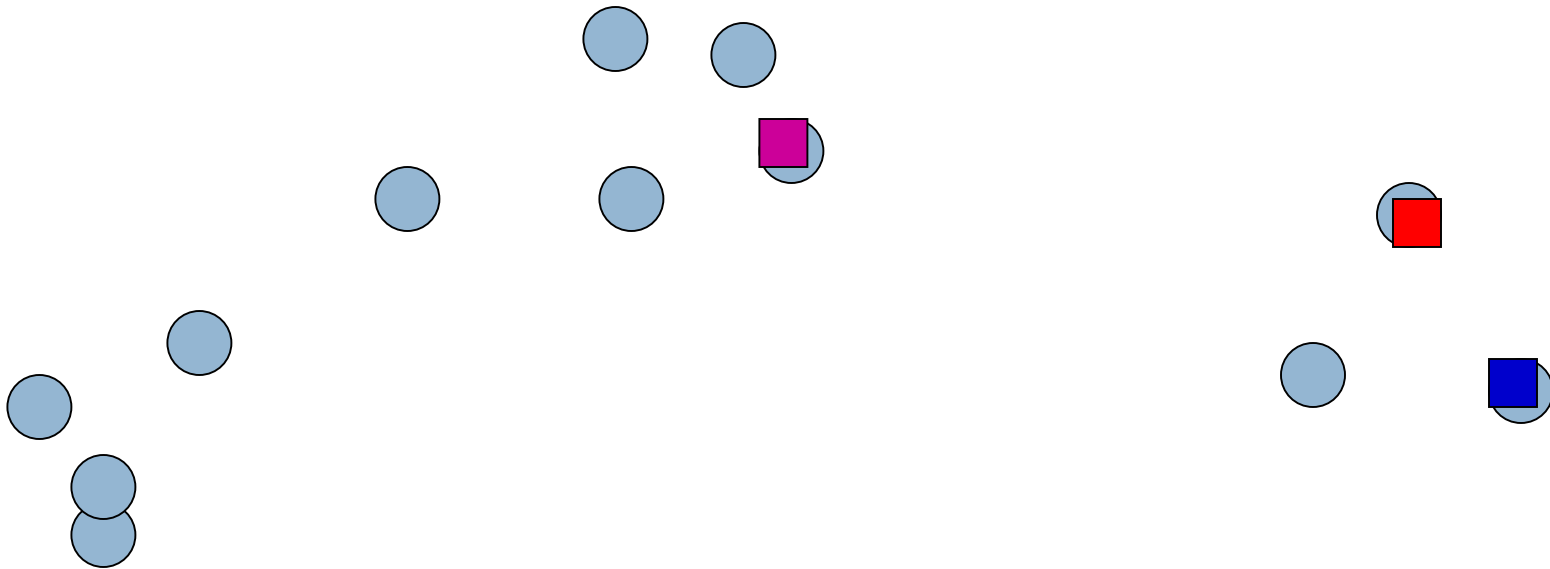
Initial (seed) cluster centers

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 :

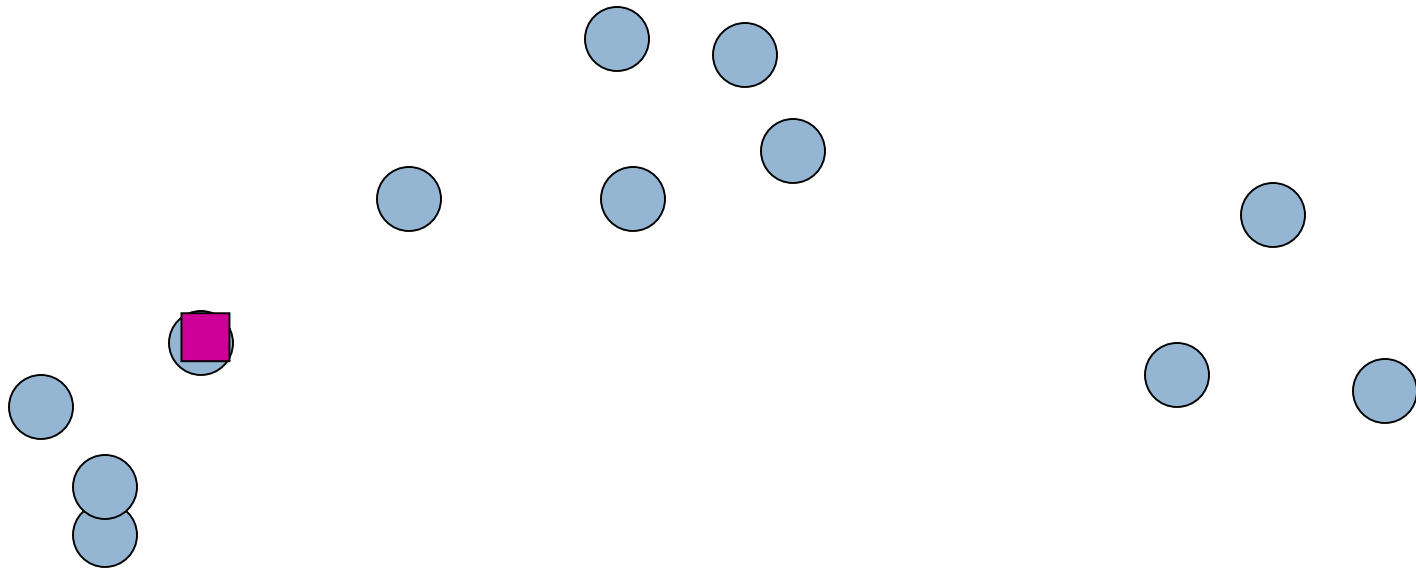
μ_i = point that is furthest from **any** previous centers

$$m_i = \underbrace{\arg \max_x}_{\text{point with the largest distance to any previous center}} \underbrace{\min_{m_j : 1 < j < i} d(x, m_j)}_{\text{smallest distance from } x \text{ to any previous center}}$$

point with the largest
distance to any previous
center

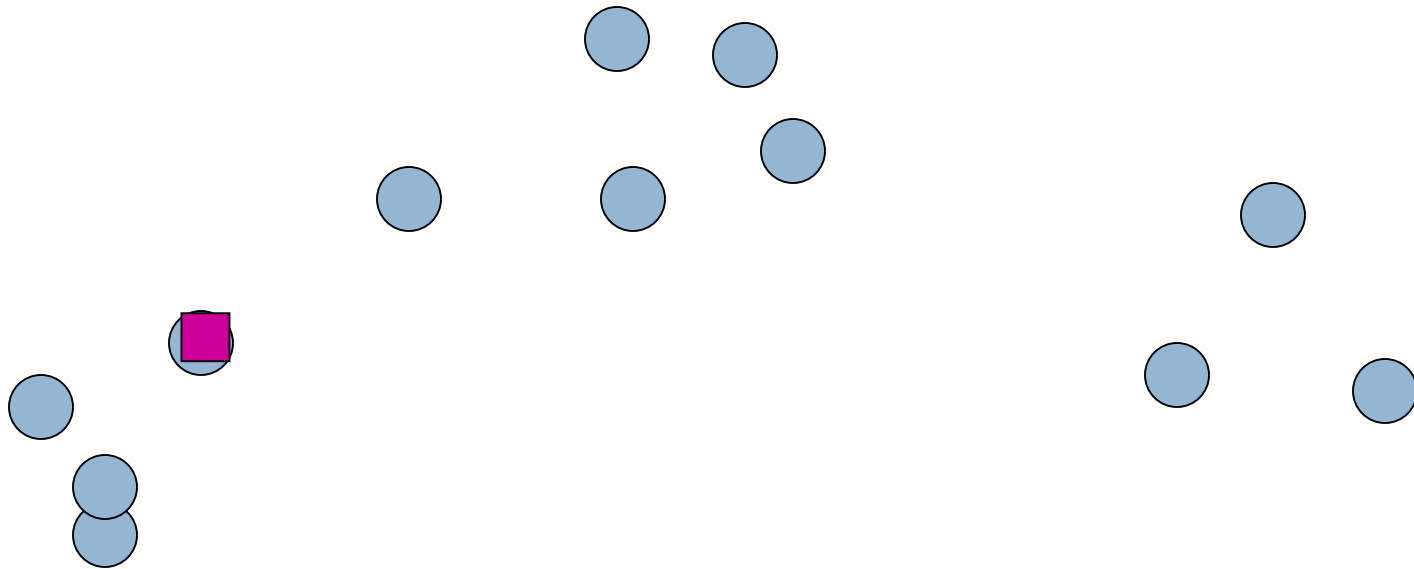
smallest distance from x to
any previous center

K-means: Initialize furthest from centers



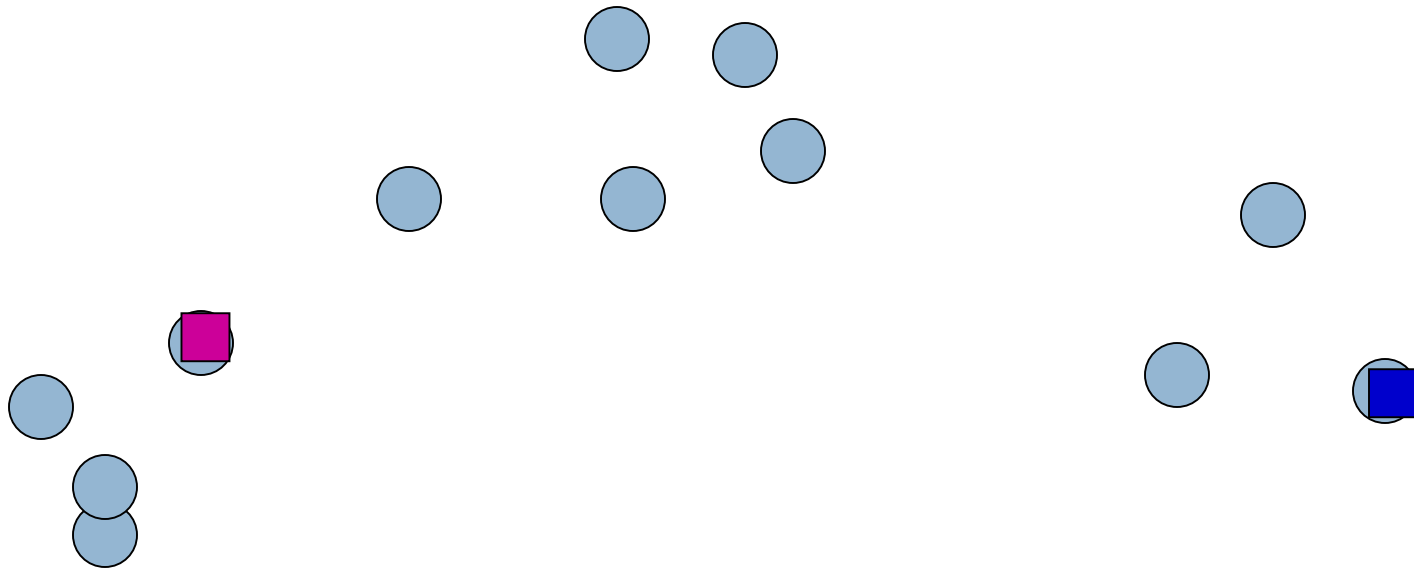
Pick a random point for the first center

K-means: Initialize furthest from centers



What point will be chosen next?

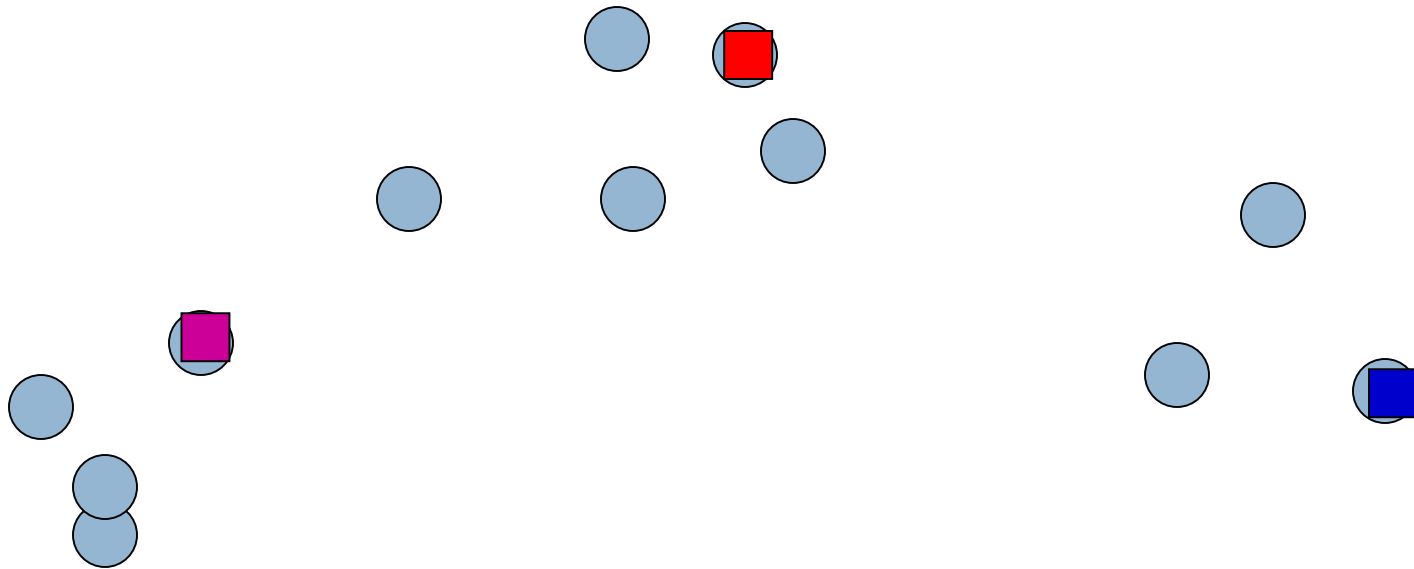
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

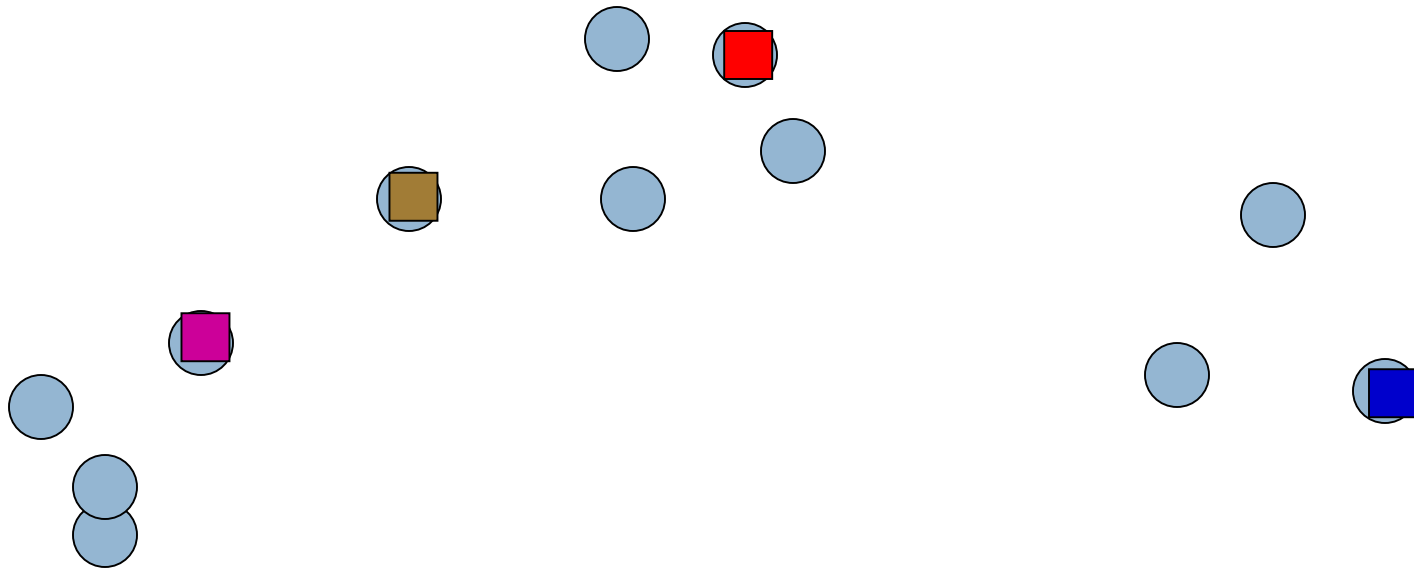
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

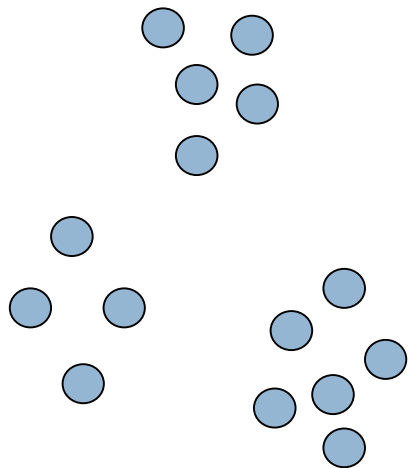
K-means: Initialize furthest from centers



Furthest point from center

Any issues/concerns with this approach?

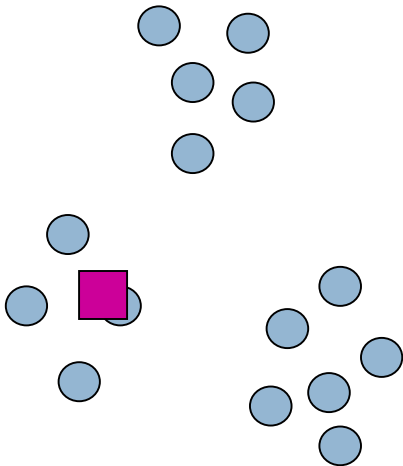
Furthest points concerns



If $k = 4$, which points will get chosen?



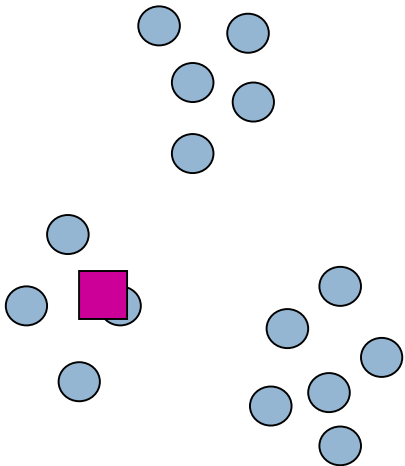
Furthest points concerns



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**:

$s_i = \min d(x_i, \mu_{1\dots k-1})$ // 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**:

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

μ_k = randomly pick point *proportionate* to **s**

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

Scalable K-Means++

- Scalable K-Means++ > <http://web.stanford.edu/group/mmds/slides2012/s-bahmani.pdf>
- K-Means Data Clustering Using C# > <https://visualstudiomagazine.com/Articles/2013/12/01/K-Means-Data-Clustering-Using-C.aspx?Page=1>
- Test Run - K-Means++ Data Clustering > <https://msdn.microsoft.com/en-us/magazine/mt185575.aspx>

K-means variations/parameters

~~Initial (seed) cluster centers~~

~~Convergence~~

- ~~■ A fixed number of iterations~~
- ~~■ partitions unchanged~~
- ~~■ Cluster centers don't change~~

K!

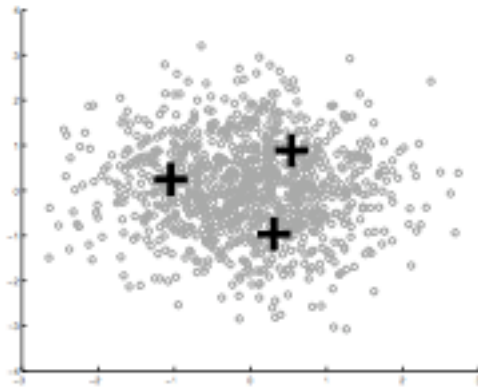
How Many Clusters?

Number of clusters K must be provided

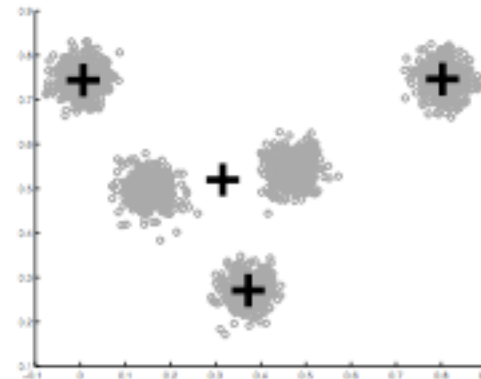
How should we determine the number of clusters?

How did we deal with models becoming too complicated previously?

too many



too few



Many approaches

Regularization!!!



Statistical test

k-means loss revisited

K-means is trying to minimize:

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

What happens when k increases?

k-means loss revisited

K-means is trying to minimize:

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

Loss goes down!

Making the model more complicated allows us more flexibility, but can “overfit” to the data

k-means loss revisited

K-means is trying to minimize:

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



2 regularization options

$$loss_{BIC} = loss_{kmeans} + K \log N \quad (\text{where } N = \text{number of points})$$

$$loss_{AIC} = loss_{kmeans} + KN$$

What effect will this have?

Which will tend to produce smaller k?

k-means loss revisited

2 regularization options

$$loss_{BIC} = loss_{kmeans} + K \log N \quad (\text{where } N = \text{number of points})$$

$$loss_{AIC} = loss_{kmeans} + KN$$

AIC penalizes increases in K more harshly

Both require a change to the K-means algorithm

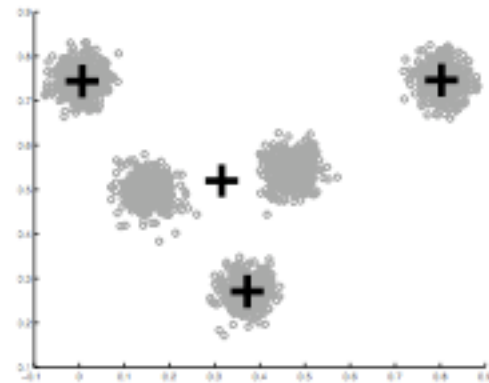
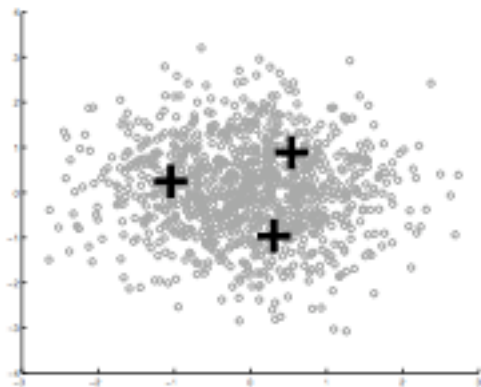
Tend to work reasonably well in practice if you don't know K

Statistical approach

Assume data is Gaussian (i.e. spherical)

Test for this

- ▣ Testing in high dimensions doesn't work well
- ▣ Testing in lower dimensions does work well

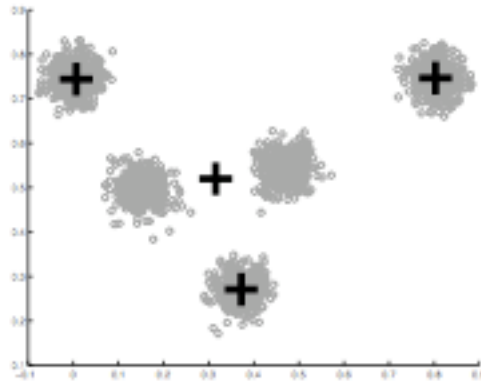


ideas?

Project to one dimension and check

For each cluster, project down to one dimension

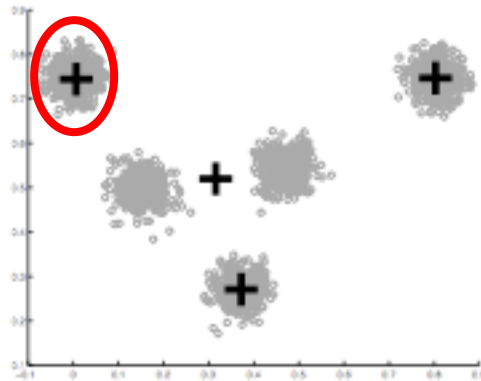
- ▣ Use a statistical test to see if the data is Gaussian



Project to one dimension and check

For each cluster, project down to one dimension

- ▣ Use a statistical test to see if the data is Gaussian

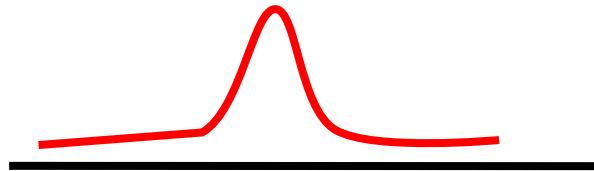
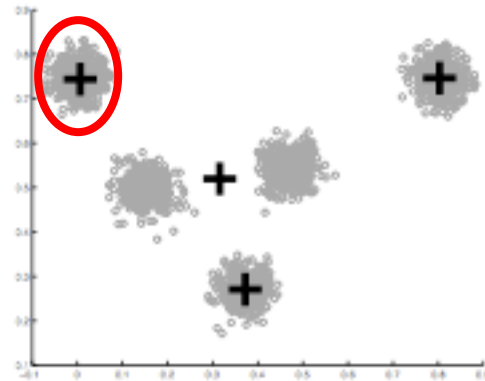


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

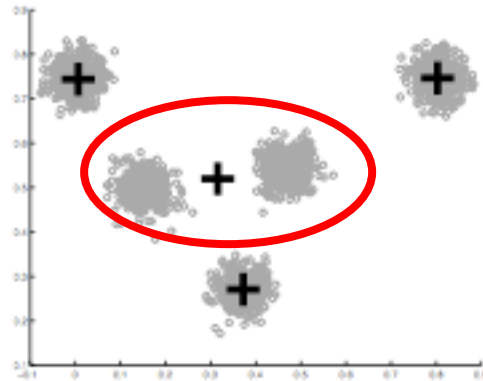
- ▣ Use a statistical test to see if the data is Gaussian



Project to one dimension and check

For each cluster, project down to one dimension

- ▣ Use a statistical test to see if the data is Gaussian

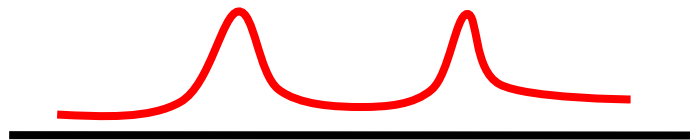
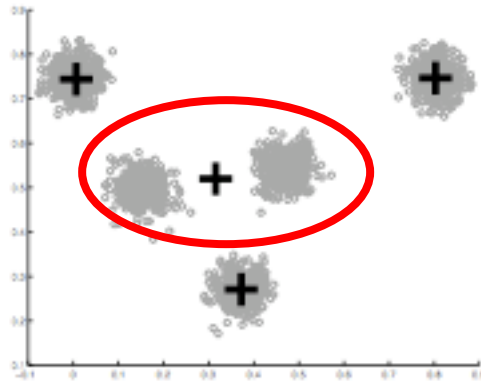


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

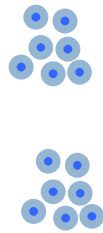
- ▣ Use a statistical test to see if the data is Gaussian



Project to one dimension and check

For each cluster, project down to one dimension

- ▣ Use a statistical test to see if the data is Gaussian

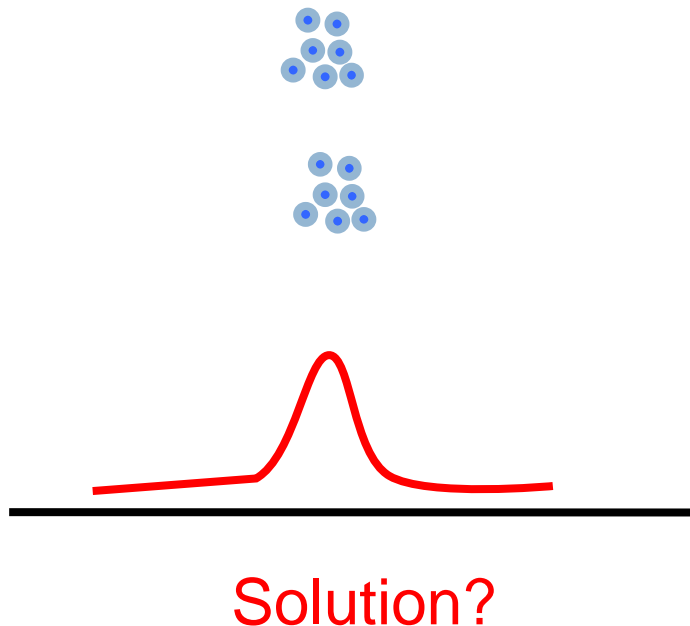


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

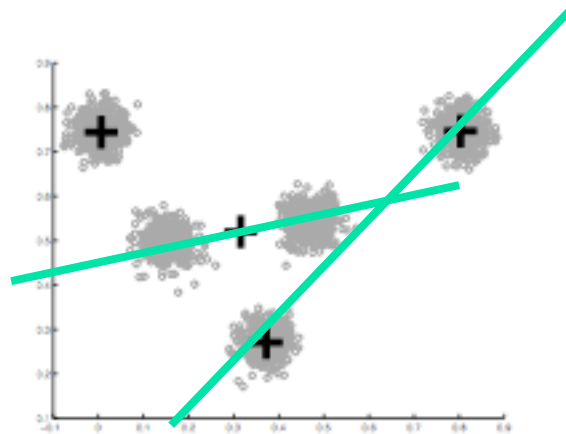
- ▣ Use a statistical test to see if the data is Gaussian



Project to one dimension and check

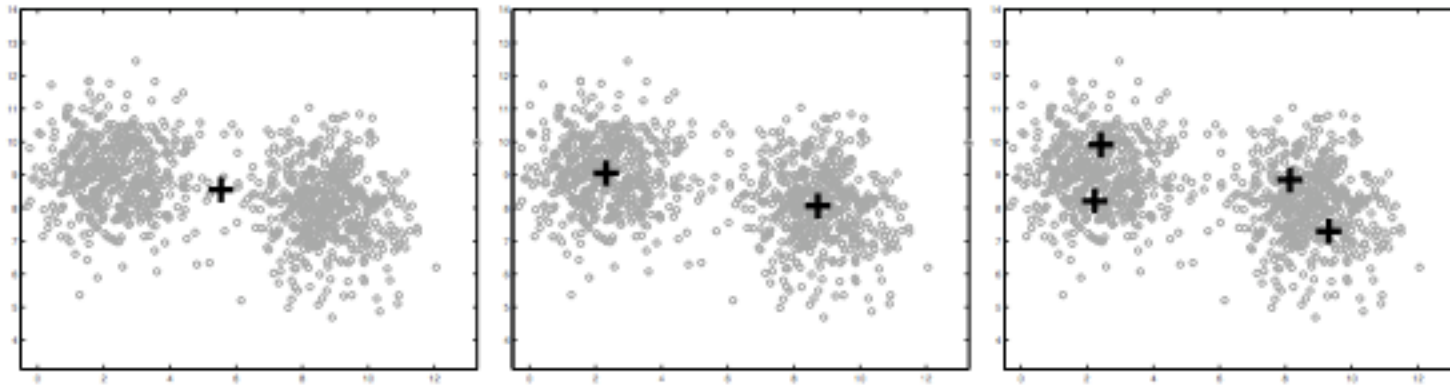
For each cluster, project down to one dimension

- ▣ Use a statistical test to see if the data is Gaussian



Chose the dimension of the projection as the dimension with highest variance

On synthetic data



Split too far

Compared to other approaches

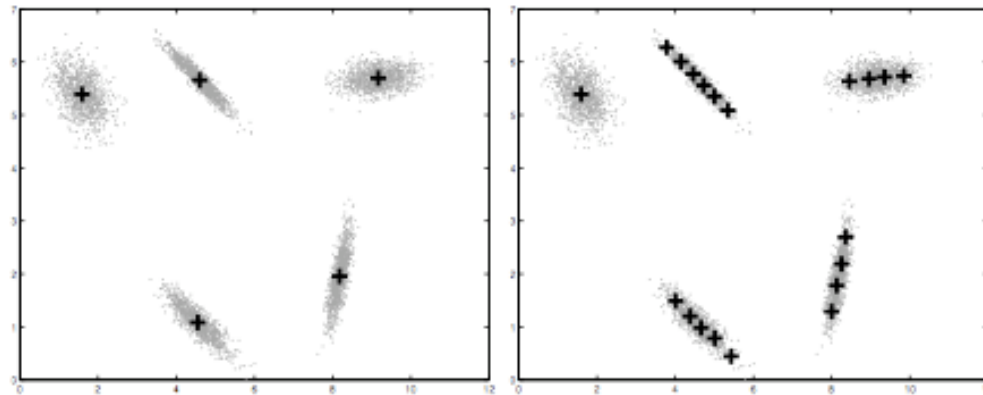


Figure 4: 2- d synthetic dataset with 5 true clusters. On the left, G-means correctly chooses 5 centers and deals well with non-spherical data. On the right, the BIC causes X -means to overfit the data, choosing 20 unevenly distributed clusters.

http://cs.baylor.edu/~hamerly/papers/nips_03.pdf

K-Means time complexity

Variables: K clusters, n data points,
 m features/dimensions, I iterations

What is the runtime complexity?

- ▣ Computing distance between two points (e.g. euclidean)
- ▣ Reassigning clusters
- ▣ Computing new centers
- ▣ Iterate...

K-Means time complexity

Variables: K clusters, n data points,
 m features/dimensions, l iterations

What is the runtime complexity?

- ▣ Computing distance between two points is $O(m)$ where m is the dimensionality of the vectors/number of features.
- ▣ Reassigning clusters: $O(Kn)$ distance computations, or $O(Knm)$
- ▣ Computing centroids: Each point gets added once to some centroid: $O(nm)$
- ▣ Assume these two steps are each done once for l iterations: $O(lknm)$

In practice, K-means converges quickly and is fairly fast

What Is A Good Clustering?

Internal criterion: A good clustering will produce high quality clusters in which:

- ▣ the intra-class (that is, intra-cluster) similarity is high
- ▣ the inter-class similarity is low

How would you evaluate clustering?

Common approach: use labeled data

Use data with known classes

- ▣ For example, document classification data

data	label
	
	
	
	
	



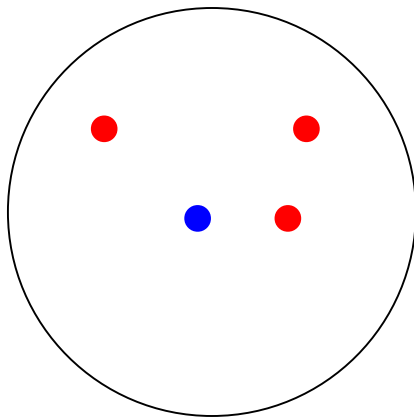
If we clustered this data (ignoring labels) what would we like to see?

Reproduces class partitions

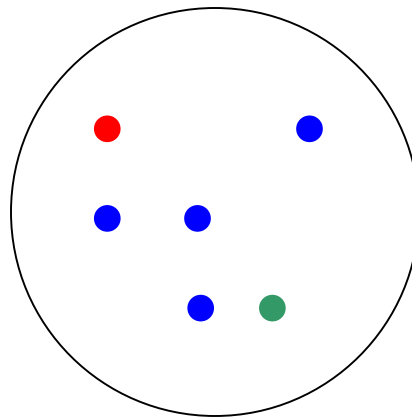
How can we quantify this?

Common approach: use labeled data

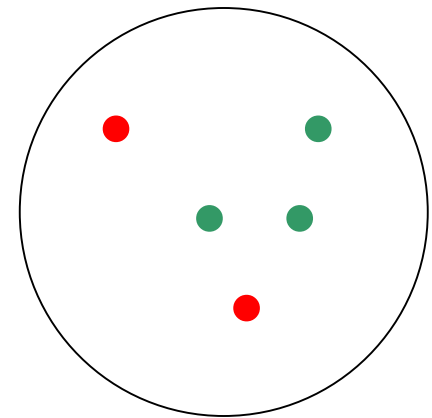
Purity, the proportion of the dominant class in the cluster



Cluster I



Cluster II



Cluster III

Cluster I: Purity = $1/4 (\max(3, 1, 0)) = 3/4$

Cluster II: Purity = $1/6 (\max(1, 4, 1)) = 4/6$

Cluster III: Purity = $1/5 (\max(2, 0, 3)) = 3/5$

Overall purity?

Overall purity

Cluster I: Purity = $1/4 (\max(3, 1, 0)) = 3/4$

Cluster II: Purity = $1/6 (\max(1, 4, 1)) = 4/6$

Cluster III: Purity = $1/5 (\max(2, 0, 3)) = 3/5$

Cluster average:

$$\frac{\frac{3}{4} + \frac{4}{6} + \frac{3}{5}}{3} = 0.672$$

Weighted average:

$$\frac{4 * \frac{3}{4} + 6 * \frac{4}{6} + 5 * \frac{3}{5}}{15} = \frac{3 + 4 + 3}{15} = 0.667$$

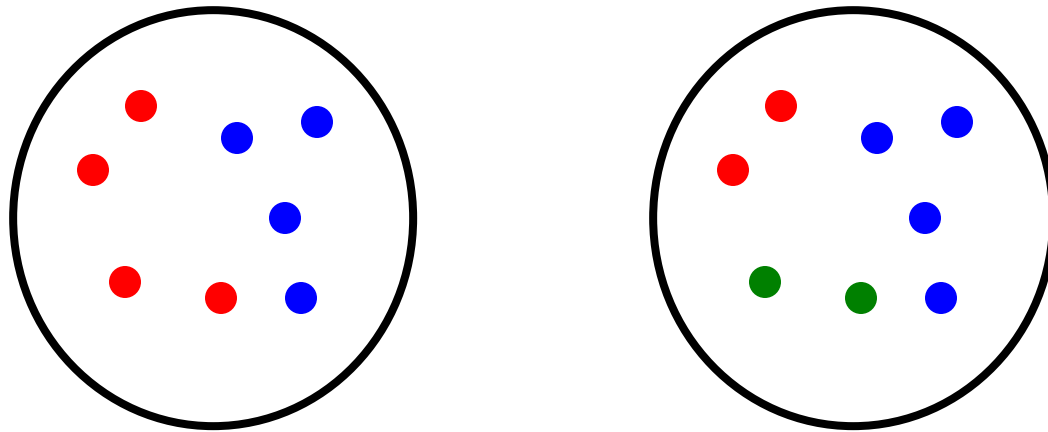
Purity issues...

Purity, the proportion of the dominant class in the cluster

Good for comparing two algorithms, but not understanding how well a single algorithm is doing, **why?**

- ▣ Increasing the number of clusters increases purity

Purity isn't perfect



Which is better based on purity?

Which do you think is better?

Ideas?

Common approach: use labeled data

Average entropy of classes in clusters

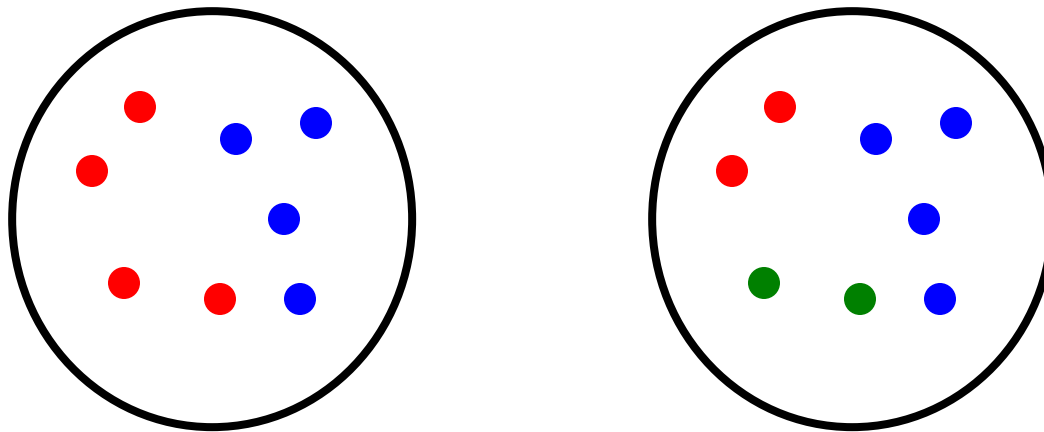
$$\text{entropy}(\text{cluster}) = - \sum_i p(\text{class}_i) \log p(\text{class}_i)$$

where $p(\text{class}_i)$ is proportion of class i in cluster

Common approach: use labeled data

Average entropy of classes in clusters

$$\text{entropy}(\text{cluster}) = - \sum_i p(\text{class}_i) \log p(\text{class}_i)$$

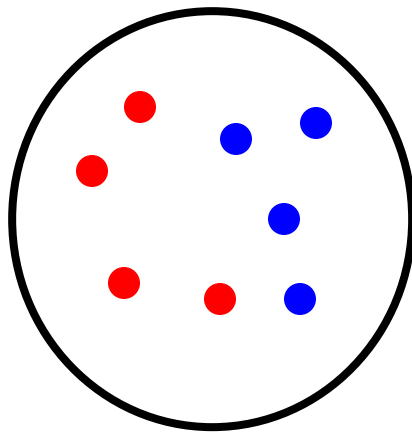


entropy?

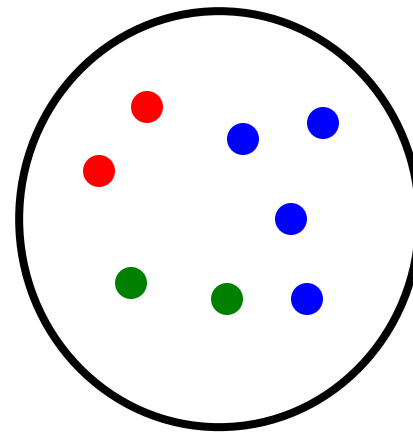
Common approach: use labeled data

Average entropy of classes in clusters

$$\text{entropy}(\text{cluster}) = - \sum_i p(\text{class}_i) \log p(\text{class}_i)$$



$$-0.5 \log 0.5 - 0.5 \log 0.5 = 1$$



$$-0.5 \log 0.5 - 0.25 \log 0.25 - 0.25 \log 0.25 = 1.5$$

Problems with K-means



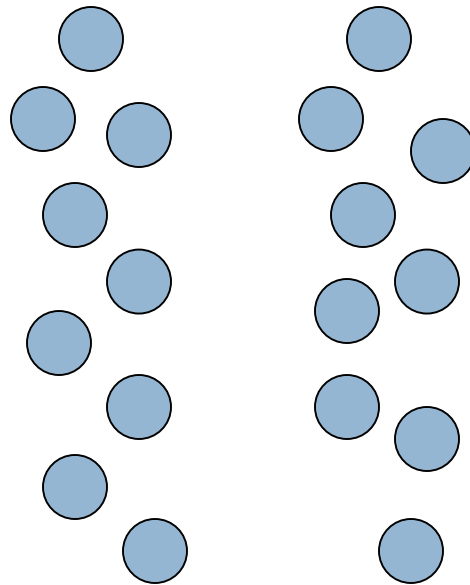
Determining K is challenging

Spherical assumption about the data (distance to cluster center)

Hard clustering isn't always right

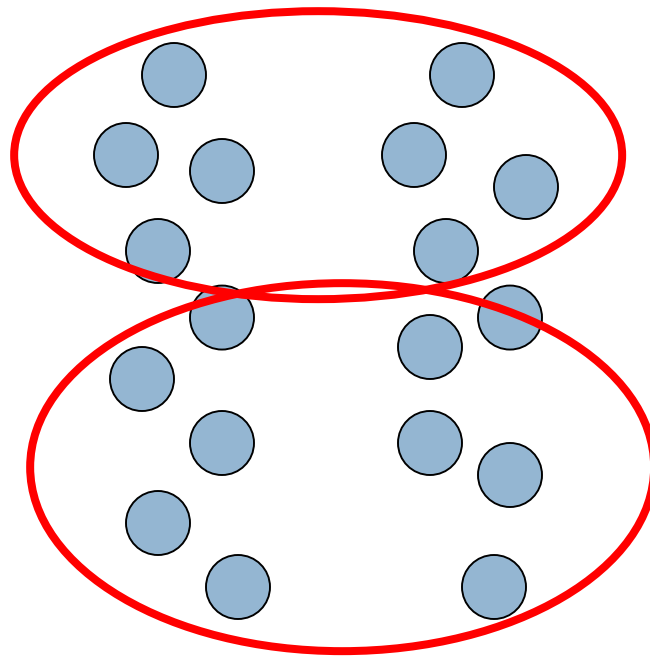
Greedy approach

Problems with K-means



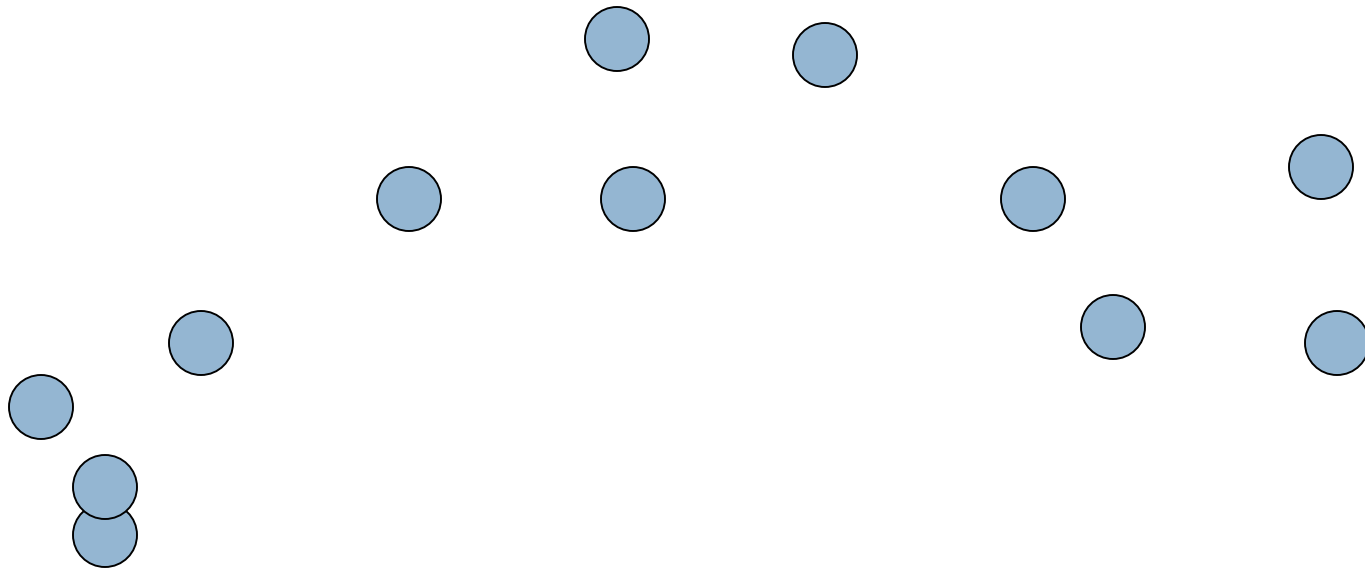
What would K-means give us here?

Assumes spherical clusters

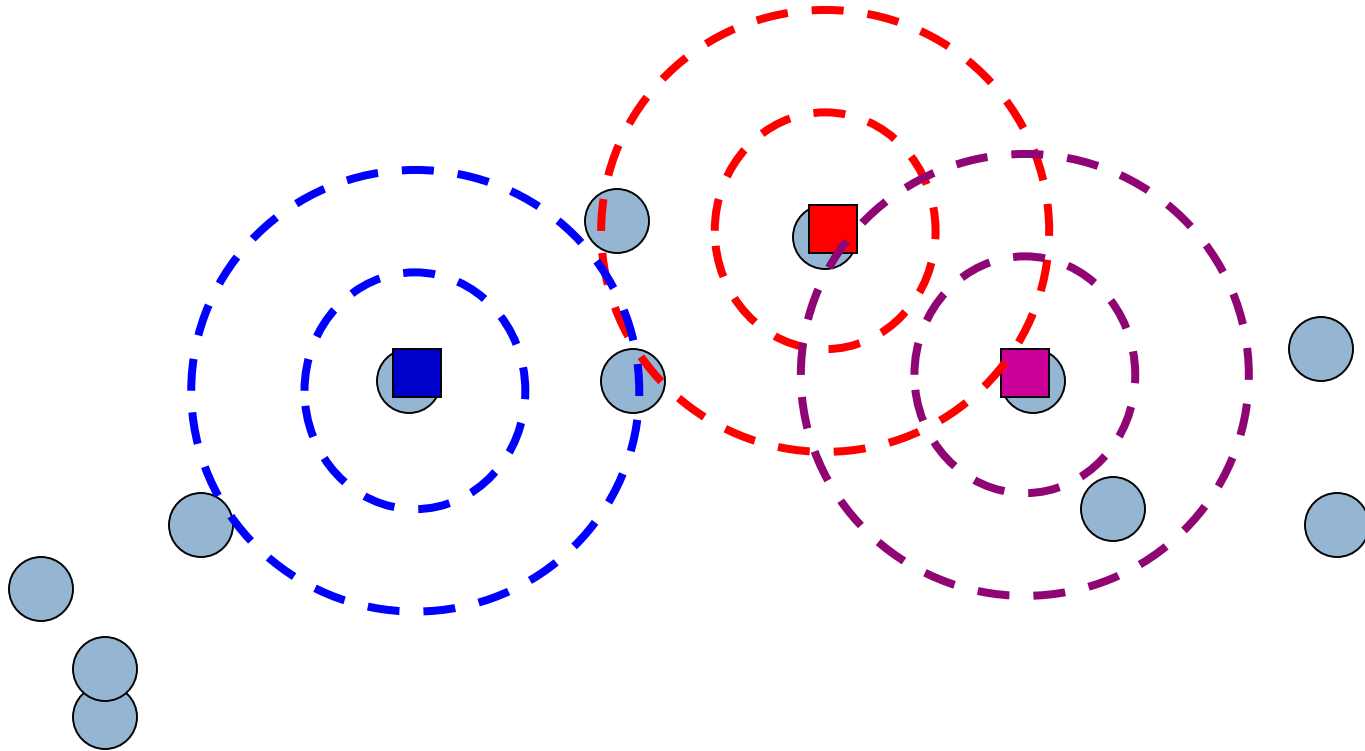


k-means assumes spherical clusters!

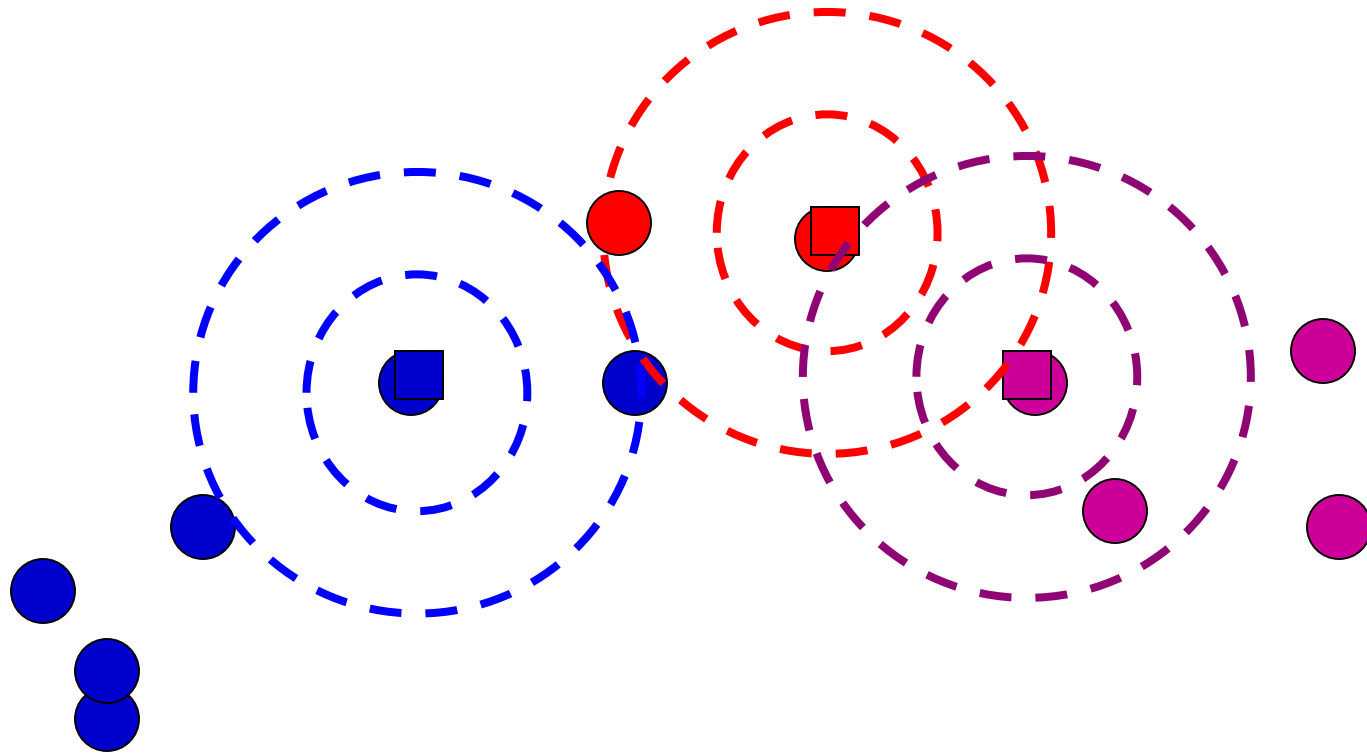
K-means: another view



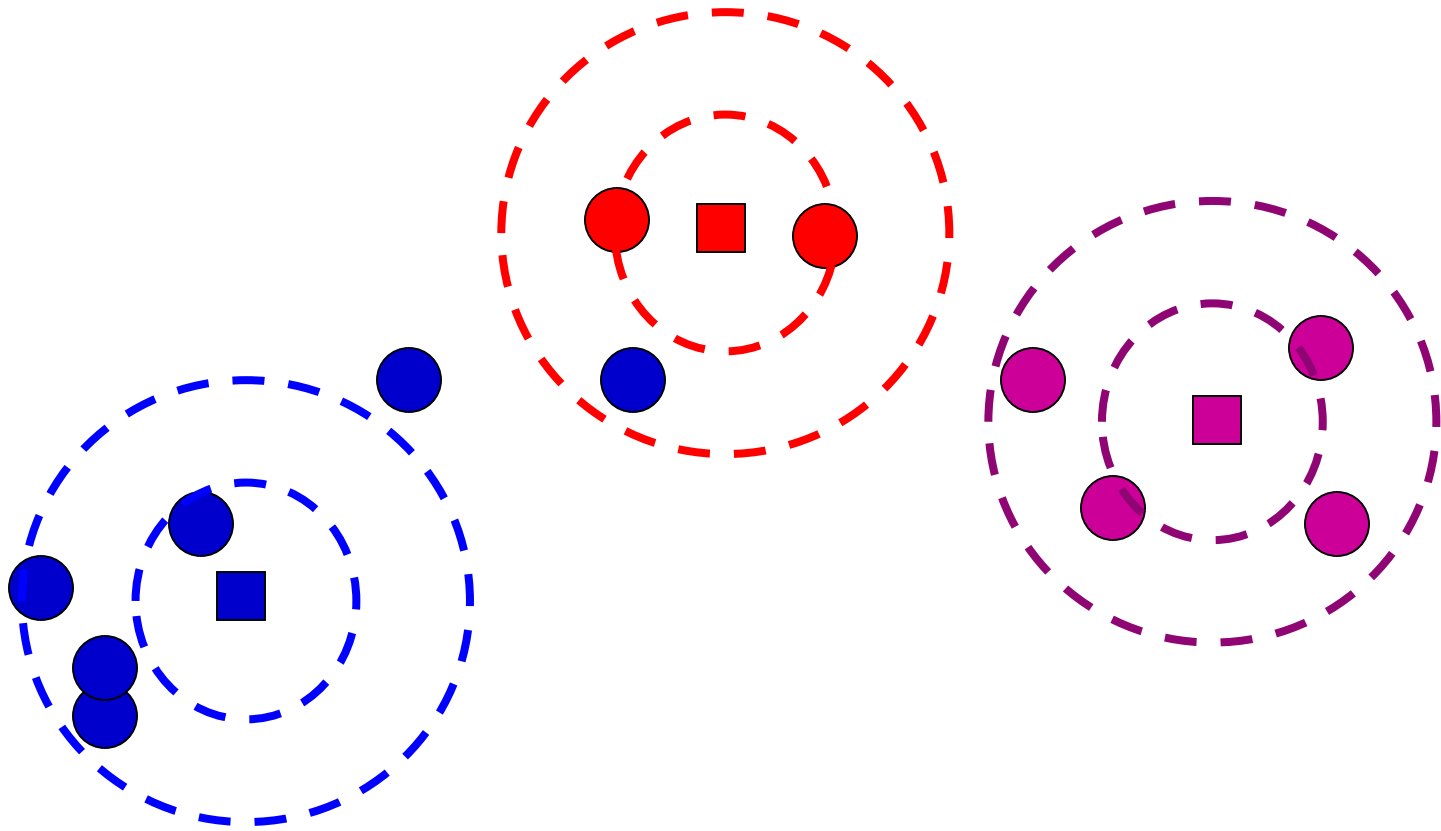
K-means: another view



K-means: assign points to nearest center



K-means: readjust centers

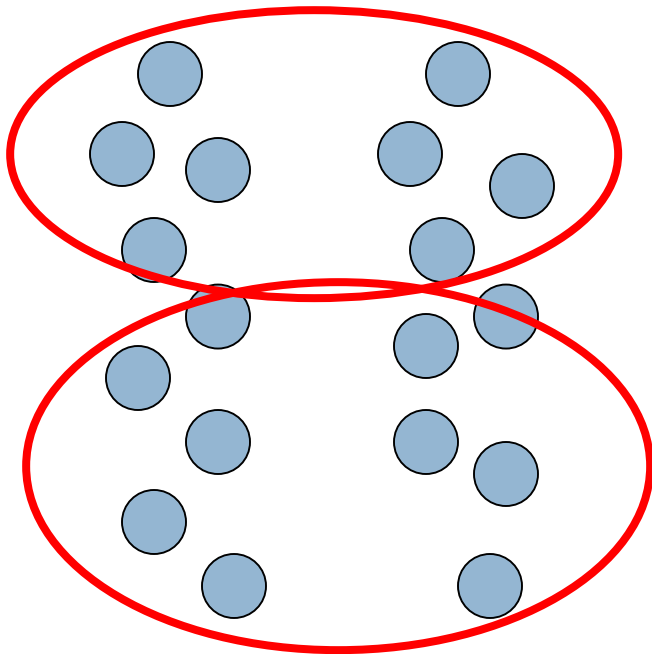


Iteratively learning a collection of spherical clusters

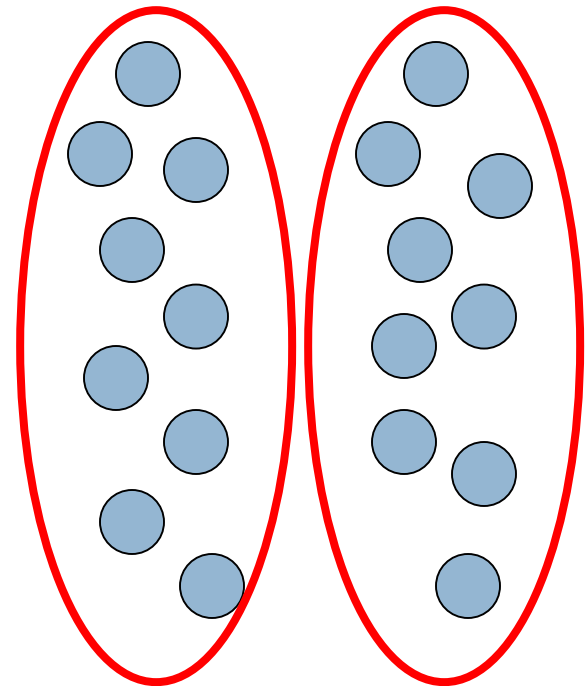
Expectation Maximization (EM) clustering: mixtures of Gaussians

Assume data came from a mixture of Gaussians (**elliptical data**), assign data to cluster with a certain *probability*

k-means



EM



EM clustering

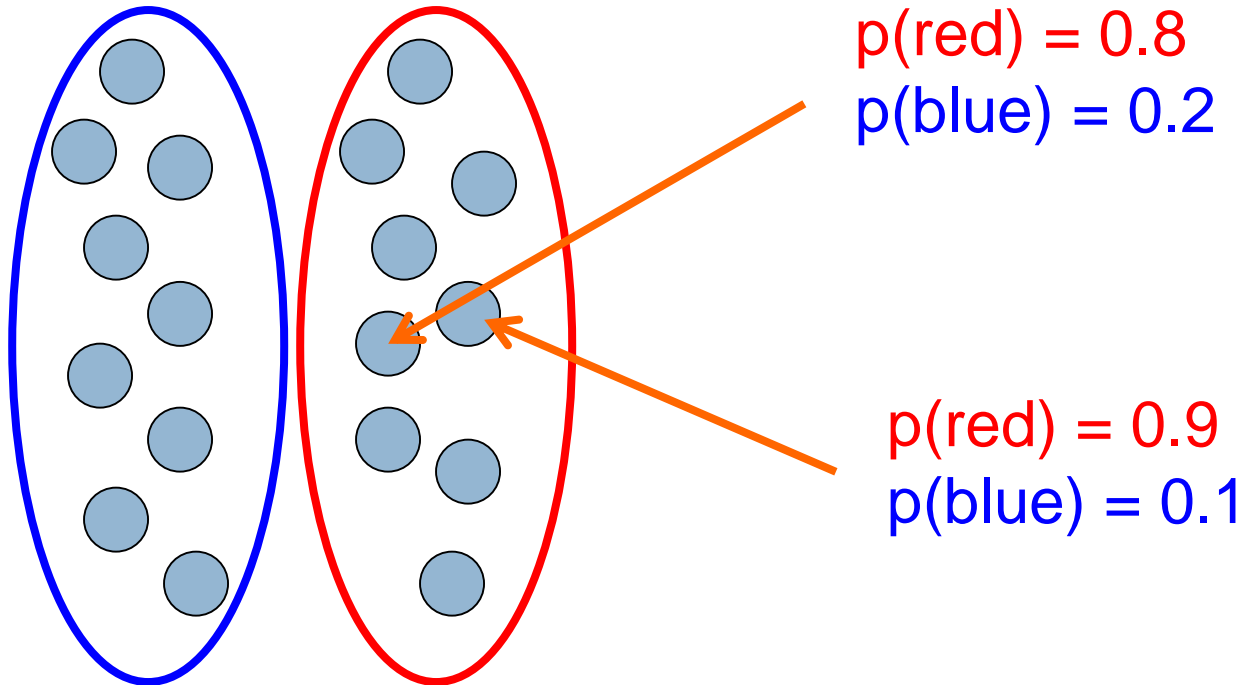
Very similar at a high-level to K-means

Iterate between assigning points and recalculating cluster centers

Two main differences between K-means and EM clustering:

1. We assume elliptical clusters (instead of spherical)
2. It is a “soft” clustering algorithm

Soft clustering



EM clustering

Start with some initial cluster centers

Iterate:

- soft assigned points to each cluster

Calculate: $p(\theta_c | x)$

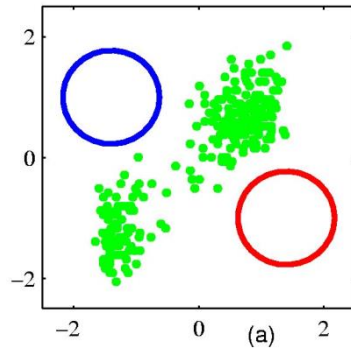
the probability of each point belonging to each cluster

- recalculate the cluster centers

Calculate new cluster parameters, θ_c

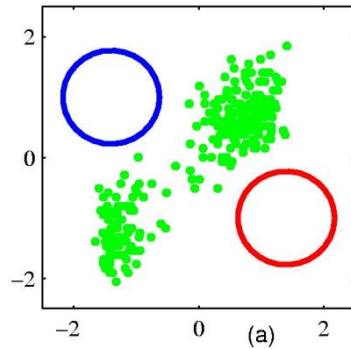
maximum likelihood cluster centers given the current soft clustering

EM example



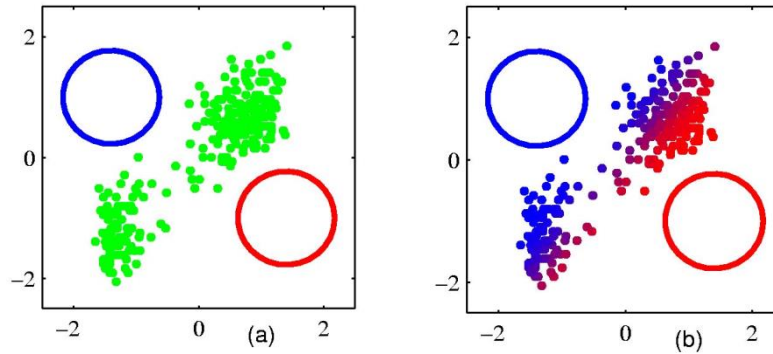
Start with some initial cluster centers

Step 1: soft cluster points



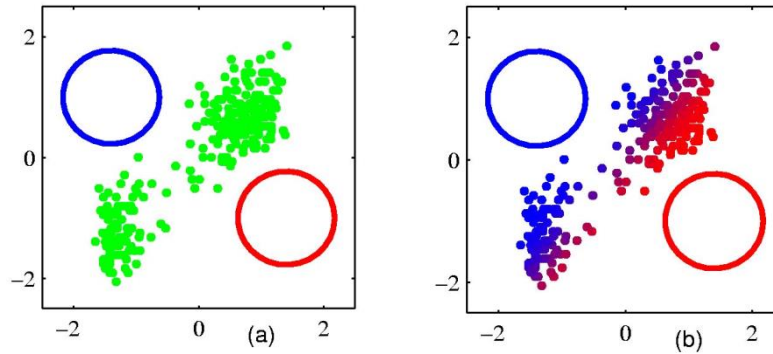
Which points belong to which clusters (soft)?

Step 1: soft cluster points



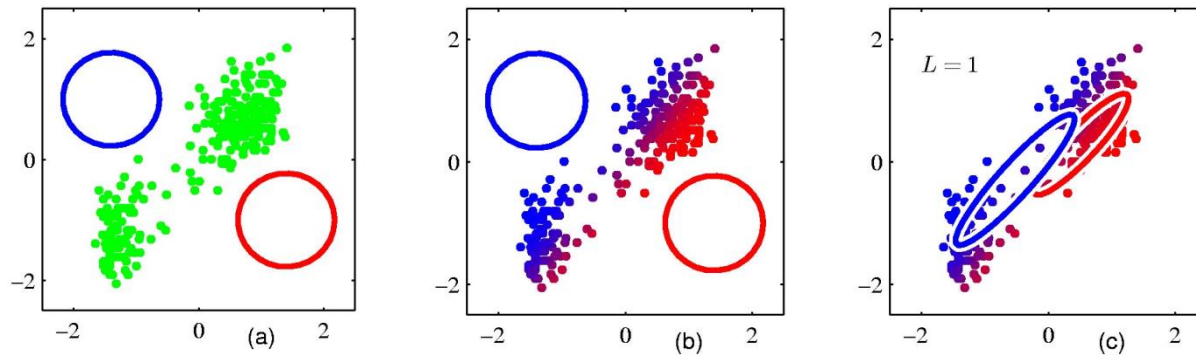
Notice it's a soft (probabilistic) assignment

Step 2: recalculate centers



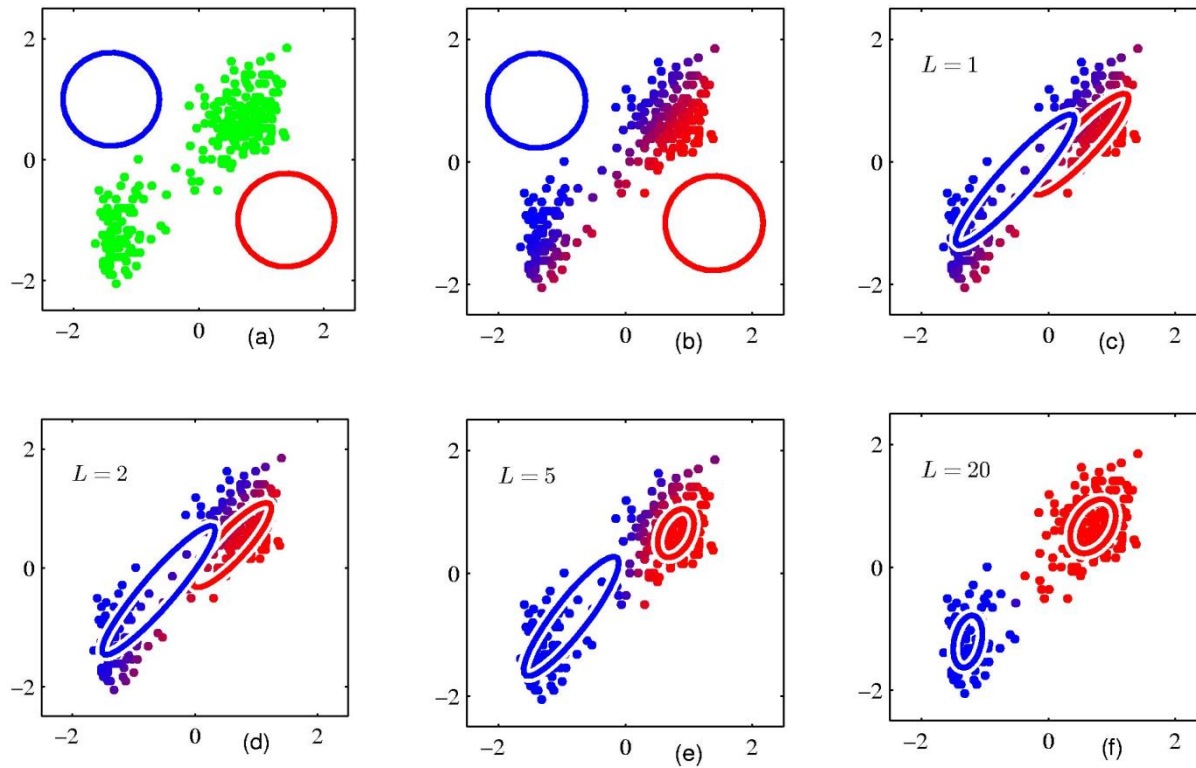
What do the new centers look like?

Step 2: recalculate centers



Cluster centers get a **weighted** contribution from points

keep iterating...



Model: mixture of Gaussians



How do you define a Gaussian (i.e. ellipse)?

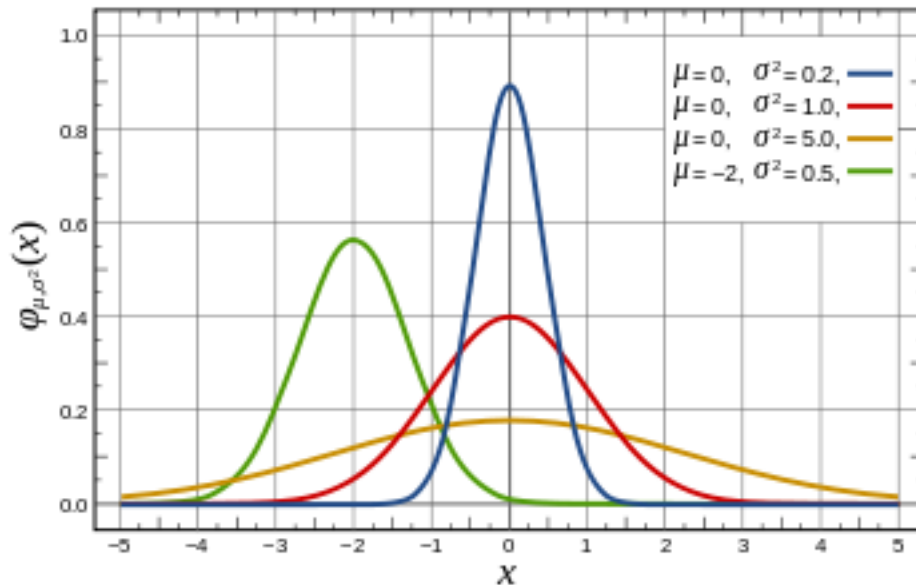
In 1-D?

In M-D?

Gaussian in 1D

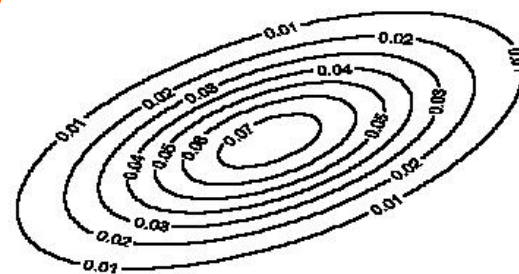
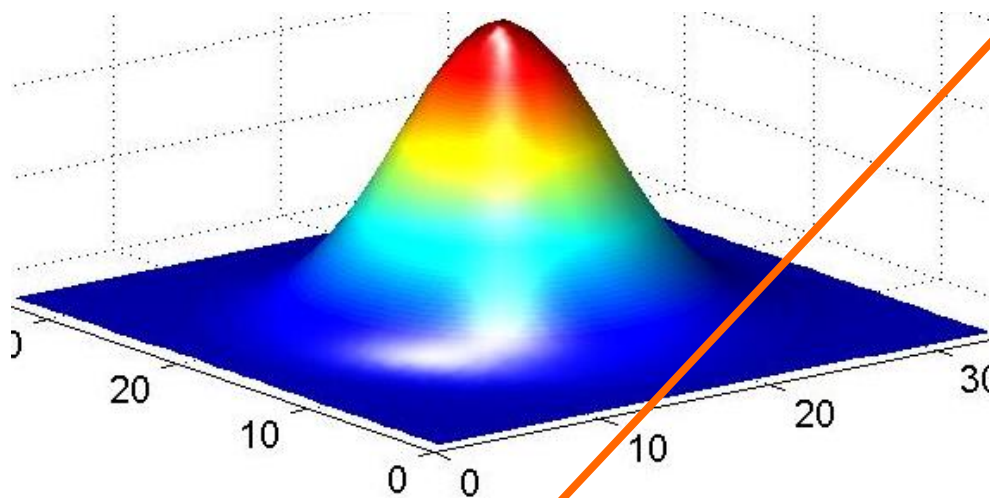
$$f(x; S, q) = \frac{1}{s\sqrt{2p}} e^{-\frac{(x-m)^2}{2s^2}}$$

parameterized by the mean and the standard deviation/variance



Gaussian in multiple dimensions

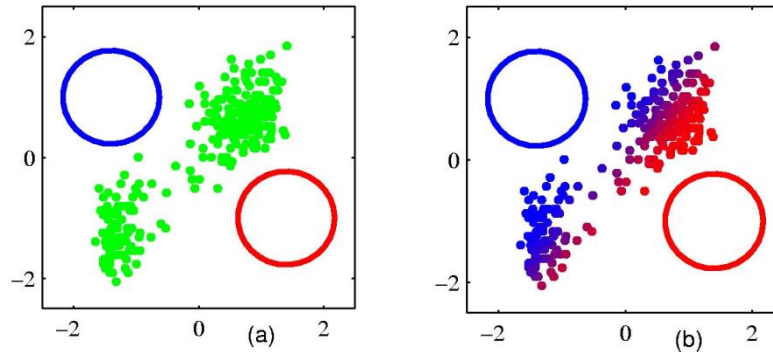
$$N[\mathbf{x}; \mu, \Sigma] = \frac{1}{(2\pi)^{d/2} \sqrt{\det(\Sigma)}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right]$$



Covariance determines the shape of these contours

We learn the means of each cluster (i.e. the center) and the covariance matrix (i.e. how spread out it is in any given direction)

Step 1: soft cluster points



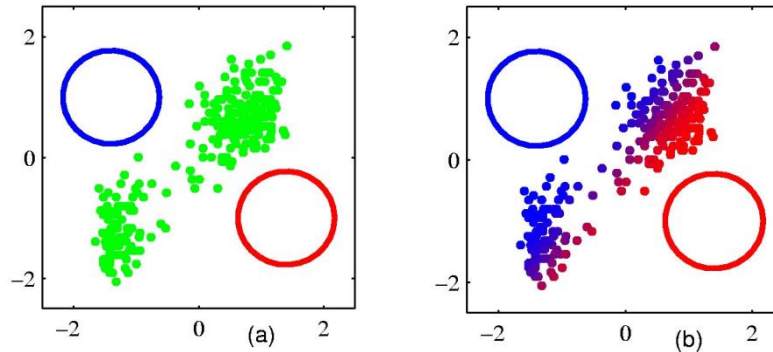
- soft assigned points to each cluster

Calculate: $p(\theta_c|x)$

the probability of each point belonging to each cluster

How do we calculate these probabilities?

Step 1: soft cluster points



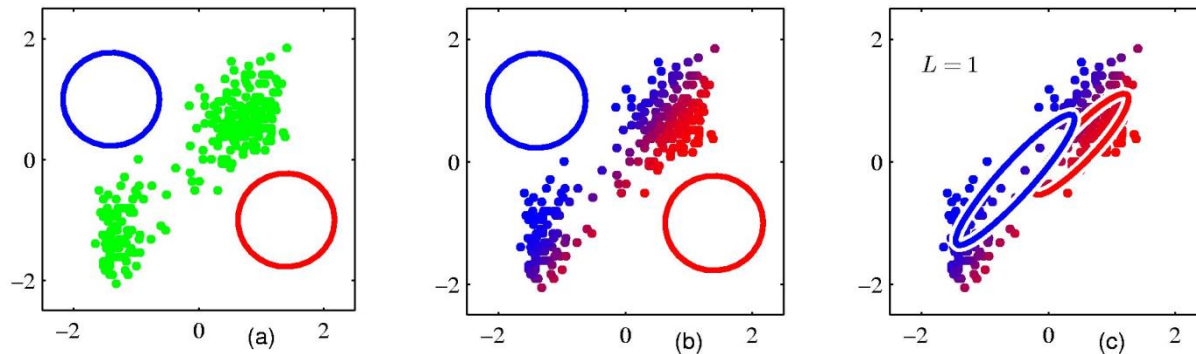
- soft assigned points to each cluster

Calculate: $p(\theta_c | x)$

the probability of each point belonging to each cluster

Just plug into the Gaussian equation for each cluster!
(and normalize to make a probability)

Step 2: recalculate centers



Recalculate centers:

calculate new cluster parameters, θ_c
maximum likelihood cluster centers given the
current soft clustering

How do calculate the cluster centers?

Fitting a Gaussian

What is the “best”-fit Gaussian for this data?

10, 10, 10, 9, 9, 8, 11, 7, 6, ...

Recall this is the 1-D Gaussian equation:

$$f(x; S, q) = \frac{1}{S\sqrt{2p}} e^{-\frac{(x-m)^2}{2S^2}}$$

Fitting a Gaussian

What is the “best”-fit Gaussian for this data?

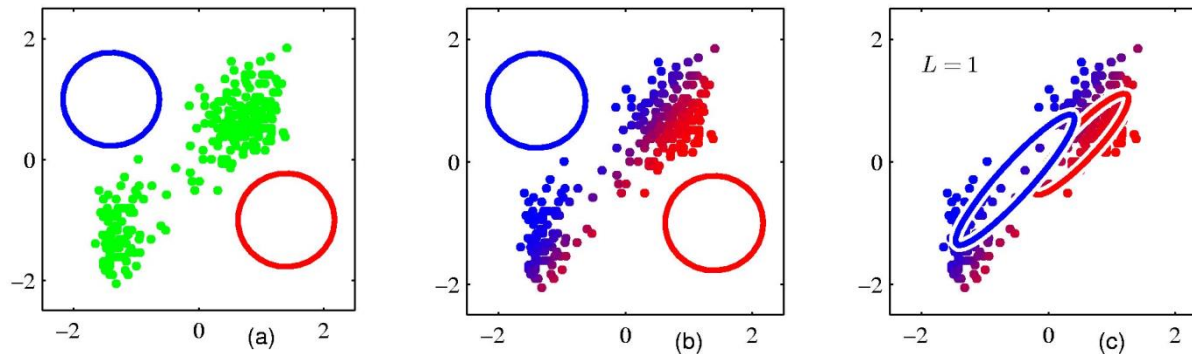
10, 10, 10, 9, 9, 8, 11, 7, 6, ...

The MLE is just the mean and variance of the data!

Recall this is the 1-D Gaussian equation:

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Step 2: recalculate centers

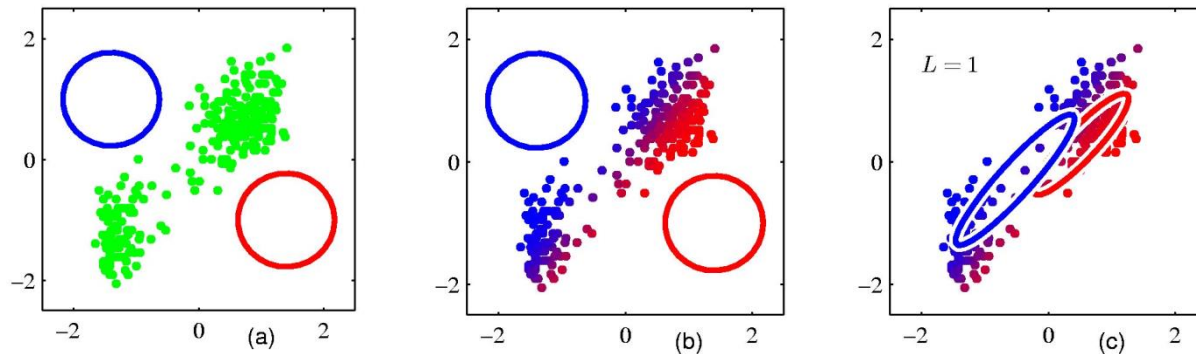


Recalculate centers:

Calculate θ_c
maximum likelihood cluster centers given the
current ***soft clustering***

How do we deal with “soft” data points?

Step 2: recalculate centers



Recalculate centers:

Calculate θ_c
maximum likelihood cluster centers given the
current **soft clustering**

Use fractional counts!

E and M steps: creating a better model

EM stands for Expectation Maximization

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

$p(\theta_c | x)$ What is the probability of each point belonging to each cluster?

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

Just like NB maximum likelihood estimation, except we use fractional counts instead of whole counts

Similar to *k*-means

Iterate:

Assign/cluster each point to closest center

Expectation: Given the current model, figure out the expected probabilities of the points to each cluster

$$p(\theta_c | x)$$

Recalculate centers as the mean of the points in a cluster

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

E and M steps

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

Iterate:

each iterations increases the likelihood of the data and guaranteed to converge (though to a local optimum)!

EM



EM is a general purpose approach for training a model when you don't have labels

Not just for clustering!

- ▣ K-means is just for clustering

One of the most general purpose unsupervised approaches

- ▣ can be hard to get right!

EM is a general framework

Create an initial model, θ'

- ▣ Arbitrarily, randomly, or with a small set of training examples

Use the model θ' to obtain another model θ such that

$$\sum_i \log P_{\theta}(\text{data}_i) > \sum_i \log P_{\theta'}(\text{data}_i) \quad \text{i.e. better models data} \\ \text{(increased log likelihood)}$$

Let $\theta' = \theta$ and repeat the above step until reaching a local maximum

- ▣ Guaranteed to find a better model after each iteration

Where else have you seen EM?

EM shows up all over the place

Training HMMs (Baum-Welch algorithm)

Learning probabilities for Bayesian networks

EM-clustering

Learning word alignments for language translation

Learning Twitter friend network

Genetics

Finance

Anytime you have a model and unlabeled data!

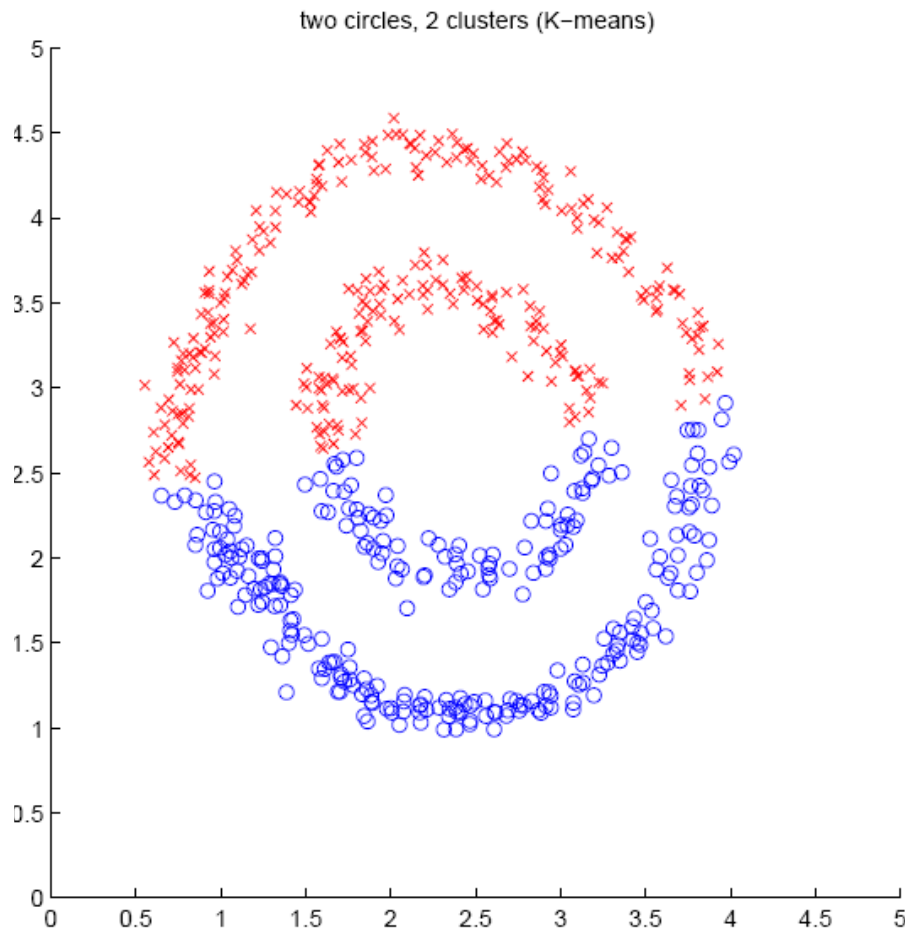
Other clustering algorithms

K-means and EM-clustering are by far the most popular for clustering

However, they can't handle all clustering tasks

What types of clustering problems can't they handle?

Non-gaussian data

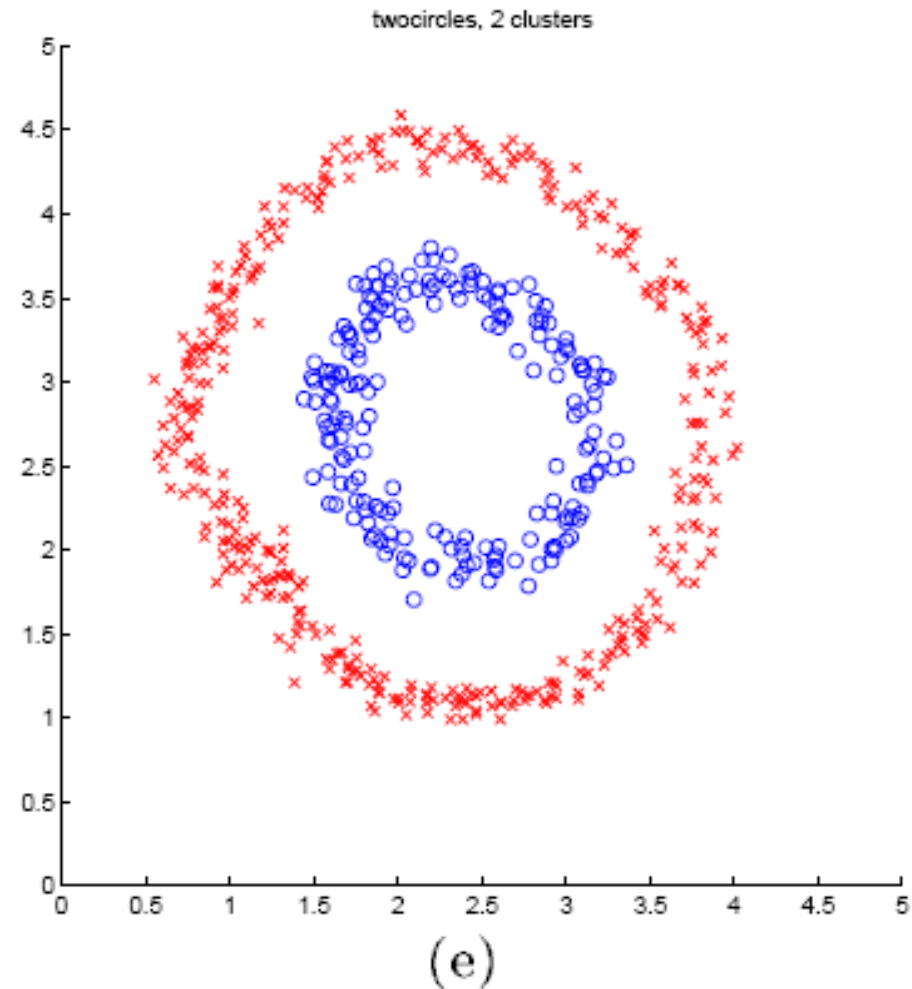
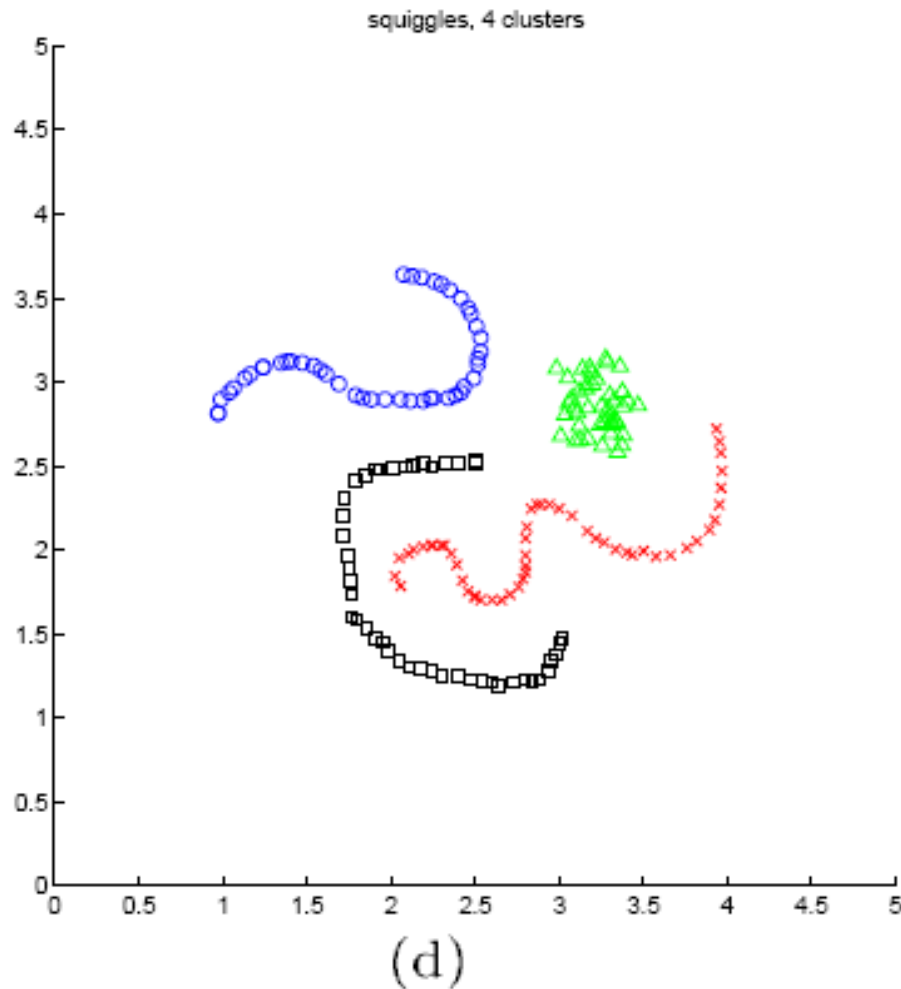


What is the problem?

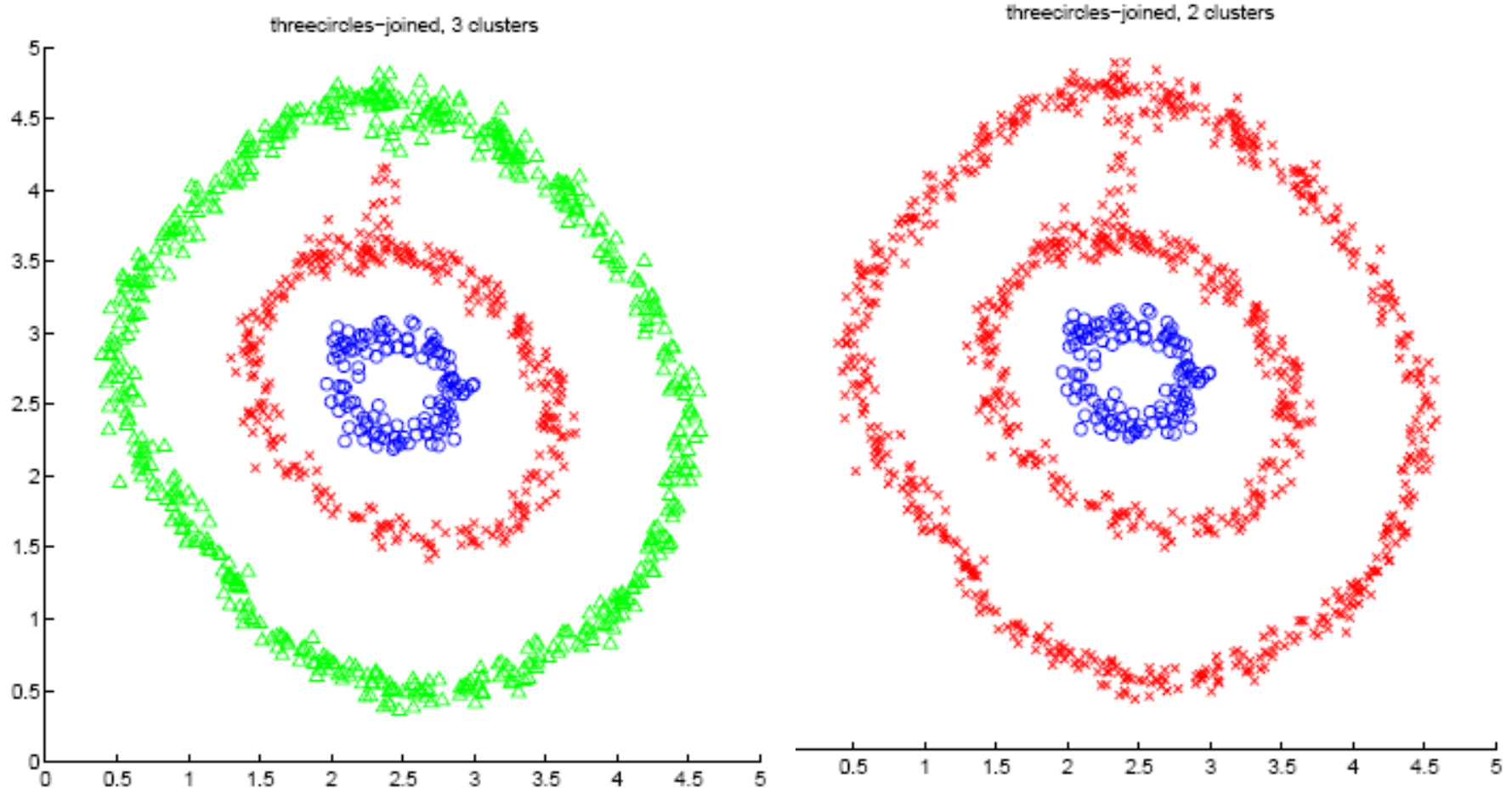
Similar to
classification:
global decision (linear
model) vs. local
decision (K-NN)

Spectral clustering

Spectral clustering examples



Spectral clustering examples



Spectral clustering examples

