UNSUPERVISED LEARNING

David Kauchak
CS 158 – Fall 2016

Administrative

Final project
- Nice work forming groups ☺
- Status report due tomorrow (Wednesday)
- In-class presentation next Tuesday

Midterm

Grading

Supervised learning

label

$label_1$

$label_3$

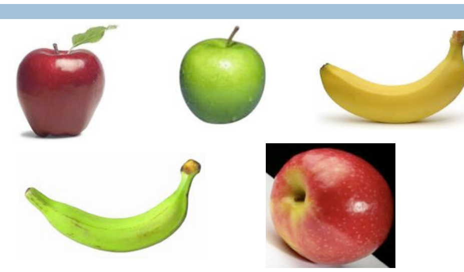
$label_4$

$label_5$

model/ predictor

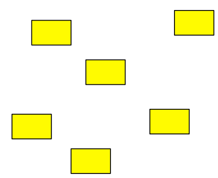
Supervised learning: given labeled examples

Unsupervised learning



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

Unsupervised learning



Given some example without labels, do something!

Unsupervised applications areas

learn clusters/groups without any label

customer segmentation (i.e. grouping)

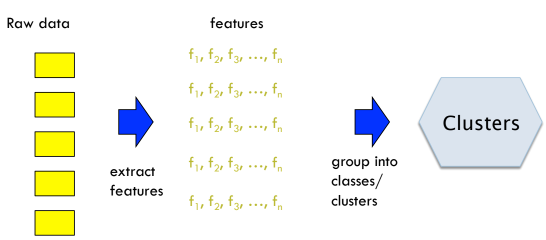
image compression

bioinformatics: learn motifs

find important features

…

Unsupervised learning: clustering



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

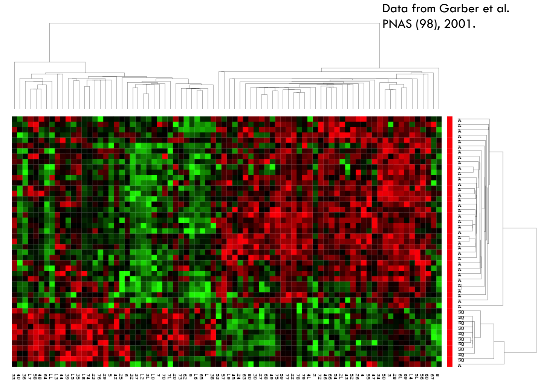
Clustering

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

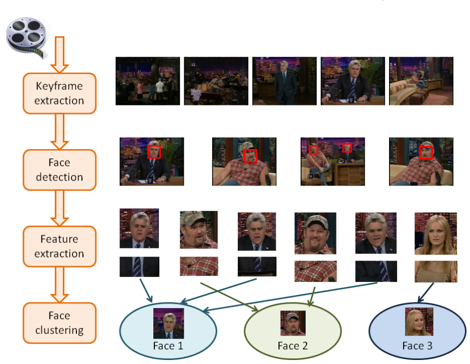
Applications?

Gene expression data

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

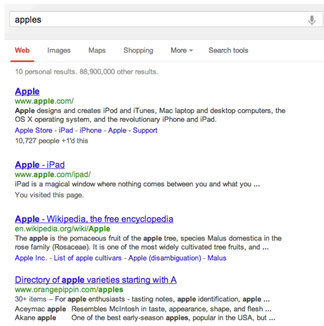


Face Clustering

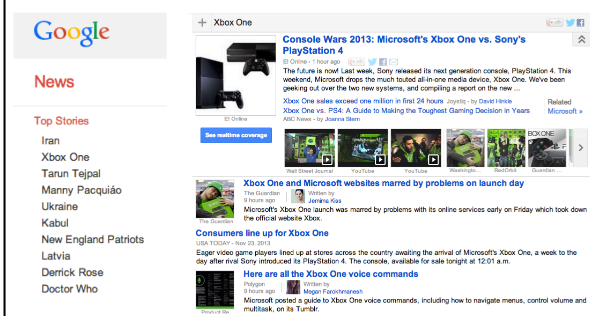


Face clustering

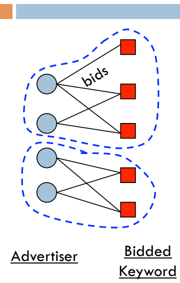
Search result clustering



Google News



Clustering in search advertising

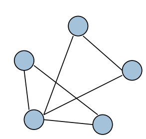


~10M nodes

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

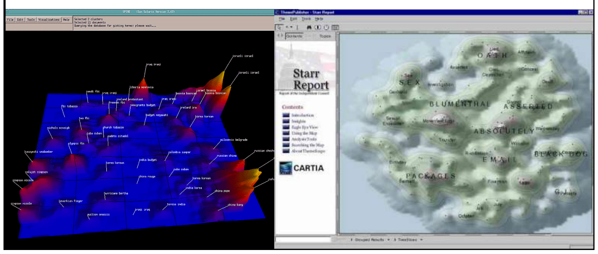
- Distributed pagerank computation

Data visualization

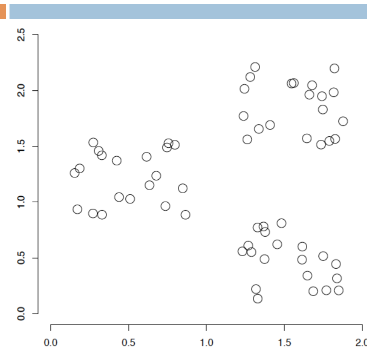
Wise et al, "Visualizing the non-visual" PNNL

ThemeScapes, Cartia

- [Mountain height = cluster size]



A data set with clear cluster structure



What are some of the issues for clustering?

What clustering algorithms have you seen/used?

Issues for clustering

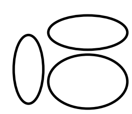
Representation for clustering
- How do we represent an example
  - features, etc.
- Similarity/distance between examples

Flat clustering or hierarchical

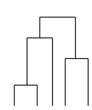
Number of clusters
- Fixed a priori
- Data driven?

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

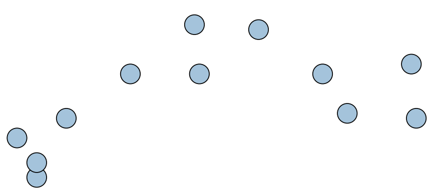
Most well-known and popular clustering algorithm:

Start with some initial cluster centers

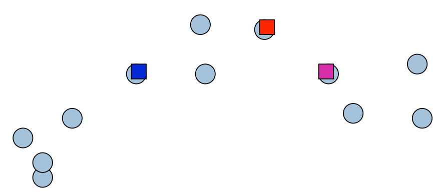
Iterate:

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

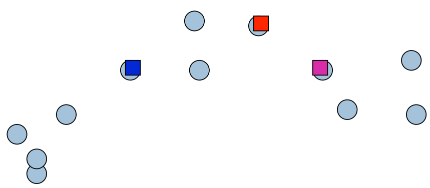
K-means: an example



K-means: Initialize centers randomly

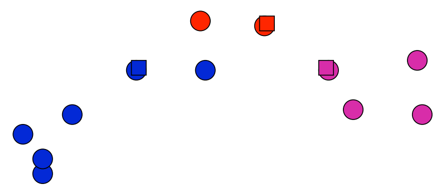


K-means: Initialize centers randomly

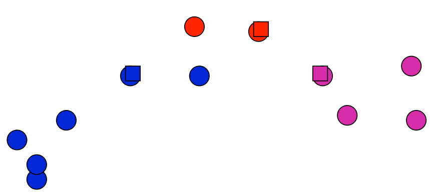


What points are closest?

K-means: assign points to nearest center

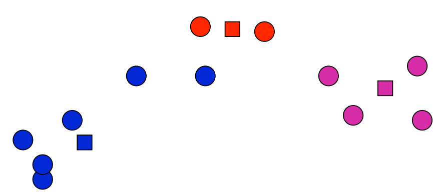


K-means: assign points to nearest center

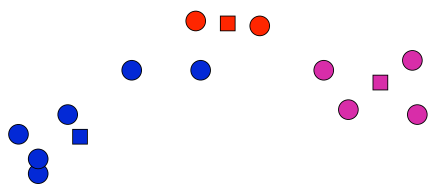


Where are the new centers?

K-means: readjust centers

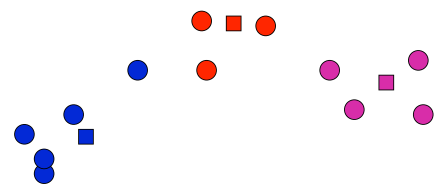


K-means: readjust centers

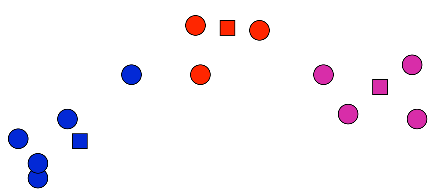


What points are closest?

K-means: assign points to nearest center

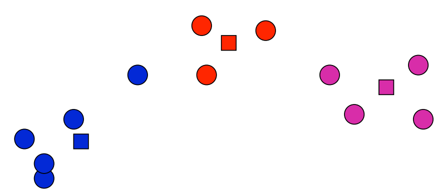


K-means: assign points to nearest center

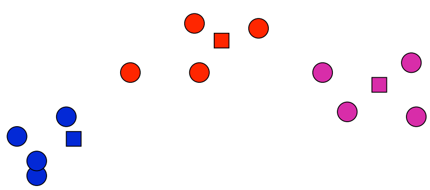


Where are the new centers?

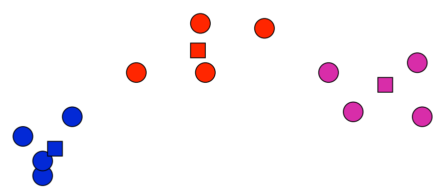
K-means: readjust centers



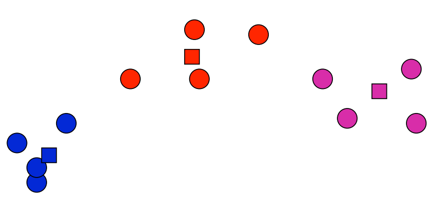
K-means: assign points to nearest center



K-means: readjust centers

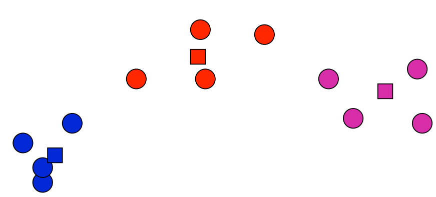


K-means: readjust centers



When do we stop?

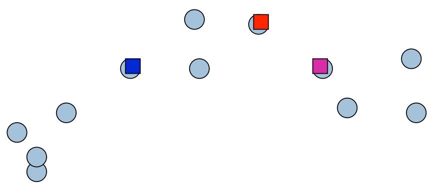
K-means: assign points to nearest center



No changes: Done

K-means

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

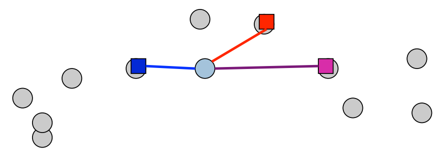


How do we do this?

K-means

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

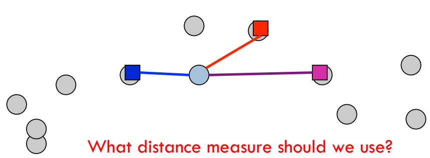
  iterate over each point:
  - get distance to each cluster center
  - assign to closest center (hard cluster)
- Recalculate centers as the mean of the points in a cluster



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

Euclidean:

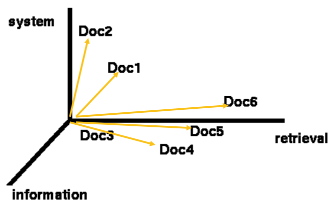
$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

good for spatial data

Clustering documents (e.g. wine data)

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

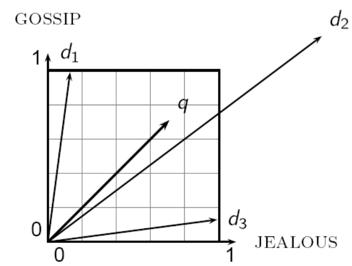
Documents are points or vectors in this space



When Euclidean distance doesn't work

Which document is closest to q using Euclidian distance?

Which do you think should be closer?

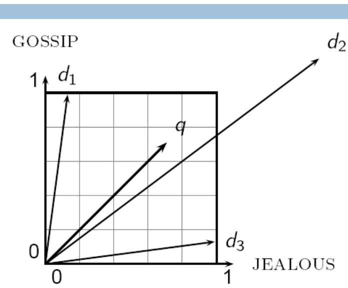


Issues with Euclidian distance

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

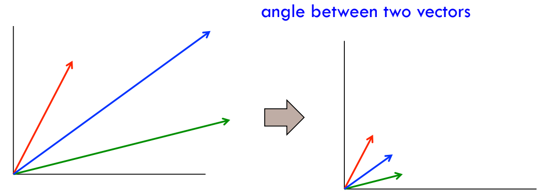
This is not what we want!



cosine similarity

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

correlated with the angle between two vectors



cosine distance

cosine similarity ranges from 0 and 1, with things that are similar 1 and dissimilar 0

cosine distance:

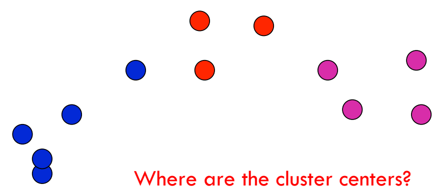
$$d(x, y) = 1 - sim(x, y)$$

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

K-means

Iterate:

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

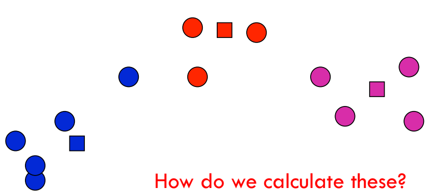


Where are the cluster centers?

K-means

Iterate:

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



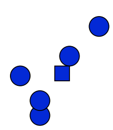
How do we calculate these?

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:



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

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:

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

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

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

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

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

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

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

1. Any other assignment would end up in a larger loss
2. 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

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

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

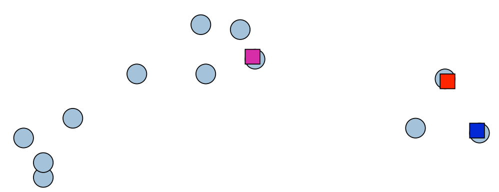
Initial (seed) cluster centers

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

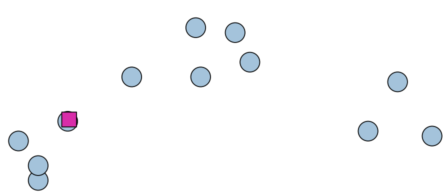
μ_1 = pick random point

for i = 2 to K:

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

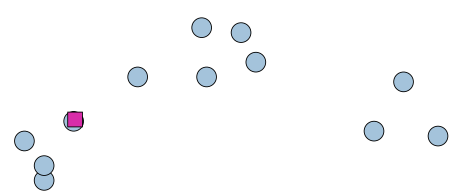
$$\mu_i = \underbrace{\underset{x}{\arg\max}}_{\text{point with the largest distance to any previous center}} \underbrace{\min_{\mu_j : 1 < j < i} d(x, \mu_j)}_{\text{smallest distance from x to any previous center}}$$

K-means: Initialize furthest from centers



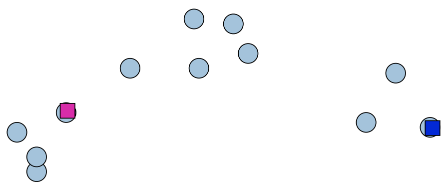
Pick a random point for the first center

K-means: Initialize furthest from centers



What point will be chosen next?

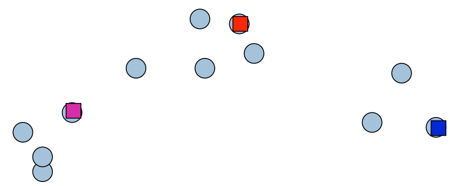
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

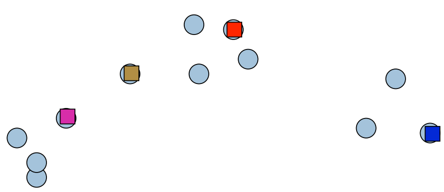
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

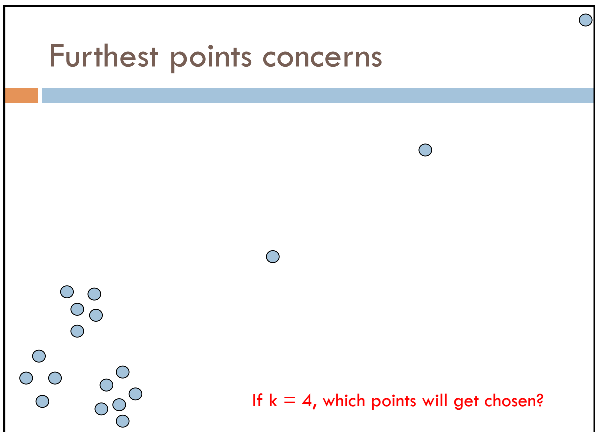
K-means: Initialize furthest from centers



Furthest point from center

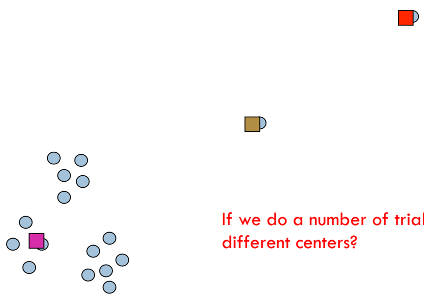
Any issues/concerns with this approach?

Furthest points concerns



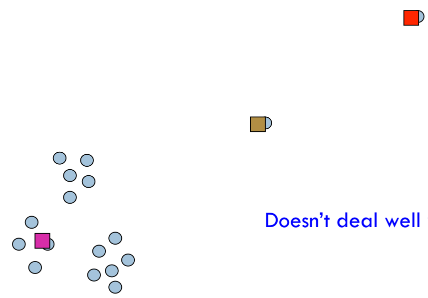
If k = 4, which points will get chosen?

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 variations/parameters

~~Initial (seed) cluster centers~~

~~Convergence~~

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

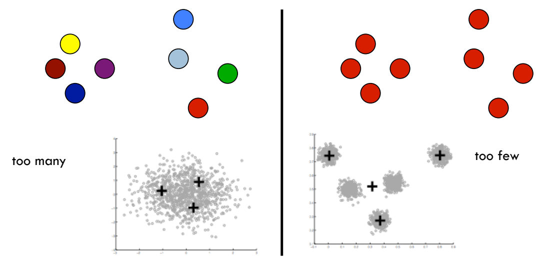
K!

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?



Many approaches

Regularization!!!

Statistical test

k-means loss revisited

K-means is trying to minimize:

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

What happens when k increases?

k-means loss revisited

K-means is trying to minimize:

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

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:

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

2 regularization options

$$loss_{BIC} = loss_{kmeans} + K \log N$$ (where N = 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

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

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

AIC penalizes increases in K more harshly

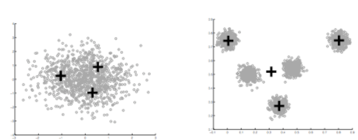
Both require a change to the K-means algorithm

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

Statistical approach

Assume data is Gaussian (i.e. spherical)

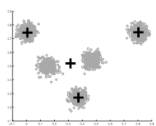
Test for this
- Testing in high dimensions doesn't work well
- Testing in lower dimensions does work well



ideas?

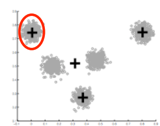
Project to one dimension and check

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

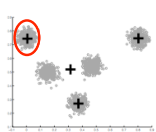
For each cluster, project down to one dimension
- Use a statistical test to see if the data is Gaussian

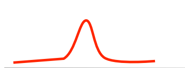


What will this look like projected to 1-D?

Project to one dimension and check

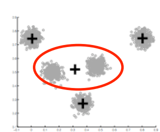
For each cluster, project down to one dimension
- Use a statistical test to see if the data is Gaussian





Project to one dimension and check

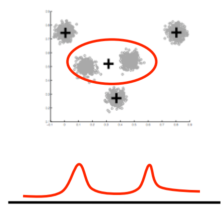
For each cluster, project down to one dimension
- Use a statistical test to see if the data is Gaussian



What will this look like projected to 1-D?

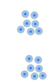
Project to one dimension and check

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

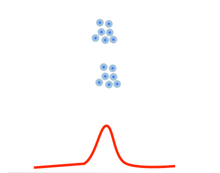
For each cluster, project down to one dimension
- Use a statistical test to see if the data is Gaussian



What will this look like projected to 1-D?

Project to one dimension and check

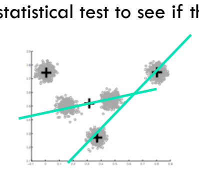
For each cluster, project down to one dimension
- Use a statistical test to see if the data is Gaussian



Solution?

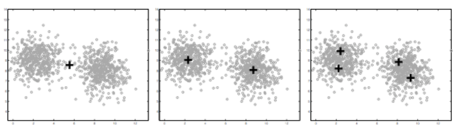
Project to one dimension and check

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

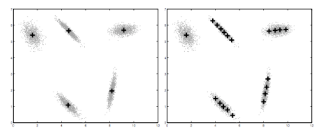


Figure 4: 2-*d* synthetic dataset with 5 true clusters. On the left, G-means correctly chooses 5 centers and deals well with non-spherical data. On the right, the BIC causes X-means to overfit the data, choosing 20 unevenly distributed clusters.

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, *I* 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 points gets added once to some centroid: O(*nm*)
- Assume these two steps are each done once for *I* iterations: O(*Iknm*)

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