UNSUPERVISED LEARNING

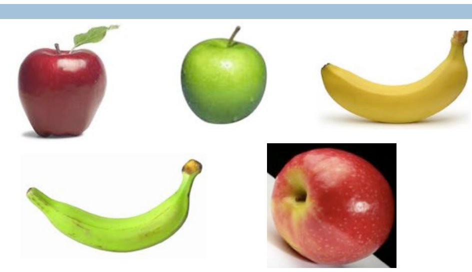
David Kauchak
CS 451 – Fall 2013

Administrative

Final project

Schedule for the rest of the semester

Unsupervised learning



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

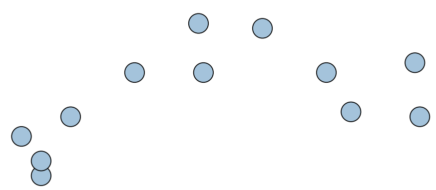
K-means

Start with some initial cluster centers

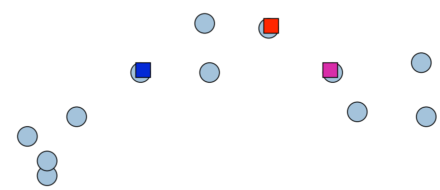
Iterate:

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

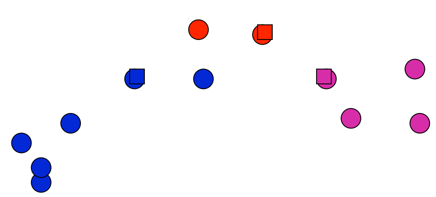
K-means: an example



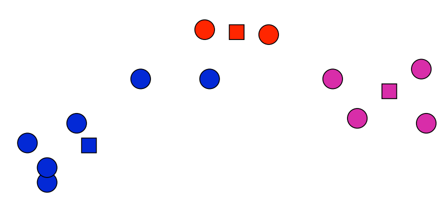
K-means: Initialize centers randomly



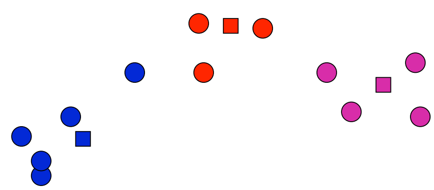
K-means: assign points to nearest center



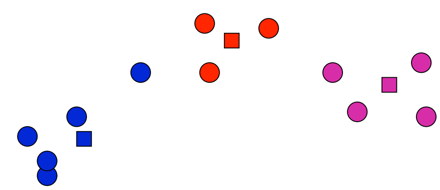
K-means: readjust centers



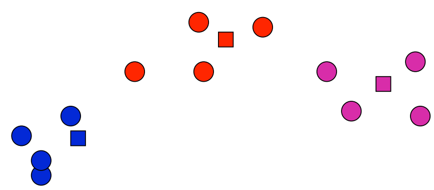
K-means: assign points to nearest center



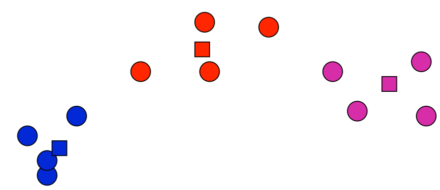
K-means: readjust centers



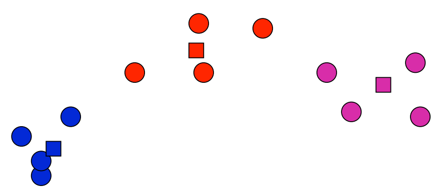
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center



No changes: Done

K-means variations/parameters

~~Initial (seed) cluster centers~~

~~Convergence~~

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

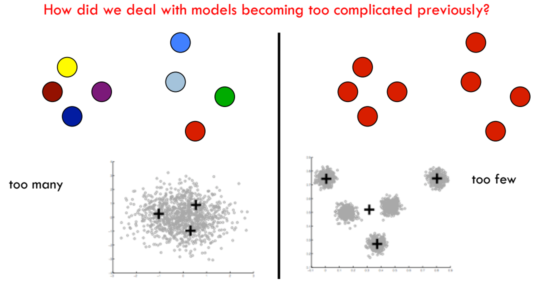
K!

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

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

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

What effect will this have?
Which will tend to produce smaller k?

k-means loss revisited

2 regularization options

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

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

AIC penalizes increases in K more harshly

Both require a change to the K-means algorithm

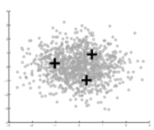
Tend to work reasonably well in practice if you don't know K

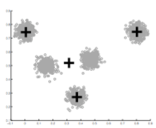
Statistical approach

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



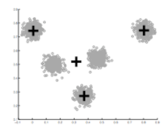


ideas?

Project to one dimension and check

For each cluster, project down to one dimension

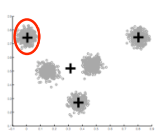
- Use a statistical test to see if the data is Gaussian



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

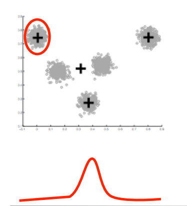


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

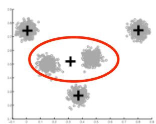
- Use a statistical test to see if the data is Gaussian



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

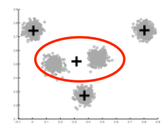


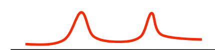
What will this look like projected to 1-D?

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

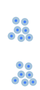




Project to one dimension and check

For each cluster, project down to one dimension

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

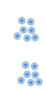


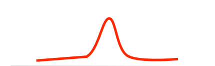
What will this look like projected to 1-D?

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

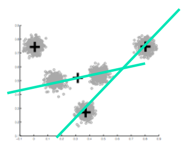




Solution?

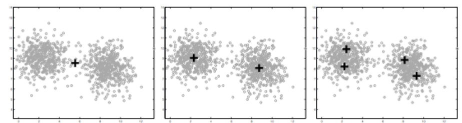
Project to one dimension and check

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

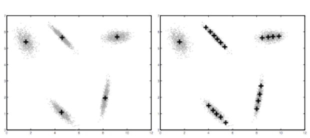


Figure 4: 2-*d* synthetic dataset with 5 true clusters. On the left, G-means correctly chooses 5 centers and deals well with non-spherical data. On the right, the BIC causes *X*-means to overfit the data, choosing 20 unevenly distributed clusters.

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

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

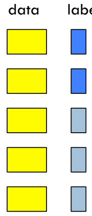
How would you evaluate clustering?

Common approach: use labeled data

Use data with known classes

- For example, document classification data

data label



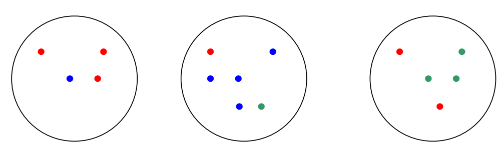
If we clustered this data (ignoring labels) what would we like to see?

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:

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

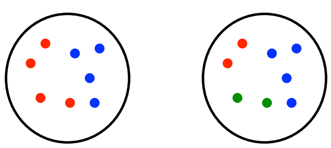
Purity issues…

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

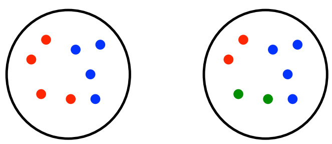
$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

where $p(class_i)$ is proportion of class *i* in cluster

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i) \log p(class_i)$$

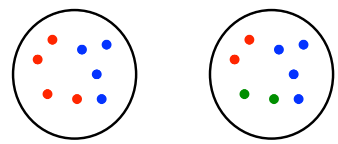


entropy?

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i) \log p(class_i)$$



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

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