Machine learning: Unsupervised learning

David Kauchak
cs311
Spring 2013

adapted from:
http://www.stanford.edu/class/cs276/handouts/lecture17-clustering.ppt

Administrative

- Assignment 5 out soon

Machine learning code

- Weka
  - Java based
  - Tons of approaches
  - Good for playing with different approaches, but faster/better of individual approaches can be found
  - http://www.cs.waikato.ac.nz/ml/weka/
- SVMs
  - SVMLight (C-based… fast)
    - http://www.cs.cornell.edu/People/tj/svm_light/
  - LIBSVM (Java)
    - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Others
  - many others out there… search for them
  - e.g. PyML (in Python, but I've never used it)

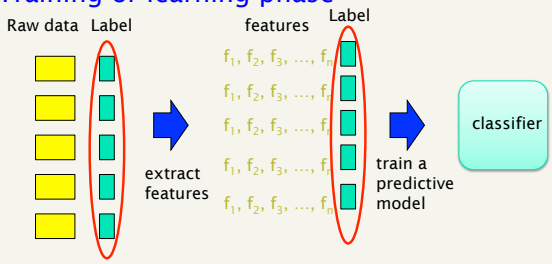
Supervised learning (e.g. classification)

APPLES **BANANAS**

Supervised learning: given labeled data

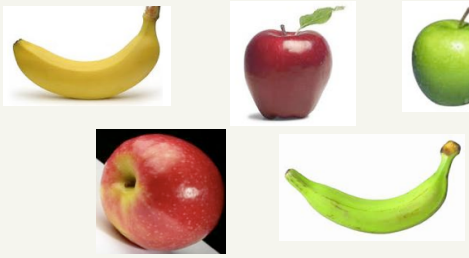
Supervised learning

Training or learning phase



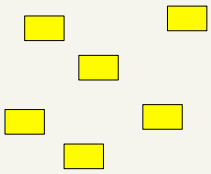
User "supervision", we're given the labels (classes)

Unsupervised learning



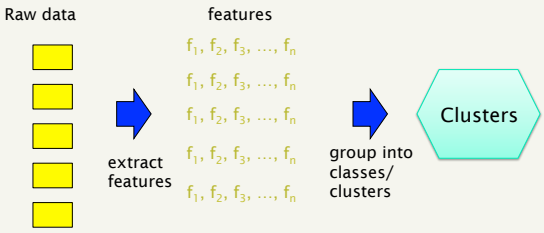
Unsupervised learning: put these into groups

Unsupervised learning



Given some example without labels, do something!

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

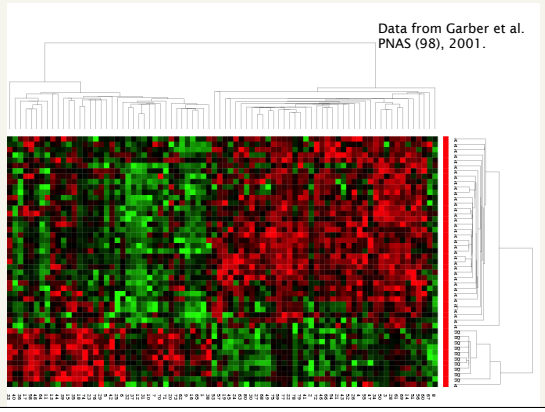
- Learn a translation dictionary
- Learn a grammar for a language
- Learn the social graph

Clustering

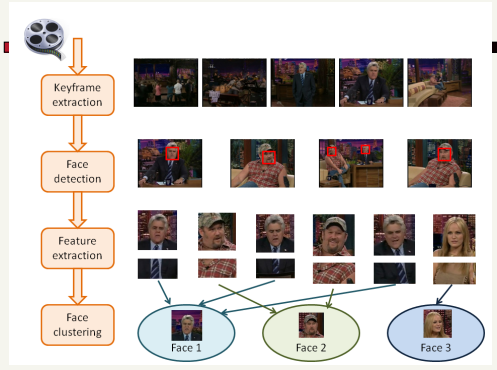
Clustering: the process of grouping a set of objects into classes of similar objects

Applications?

Gene expression data

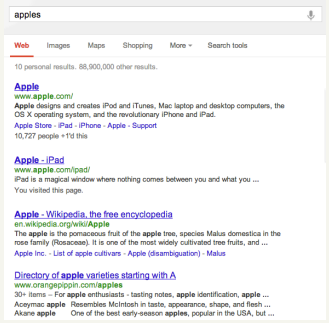


Face Clustering

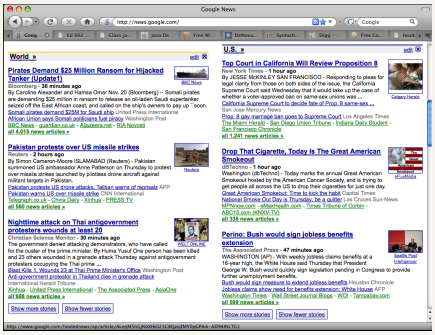


Face clustering

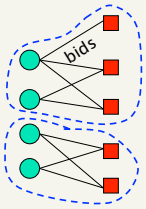
Search result clustering



Google News



Clustering in search advertising



Advertiser

Bidded Keyword

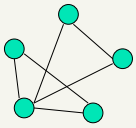
~10M nodes

Find clusters of advertisers and keywords

- Keyword suggestion
- Performance estimation

16

Clustering applications



Who-messages-who
IM/text/twitter graph

~100M
nodes

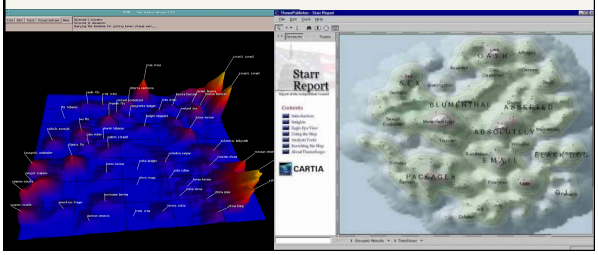
Find clusters of users
- Targeted advertising
- Exploratory analysis

Clusters of the Web Graph
- Distributed pagerank computation

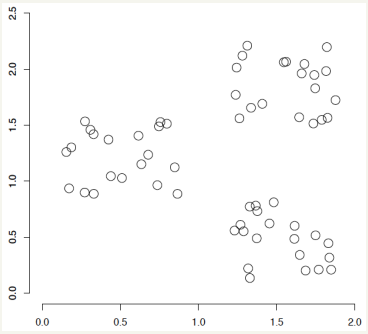
17

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

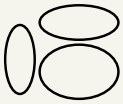
Flat clustering or hierarchical

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

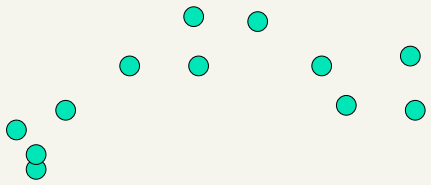
Most well-known and popular clustering algorithm:

Start with some initial cluster centers

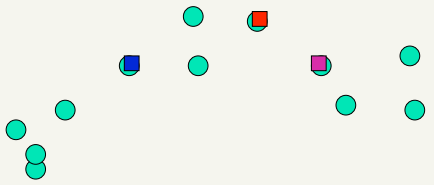
Iterate:

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

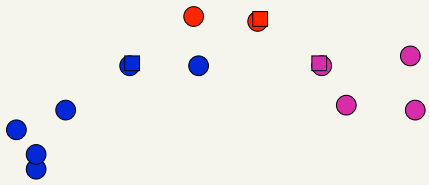
K-means: an example



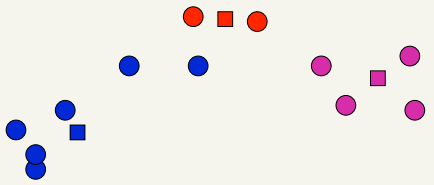
K-means: Initialize centers randomly



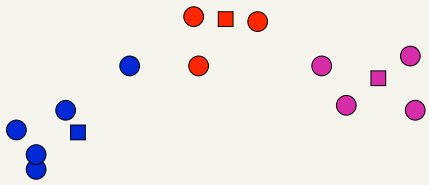
K-means: assign points to nearest center



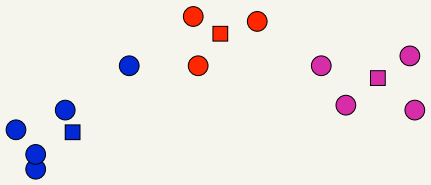
K-means: readjust centers



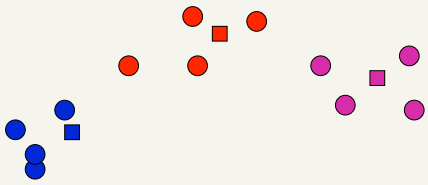
K-means: assign points to nearest center



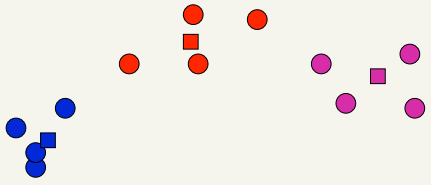
K-means: readjust centers



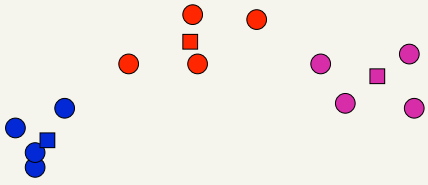
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center

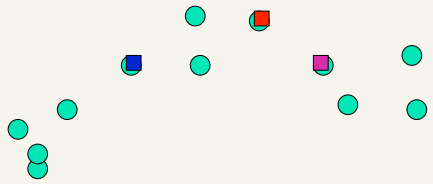


No changes: Done

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?

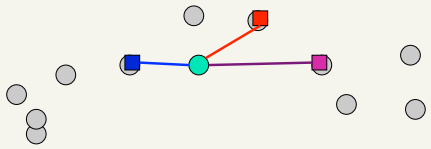
K-means

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



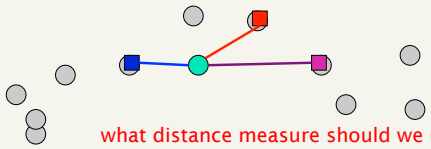
K-means

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

Euclidean:

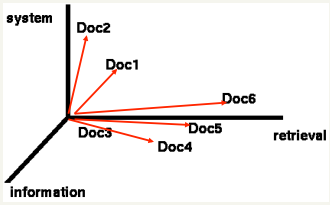
$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

good for spatial data

Clustering documents

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

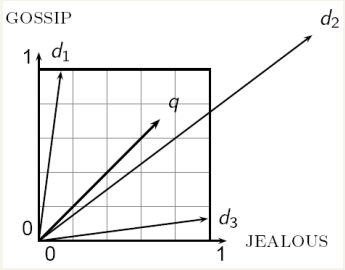
Documents are points or vectors in this space



When Euclidean distance doesn't work

Which document is closest to q using Euclidian distance?

Which do you think should be closer?

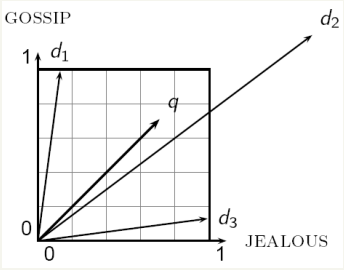


Issues with Euclidian distance

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

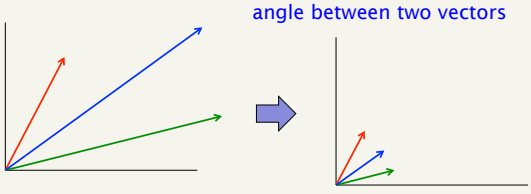
This is not what we want!



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

We want a distance measure, cosine distance:

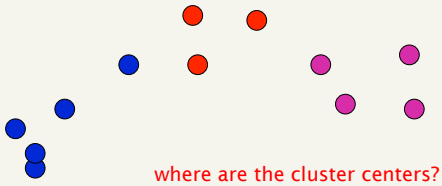
$$d(x, y) = 1 - sim(x, y)$$

good for text data and many other "real world" data sets

K-means

Iterate:

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

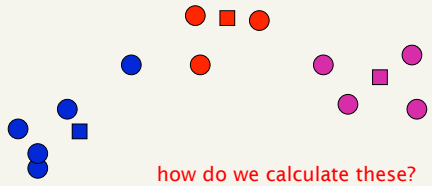


where are the cluster centers?

K-means

Iterate:

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

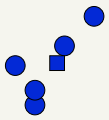


how do we calculate these?

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 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

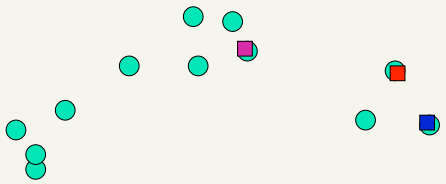
Initial (seed) cluster centers

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
- Try out multiple starting points
- Initialize with the results of another clustering method

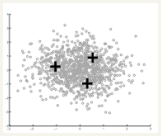
How Many Clusters?

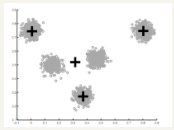
Number of clusters *K* must be provided

Somewhat application dependent

How should we determine the number of clusters?

too many





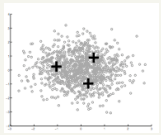
too few

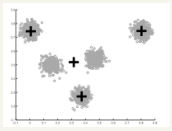
One approach

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



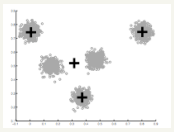


ideas?

Project to one dimension and check

For each cluster, project down to one dimension

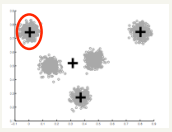
- Use a statistical test to see if the data is Gaussian



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

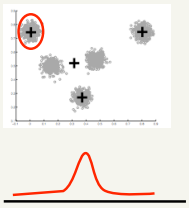


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

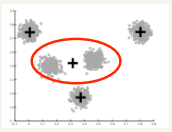
- Use a statistical test to see if the data is Gaussian



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

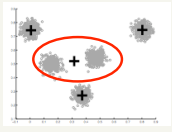


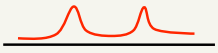
What will this look like projected to 1-D?

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

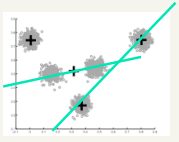




Project to one dimension and check

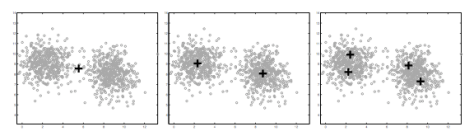
For each cluster, project down to one dimension

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



The dimension of the projection is based on the data

On synthetic data



Split too far

Compared to other approaches

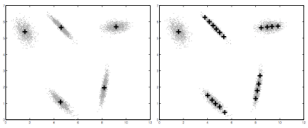


Figure 4: 2-*d* synthetic dataset with 5 true clusters. On the left, G-means correctly chooses 5 centers and deals well with non-spherical data. On the right, the BIC causes X-means to overfit the data, choosing 20 unevenly distributed clusters.

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
- 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.
- 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:

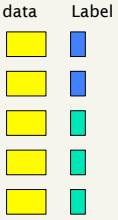
- the intra-class (that is, intra-cluster) similarity is high
- the inter-class similarity is low

How would you evaluate clustering?

Common approach: use labeled data

Use data with known classes

- For example, document classification data



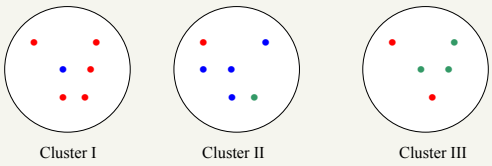
If we clustered this data (ignoring labels) what would we like to see?

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: Purity = 1/6 (max(5, 1, 0)) = 5/6

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

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

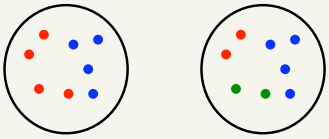
Other purity issues…

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

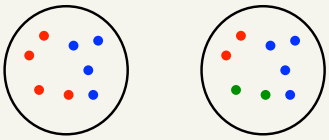
$$entropy(cluster) = -\frac{1}{num\_classes}\sum_i p(class_i)\log p(class_i)$$

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

Common approach: use labeled data

Average entropy of classes in clusters

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

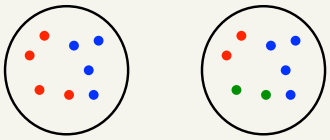


entropy?

Common approach: use labeled data

Average entropy of classes in clusters

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



$-0.5\log 0.5 - 0.5\log 0.5 = 1$ $-0.5\log 0.5 - 0.25\log 0.25 - 0.25\log 0.25 = 1.5$