# Clustering: k-means, the EM algorithm

Based partly on: Dr. P Matuszek, Dr. Mooney: www.cs.utexas.edu/~mooney/cs388/slides/TextClustering.ppt

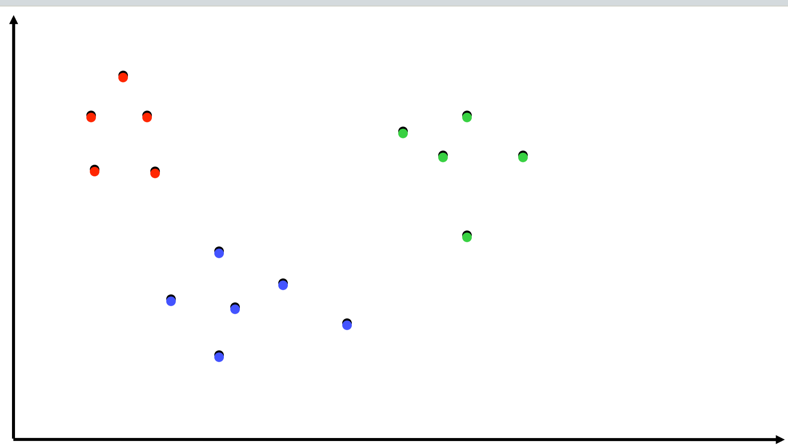
## Bookkeeping

- No HW 6
- Phase II
  - New eleusis.py, Adversary class
  - Summary:
  - Maintain a hand of 14 cards at all times
  - Call members of the Adversary class
  - Return a rule on demands; the person with the right rule gets a big bonus
- Suggestion: learn from others!

# What is Clustering?

- Given some instances with data: group instances such that
  - examples within a group are similar
  - examples in different groups are different
- These groups are clusters
- Unsupervised learning — the instances do not include a class attribute.

# Clustering Example



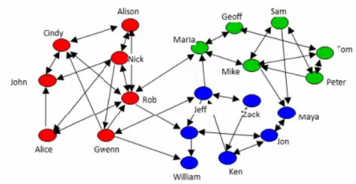
# A Different Example

- How would you group
  - 'The price of crude oil has increased significantly'
  - 'Demand for crude oil outstrips supply'
  - 'Some people do not like the flavor of olive oil'
  - 'The food was very oily'
  - 'Crude oil is in short supply'
  - 'Oil platforms extract crude oil'
  - 'Canola oil is supposed to be healthy'
  - 'Iraq has significant oil reserves'
  - 'There are different types of cooking oil'

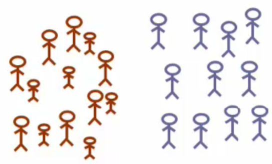
# Another Example

# Introduction


Organize computing clusters



Social network analysis


Market segmentation


Astronomical data analysis

# Clustering Basics

- Collect examples
- Compute similarity among examples **according to some metric**
- Group examples together such that
  - Examples within a cluster are similar
  - Examples in different clusters are different
- Summarize each cluster
- **Sometimes**: assign new instances to the most similar cluster

# Measures of Similarity

- In order to do clustering we need some kind of measure of similarity.
- This is basically our "critic"
- Vector of values, depends on domain:
  - documents: bag of words, linguistic features
  - purchases: cost, purchaser data, item data
  - census data: most of what is collected
- Multiple different measures available

# Measures of Similarity

- Semantic similarity (but that's hard)
- Similar attribute **counts**
  - Number of attributes with the same value.
  - Appropriate for large, sparse vectors
  - Bag-of-Words: BoW
- More complex vector comparisons:
  - Euclidian Distance
  - Cosine Similarity

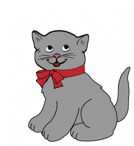
# Euclidean Distance

- Euclidean distance: distance between two measures summed across each feature
  - Squared differences to give more weight to larger difference

$$dist(x_i, x_j) = sqrt((x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+..+(x_{in}-x_{jn})^2)$$

# Euclidian

- Calculate differences
  - Ears: pointy?
  - Muzzle: how many inches long?
  - Tail: how many inches long?

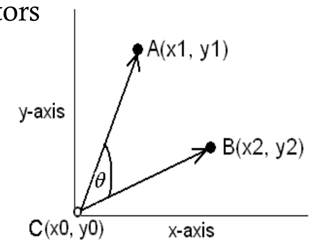
   

$dist(x_{1,} x_2) = sqrt((0-1)^2+(3-1)^2+..+(2-4)^2)=sqrt(9)=3$

$dist(x_{1,} x_3) = sqrt((0-0)^2+(3-3)^2+..+(2-3)^2)=sqrt(1)=1$

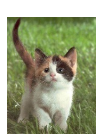
# Cosine Similarity

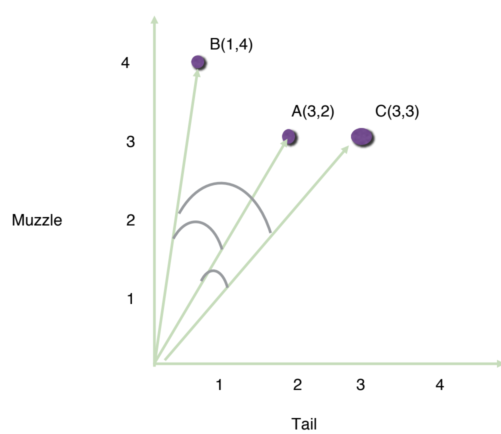
- A measure of similarity between two vectors
- Measure the **cosine of the angle** between them
- Cosine = 1 when angle = 0
- Cosine < 1 otherwise
- As angle between vectors shortens, cosine angle approaches 1
- Meaning that the two vectors are getting closer, meaning that the similarity of whatever is represented by the vectors increases




Based on home.iitk.ac.in/~mfelixor/Files/non-numeric-Clustering-seminar.ppt

# Cosine Similarity





# Clustering Algorithms

- Flat
  - K means
- Hierarchical
  - Bottom up
  - Top down (not common)
- Probabilistic
  - Expectation Maximumization (E-M)

# Partitioning (Flat) Algorithms

- Partitioning method
  - Construct a partition of n documents into a set of K clusters
- Given: a set of documents and the number K
- Find: a partition of K clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions.
  - Usually too expensive.
  - Effective heuristic methods: K-means algorithm.

http://www.csee.umbc.edu/~nicholas/676/MRSslides/lecture17-clustering.ppt

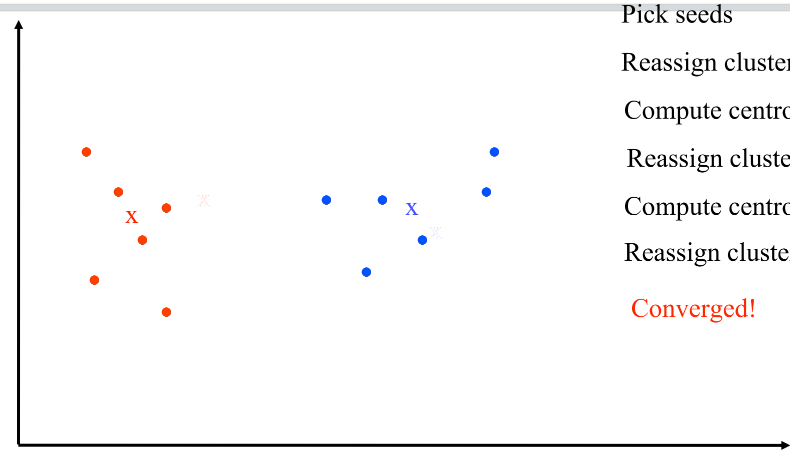
# K-Means Clustering

- Simplest hierarchical method, widely used
- Create clusters based on a centroid; each instance is assigned to the closest centroid
- K is given as a parameter
- Heuristic and iterative

# K-Means Clustering

- Provide number of desired clusters, k.
- Randomly choose k instances as seeds.
- Form initial clusters based on these seeds.
- Calculate the centroid of each cluster.
- Iterate, repeatedly reallocating instances to closest centroids and calculating the new centroids
- Stop when clustering converges or after a fixed number of iterations.

# K Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

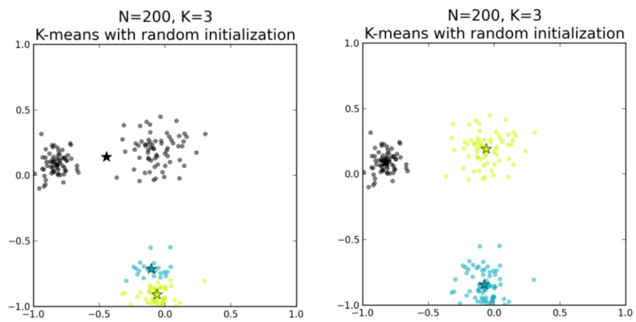
# K-Means

- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters. Overfitting again.
- Results can vary based on random seed selection.
  - Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
- The algorithm is sensitive to outliers
  - Data points that are far from other data points.
  - Could be errors in the data recording or some special data points with very different values.

http://www.csee.umbc.edu/~nicholas/676/MRSslides/lecture17-clustering.ppt

# Problem!

- Poor clusters based on initial seeds




https://datasciencelab.wordpress.com/2014/01/15/improved-seeding-for-clustering-with-k-means/

# Strengths of K-Means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: O(tkn),
    - where n is the number of data points,
    - k is the number of clusters, and
    - t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is most popular clustering algorithm.
- In practice, performs well, especially on text.

www.cs.uic.edu/~liub/teach/cs583-fall-05/CS583-unsupervised-learning.ppt

# K-Means Weaknesses

- Must choose K
  - Poor choice can lead to poor clusters
- Clusters may differ in size or density
- All attributes are weighted
- Heuristic, based on initial random seeds; clusters may differ from run to run

# Expectation Maximization (EM)

- **Probabilistic method for soft clustering**
- Assumes k clusters: $\{c_1, c_2, \ldots c_k\}$
- "Soft" version of k-means
- Assumes a probabilistic model (such as Naive Bayes) of categories
  - Allows computing $P(c_i \mid E)$ for each category, $c_i$, for a given example, E
- So basic idea is that we are learning k classifications, but starting with unlabeled data which makes this _____ learning

# EM Algorithm

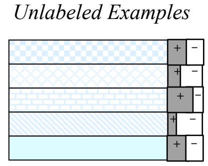
- Iteratively learn **probabilistic categorization model** from **unsupervised data**
- Initially assume random assignment of examples to categories
  - "Randomly label" data
- Learn initial probabilistic model by estimating **model parameters** $\theta$ from randomly labeled data
- Iterate until convergence:
  - **Expectation (E-step):** Compute $P(c_i \mid E)$ for each example given the current model, and probabilistically re-label the examples based on these posterior probability estimates
  - **Maximization (M-step):** Re-estimate the model parameters, $\theta$, from the probabilistically re-labeled data

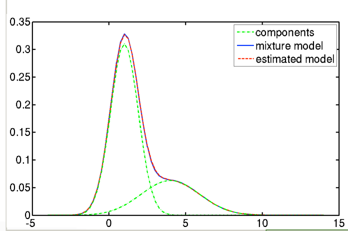
# EM

Initialize:

Assign random probabilistic labels to unlabeled data

*Unlabeled Examples*



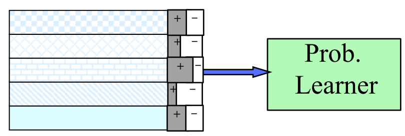



https://www.mathworks.com/matlabcentral/fileexchange/24867-gaussian-mixture-model-m

# EM

Initialize:

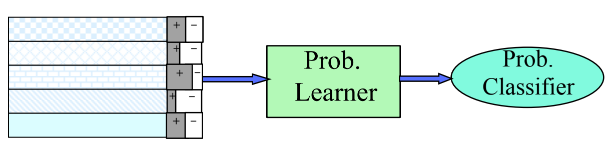
Give soft-labeled training data to a probabilistic learner




# EM

Initialize:

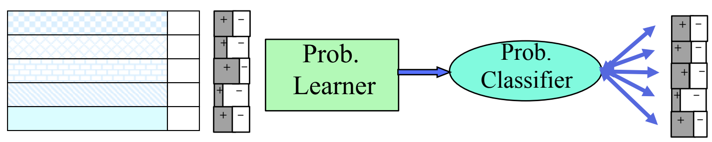
Produce a probabilistic classifier



# EM

E Step:

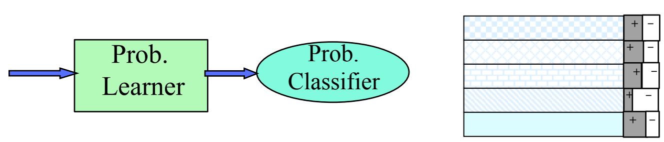
Relabel unlabled data using the trained classifier




# EM

M step:

Retrain classifier on relabeled data




**Continue EM iterations until probabilistic labels on unlabeled data converge.**

# EM Summary

- Basically a probabilistic K-Means.
- Has many of same advantages and disadvantages
  - Results are easy to understand
  - Have to choose k ahead of time
- Useful in domains where we would prefer the likelihood that an instance can belong to more than one cluster
  - Natural language processing for instance