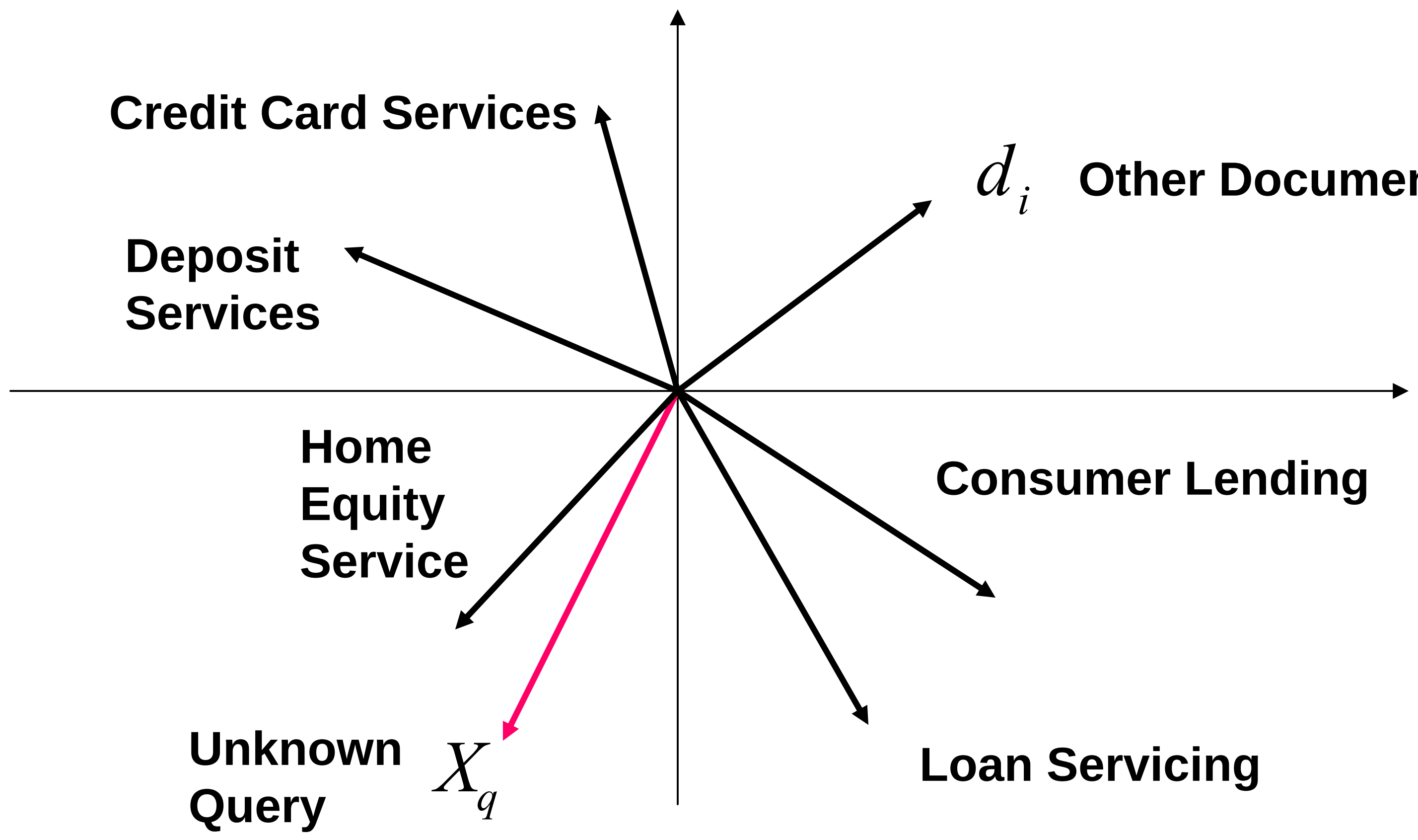
ECE8813
Statistical Natural Language Processing

Lecture 18: Clustering

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

Vector Space Representation

- **Vector distance is a key for moving from qualitative to quantitative**



Document Clustering

$$W^T W = V S^2 V^T$$

- Semantic similarity between two commands

$$K(d_i, d_j) = \cos(v_{\mathbf{i}}\mathbf{S}, v_{\mathbf{j}}\mathbf{S})$$

$$= \frac{v_i \mathbf{S}^2 \mathbf{v}^{\mathbf{T}}{}_j}{\|v_i \mathbf{S}\| \, \|v_j \mathbf{S}\|}$$

- From $W^T W$, find document clusters whose members have similarity measure exceeding a threshold (say 0.95)

Document Clustering Example

- 2000 documents into 100 clusters (one example)

- ➢ N Korea Proposes Resumed Talks with S Korea-Yonhap
- ➢ North Korea Proposes Resuming Talks with Seoul
- ➢ South Korea Set for Key Vote on Approach to North
- ➢ Korea to Replace Four to Eight Ministers on Friday
- ➢ S.Korea to Push North Policy Despite Kim Setback

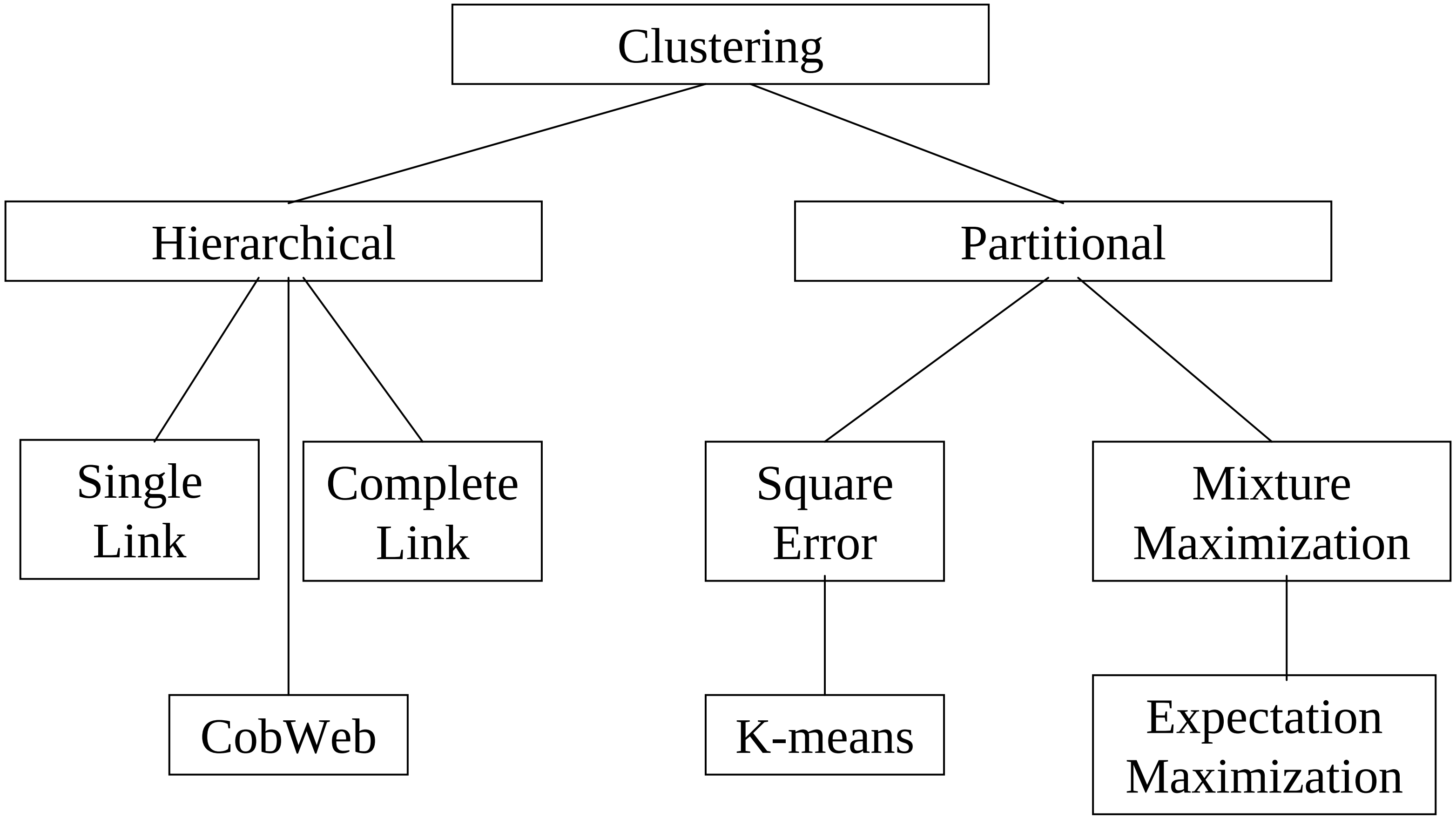
……

K-Means Term Clustering Example

- 9492 words into 100 clusters (one example)

oub bank Singapore cent uob db account share singtel trade Bangkok manage save entity annual ocbc tangible debt sti keppel custom transact currency deposit card sixth citibank integer subscribe handset creation loan auditor merger autom merge sharehold attract uncondi asx optu sembawang ibra restructur singland landlord uic yaw sgx

Clustering Techniques



Technique Characteristics

- Agglomerative vs. Divisive grouping
  - *Agglomerative*: each instance is its own cluster and the algorithm merges clusters
  - *Divisive*: begins with all instances in one cluster and divides it up
- Hard vs. Fuzzy membership
  - Hard clustering assigns each instance to one cluster whereas in fuzzy clustering assigns degree of membership

More Characteristics

- Monothetic vs Polythetic
  - *Polythetic*: all attributes are used simultaneously, e.g., to calculate distance (most algorithms)
  - *Monothetic*: attributes are considered one at a time
- Incremental vs Non-Incremental
  - With large data sets it may be necessary to consider only part of the data at a time (data mining)
  - Incremental works instance by instance

Pattern Representation

- Number of classes
- Number of available patterns
  - Circles, ellipses, squares, etc.
- Feature selection
  - Which key linguistic property?
- Feature extraction
  - Produce new features
  - e.g., principle component analysis (PCA)

Pattern Proximity

- *Want clusters of instances that are similar to each other but dissimilar to others*
- Need a similarity measure
- Continuous case
  - Euclidean measure (compact isolated clusters)
  - The squared Mahalanobis distance
    $$d_M(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)^T$$
    alleviates problems with correlation
  - Many more measures

K-means Clustering

- Suppose that we have decided how many centroids we need - denote this number by *K*
- Suppose that we have an initial estimate of suitable positions for our *K* centroids
- *K*-means clustering is an iterative procedure for moving these centroids to reduce distortion

K-means Clustering - Notation

- Suppose there are *T* data points, denoted by:

$$Y = y_1, y_2, \ldots, y_t, \ldots, y_T$$

- Suppose that the initial *K* clusters are denoted by:

$$C^0 = c_1^0, c_2^0, \ldots, c_k^0, \ldots, c_K^0$$

- One iteration of *K*-means clustering will produce a new set of clusters
- $$C^1 = c_1^1, c_2^1, \ldots, c_k^1, \ldots, c_K^1$$
- Such that

$$Dist\left(C^1\right) \leq Dist\left(C^0\right)$$

K-means Clustering (1)

- For each data point y_t let $c_{i(t)}$ be the closest centroid
- In other words: $d(y_t, c_{i(t)}) = \min_m d(y_t, c_m)$
- Now, for each centroid c^0_k define:

$$Y_k^0 = \{y_t : i(t) = k\}$$

- In other words, Y^0_k is the set of data points which are closer to c^0_k than any other cluster

K-means Clustering (2)

- Now define a new *k*th centroid c^1_k by:

$$c_k^1 = \frac{1}{\left|Y_k^0\right|} \sum_{y_t \in Y_k^0} y_t$$

  where $|Y_k^0|$ is the number of samples in Y_k^0
- In other words, c^1_k is the average value of the samples which were closest to c^0_k

K-means Clustering (3)

- Now repeat the same process starting with the new centroids (is this similar to EM):

$$C^1 = c_1^1, c_2^1, \ldots, c_k^1, \ldots, c_K^1$$

to create a new set of centroids:

$$C^2 = c_1^2, c_2^2, \ldots, c_k^2, \ldots, c_K^2$$

… and so on until the process converges

- Each new set of centroids has smaller distortion than the previous set

So….Basically

- Start with randomly *k* data points (objects).
- Find the set of data points that are closer to C^0_k (Y^0_k).
- Compute average of these points, notate C^1_k -> new centroid.
- Now repeat again this process and find the closest objects to C^1_k
- Compute the average to get C^2_k -> new centroid, and so on….
- Until convergence.

Comments on *K-Means*

- Strength
  - *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Soft K-means

- Instead of making hard assignments of data points to clusters, we can make soft assignments. One cluster may have a responsibility of .7 for a data point and another may have a responsibility of .3
  - Allows a cluster to use more information about the data in the refitting step.
  - What happens to our convergence guarantee?
  - How do we decide on the soft assignments?

A Generative View of Clustering

- We need a sensible measure of what it means to cluster the data well
  - This makes it possible to judge different methods
  - It may make it possible to decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we can adjust the parameters of the model to maximize the probability density that it would produce exactly the data we observed

Generating Gaussians Mixture Data

- Gaussian Mixture Model (GMM) for non-Gaussian data
- First pick one of the *k* Gaussians with a probability that is called its “mixing proportion”
- Then generate a random point from the chosen Gaussian with a specific combination of mean and variance
- The probability of generating the exact data we observed is zero, but we can still try to approximate the density by
  - Adjusting the means of the Gaussians
  - Adjusting the variances of the Gaussians on each dimension
  - Adjusting the mixing proportions of the Gaussians

$$GMM(x) = \sum_{m=1}^{M} \pi_m N(x; \mu_m, \sigma_m^2), \quad \sum_{m=1}^{M} \pi_m = 1, \quad 0 < \pi_m < 1, \quad \sigma_i > 0$$

E-step: Computing Probabilities

- In order to adjust the parameters, we must first solve the inference problem: Which Gaussian generated each point?
  - We cannot be sure, so it's a distribution over all possibilities
- Use Bayes theorem to get posterior probabilities

Posterior for Gaussian i

Prior for Gaussian i

$$p(i \mid \mathbf{x}^c) = \frac{p(i)\,p(\mathbf{x}^c \mid i)}{p(\mathbf{x}^c)}$$

Bayes theorem

$$p(\mathbf{x}^c) = \sum_j p(j)\,p(\mathbf{x}^c \mid j)$$

$$p(i) = \pi_i$$

Mixing proportion

$$p(\mathbf{x}^c \mid i) = \prod_{d=1}^{d=D} \frac{1}{\sqrt{2\pi}\sigma_{i,d}} e^{-\frac{\| x_d^c - \mu_{i,d} \|^2}{2\sigma_{i,d}^2}}$$

Product over all data dimensions

M-step: Computing Mixing Proportions

- Each Gaussian gets a certain amount of posterior probability for each data point
- The optimal mixing proportion to use (given these posterior probabilities) is just the fraction of the data that the Gaussian gets responsibility for.

Posterior for Gaussian i

Data for training case c

$$\pi_i^{new} = \frac{\sum_{c=1}^{c=N} p(i \mid \mathbf{x}^c)}{N}$$

Number of training cases

M-step: Computing New Means

- We just take the center-of gravity of the data that the Gaussian is responsible for
  - Just like in K-means, except the data is weighted by the posterior probability of the Gaussian
  - Guaranteed to lie in the convex hull of the data
    - Could be big initial jump

$$\boldsymbol{\mu}_i^{new} = \frac{\sum_c p(i \mid \mathbf{x}^c)\, \mathbf{x}^c}{\sum_c p(i \mid \mathbf{x}^c)}$$

M-step: Computing New Variances

- We fit the variance of each Gaussian *i*, on each dimension *d*, to the posterior-weighted data
  - Its more complicated if we use a full-covariance Gaussian that is not aligned with the axes.

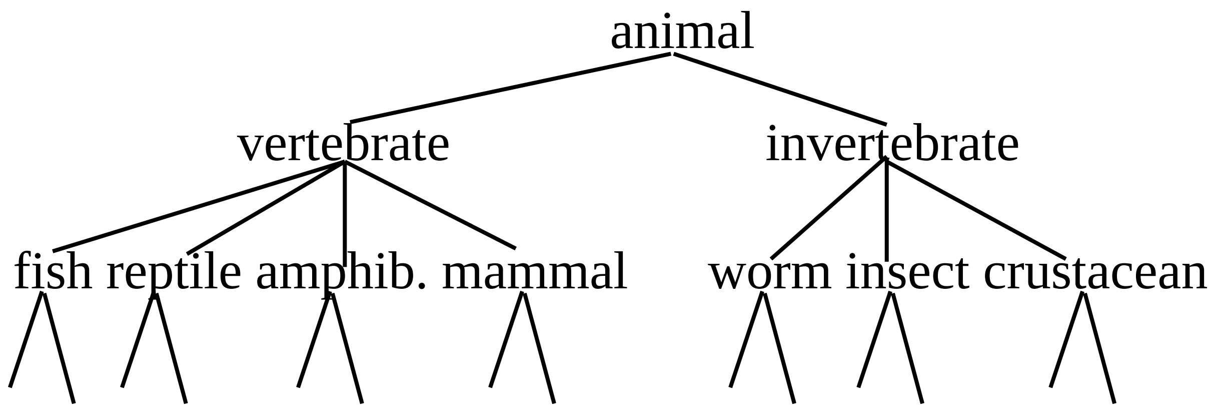
$$\sigma_{i,d}^{2} = \frac{\sum_{c} p(i \mid \mathbf{x}^{c}) \; \| x_{d}^{c} - \mu_{i,d}^{new} \|^{2}}{\sum_{c} p(i \mid \mathbf{x}^{c})}$$

“The Curse of Dimensionality”

- Why document clustering is difficult
  - While clustering looks intuitive in 2 dimensions, many of our applications involve 10,000 or more dimensions…
  - High-dimensional spaces look different: the probability of random points being close drops quickly as the dimensionality grows
  - One way to look at it: in large-dimension spaces, random vectors are almost all almost perpendicular. Why?

Hierarchical Clustering

- Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of unlabeled examples



- One option to produce a hierarchical clustering is recursive application of a partition clustering algorithm to produce a hierarchical clustering

Hierarchical Clustering Algorithms

- **Agglomerative (bottom-up):**
  - Starting with each document being a single cluster
  - Eventually all documents belong to the same cluster
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster
  - Eventually each node forms a cluster on its own
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both agglomerative and divisive is of no use

Hierarchical Agglomerative Clustering

- HAC: Assuming a goodness-of-fit function for determining the similarity of two instances
- Starting with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster

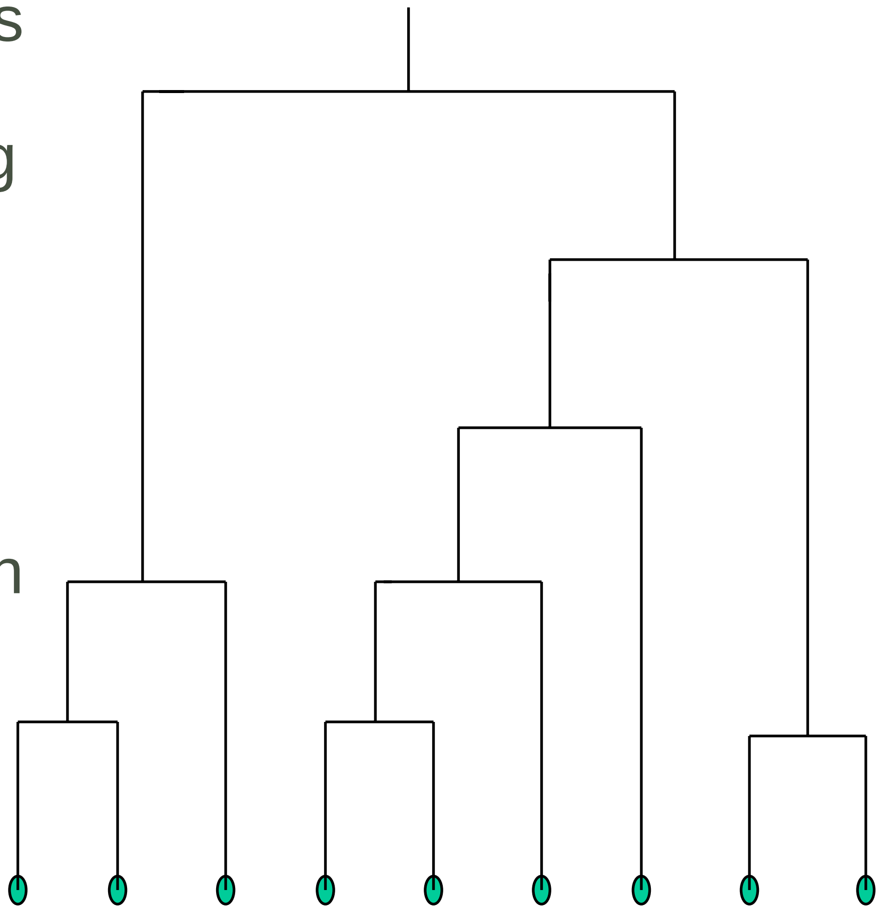
```
Among the current clusters,
    determine the two clusters, $ci$ and $cj$, that are most similar
Replace $ci$ and $cj$ with a single cluster $ci \cup cj$
```

- The history of merging forms a binary tree or hierarchy

A Dendrogram: Hierarchical Clustering

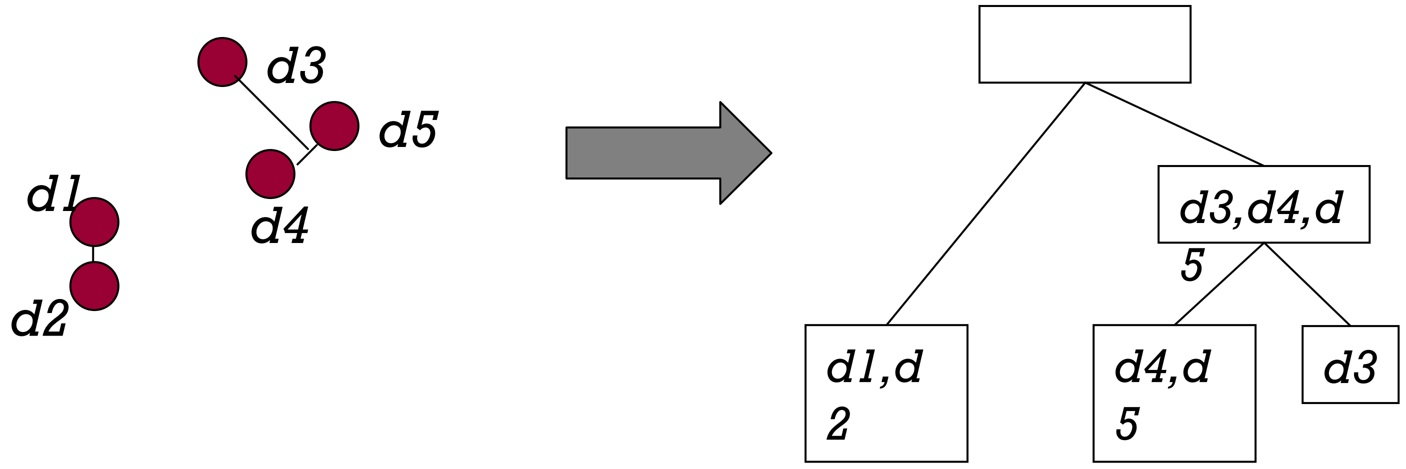
- Dendrogram: Decomposes data objects into a several levels of nested partitioning (tree of clusters).

- Clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



Dendrogram: Document Example

- As clusters *agglomerate*, docs likely to fall into a hierarchy of "topics" or concepts



“Closest Pair” of Clusters

- Many variants to defining closest pair of clusters
- “Center of gravity”
  - Clusters whose centroids (centers of gravity) are the most cosine-similar
- Average-link
  - Average cosine between pairs of elements
- Single-link
  - Similarity of the most cosine-similar (single-link)
- Complete-link
  - Similarity of the “furthest” points, the least cosine-similar

Key Concerns with HAC

- Key problem: as clusters are being formed, how to represent the location of each cluster, to tell which pair of clusters is closest?
- Euclidean case: each cluster has a *centroid* = average of its points
  - Measure inter-cluster distances by distances of centroids

Single Link Agglomerative Clustering

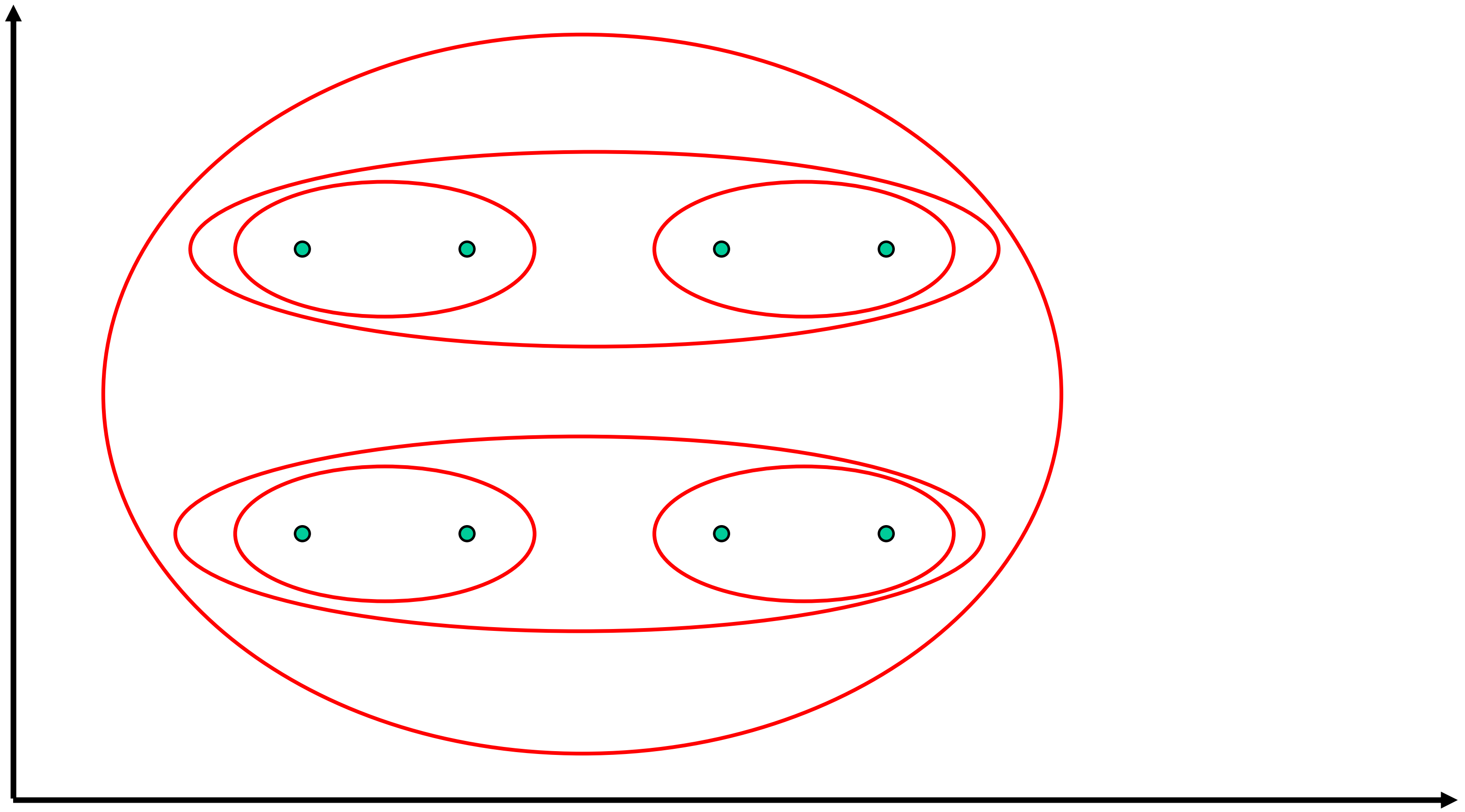
- Use maximum similarity of pairs:

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

- Can result in “straggly” (long and thin) clusters due to chaining effect.
- After merging *c*$_i$ and *c*$_j$, the similarity of the resulting cluster to another cluster, *c*$_k$, is:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

A Single Link Example



Complete Link Agglomerative Clustering

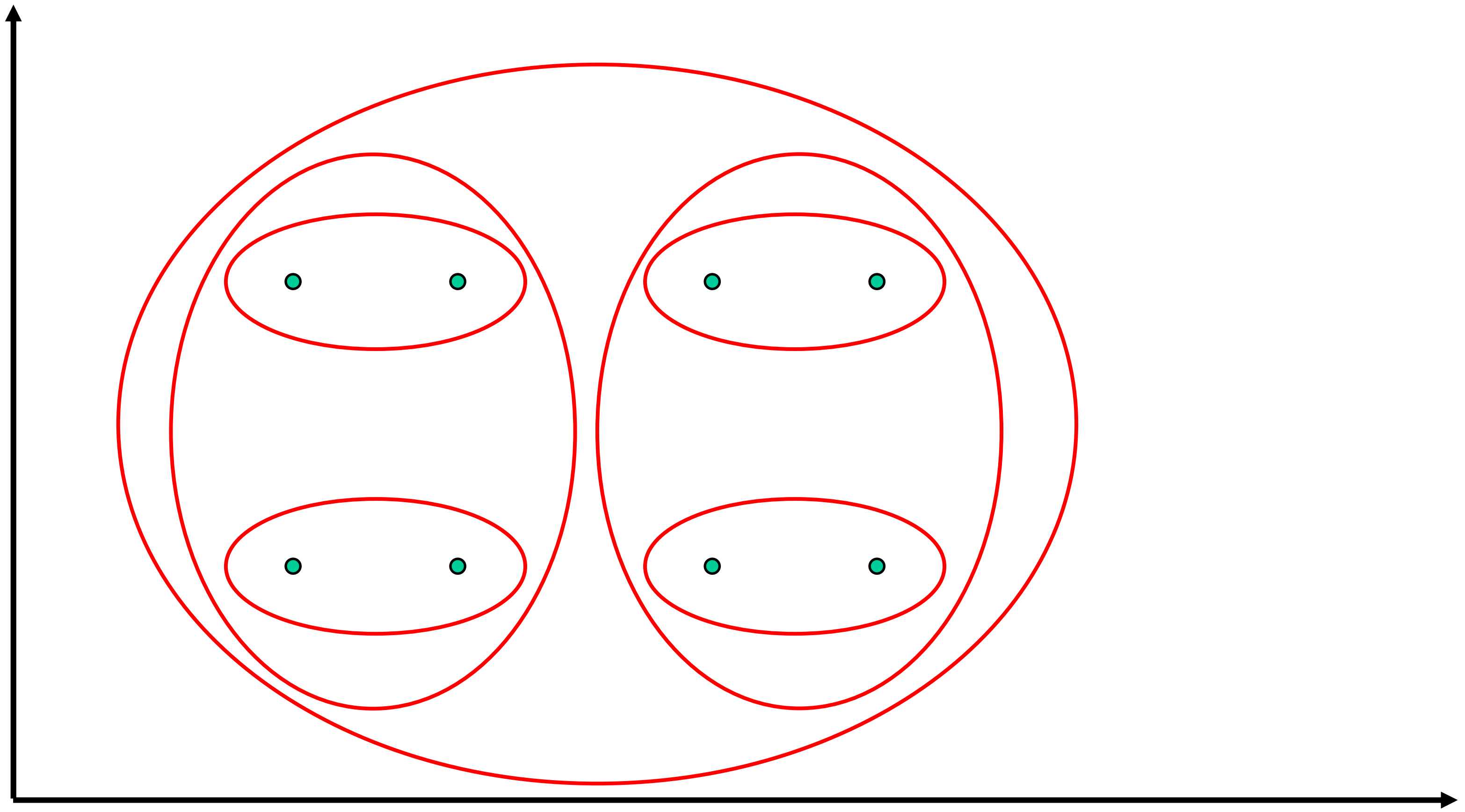
- Use minimum similarity of pairs:

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.
- After merging c_i and c_j, the similarity of the resulting cluster to another cluster, c_k, is:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

A Complete Link Example

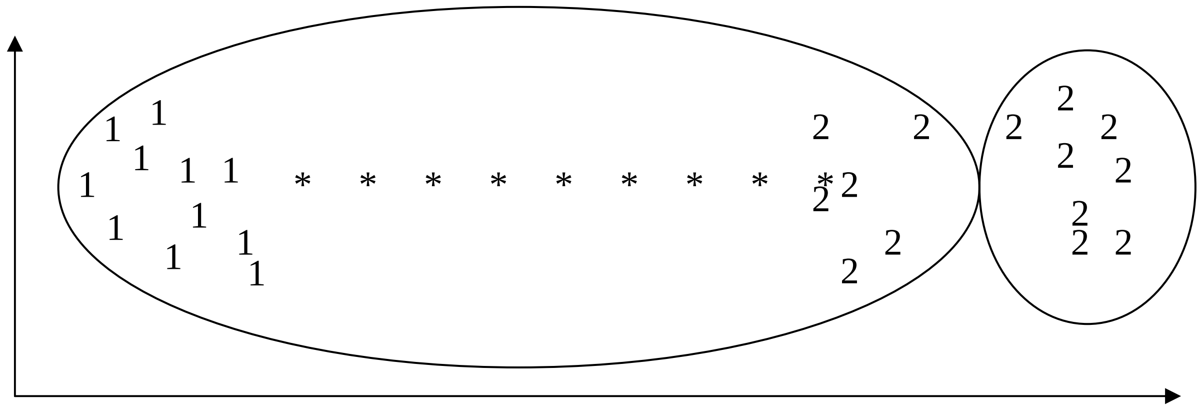


Summary: Hierarchical Algorithms

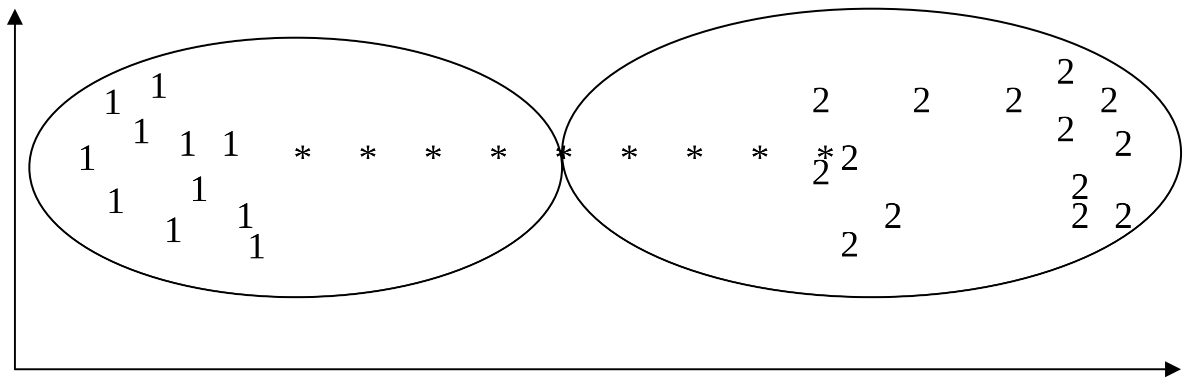
- Single-link
  - Distance between two clusters set equal to the *minimum* of distances between all instances
  - More versatile
  - Produces (sometimes too) elongated clusters
- Complete-link
  - Distance between two clusters set equal to *maximum* of all distances between instances in the clusters
  - Tightly bound, compact clusters
  - Often more useful in practice

Example: Clusters Found

Single-Link



Complete-Link



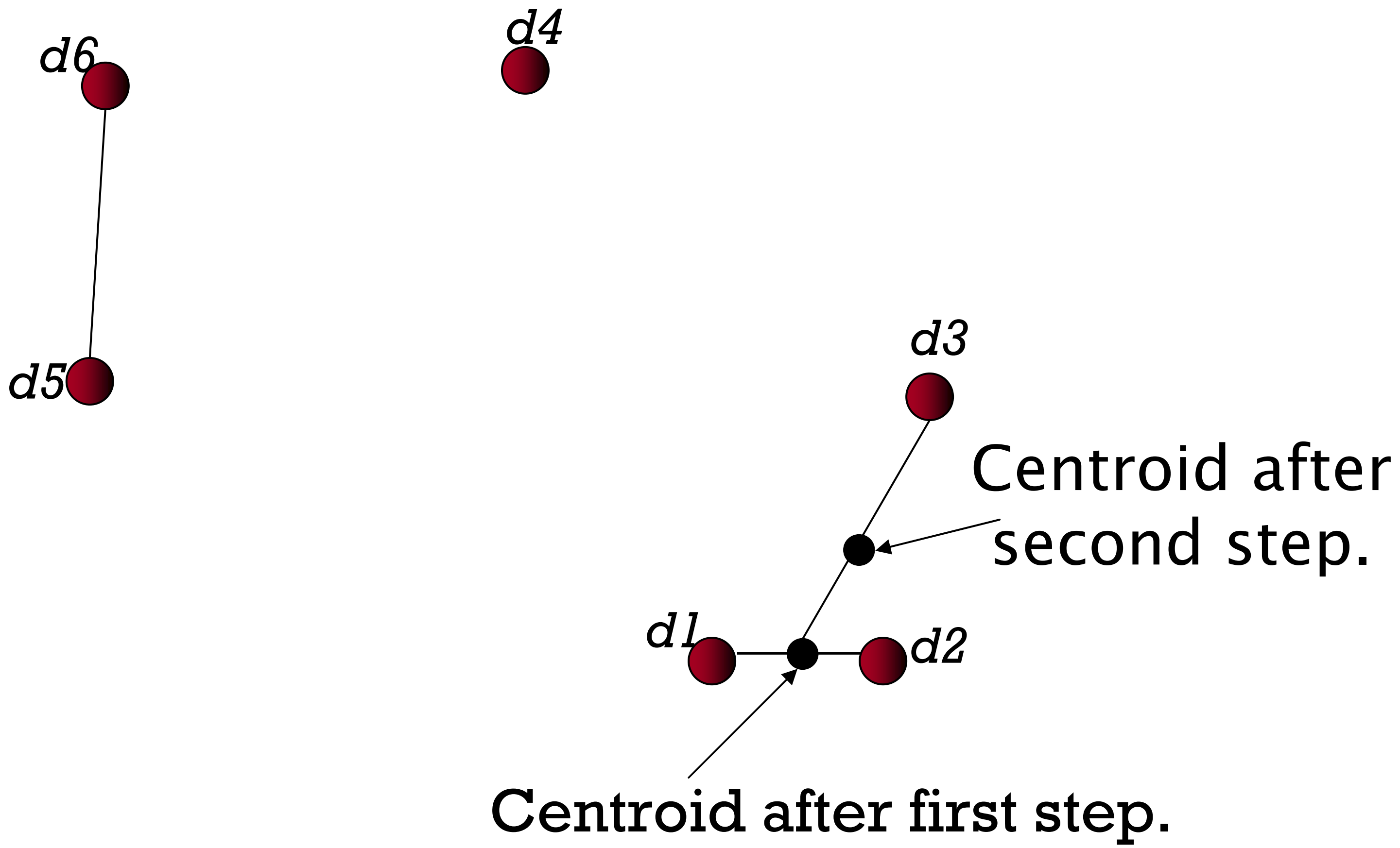
Computational Complexity

- In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- In each of the subsequent (n–2) merging iterations, it must compute the distance between the most recently created cluster and all other existing clusters.
  - Since we can just store unchanged similarities
- In order to maintain an overall $O(n^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Else $O(n^2 \log n)$ or $O(n^3)$ if done naively

Key Notion: *Cluster Representative*

- We want a notion of a representative point in a cluster
- Representative should be some sort of "typical" or central point in the cluster, e.g.,
  - point inducing smallest radii to docs in cluster
  - smallest squared distances, etc.
  - point that is the "average" of all docs in the cluster
    - Centroid or center of gravity

Example: n=6, k=3, Closest Pair of Centroids



Outliers in Centroid Computation

- Can ignore outliers when computing centroid.
- What is an outlier?
  - Lots of statistical definitions, e.g.
  - *moment* of point to centroid > $M \times$ some cluster *moment*

$\uparrow$ Say 10.

Centroid
Outlier

Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters

$$sim(c_i, c_j) = \frac{1}{\left|c_i \cup c_j\right|\left(\left|c_i \cup c_j\right| - 1\right)} \sum_{\vec{x} \in (c_i \cup c_j)} \sum_{\vec{y} \in (c_i \cup c_j) : \vec{y} \neq \vec{x}} sim(\vec{x}, \vec{y})$$

- Compromise between single and complete link
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters
- Some previous work has used one of these options; some the other. No clear difference in efficacy

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length.
- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

- Compute similarity of clusters in constant time:

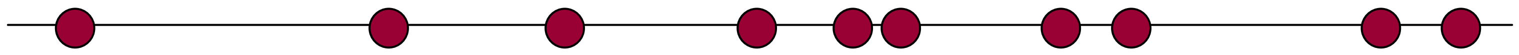
$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

Efficiency: Medoid As Cluster Center

- The centroid does not have to be a document
- Medoid: A cluster representative that is one of the documents (not the centroid of the cluster)
- Example: the document closest to the centroid
- One reason this is useful
  - Consider the representative of a large cluster (>1000 documents)
  - The centroid of this cluster will be a dense vector
  - The medoid of this cluster will be a sparse vector
- Compare: mean/centroid vs. median/medoid

Exercise

- Consider agglomerative clustering on *n* points on a line. Explain how you could avoid n^3 distance computations - how many will your scheme use?
- An optimal scheme can be worked (a good lead to scalar quantization)
- How extension to vector quantization?



Resources

- Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections (1992)
  - Cutting/Karger/Pedersen/Tukey:

  http://citeseer.ist.psu.edu/cutting92scattergather.html
- Data Clustering: A Review (1999)
  - Jain/Murty/Flynn: http://citeseer.ist.psu.edu/jain99data.html
- A Comparison of Document Clustering Techniques
  - Michael Steinbach, George Karypis and Vipin Kumar. TextMining Workshop. KDD. 2000
- Initialization of iterative refinement clustering algorithms. (1998)
  - Fayyad, Reina, and Bradley: http://citeseer.ist.psu.edu/fayyad98initialization.html
- Scaling Clustering Algorithms to Large Databases (1998)
  - Bradley, Fayyad, and Rein: http://citeseer.ist.psu.edu/bradley98scaling.html

CobWeb (in Weka)

- Algorithm (main) characteristics:
  - Hierarchical and incremental
  - Uses **category utility**

The k clusters

$$CU(C_1, C_2, \ldots, C_k) = \frac{\sum_l \Pr[C_l] \sum_i \sum_j \overbrace{\left(\Pr[a_i = v_{ij} \mid C_l]^2 - \Pr[a_i = v_{ij}]^2\right)}^{\text{Improvement in probability estimate because of instance cluster assigment}}}{k}$$

All possible values for attribute a_i

Category Utility

- If each instance in its own cluster

$$\Pr\left[a_i = v_{ij} \mid C_l\right] = \begin{cases} 1 & v_{ij} = \text{actual value of instance} \\ 0 & \text{otherwise} \end{cases}$$

- Category utility function becomes

$$CU\left(C_1, C_2, \ldots, C_k\right) = \frac{n - \sum_i \sum_j \Pr\left[a_i = v_{ij}\right]^2}{k}$$

- Without *k* it would always be best for each instance to have its own cluster, **overfitting**!

Summary

- Today's Class
  - Unsupervised Clustering
- Next Classes
  - Quiz on 3/12 (3 problems), Spring Break after that
  - More clustering on 3/24
  - Text Categorization on 3/26
  - Labs 4 (tagging) and 5 (clustering) due after break
- Reading Assignments
  - Manning and Schutze, Chapters 14-16