Clustering

1. **Each dimension** corresponds to a **feature/attribute**

2. Why Clustering Hard (1) Number of clusters is unknown Arbitrary shapes and sizes (3) Quality of clustering result: a. depends on the distance measures b. measured by the ability to find hidden patterns (4) applications in high dimensional space **3. Application:** Sloan Digital Sky Survey & Music CDs& Cluster Documents (Find topics) **4. Distance Measure:** Non-negativity: d(x,y) ≥ 0, Identity: d(x,y)=0 if and only if x=y, symmetry, d(x,y) = d(y,x), triangle inequality: d(x,y) ≤d(x,z) + d(z,y) **5. Examples:** **Lp-norm**:

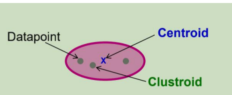
$$d(\mathbf{x},\mathbf{y}) = \left(\sum_i (x_i - y_i)^p\right)^{\frac{1}{p}}$$

L2-norm=Euclidean Distance

$d(x,y) = sqrt((x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2)$

L1-norm: Manhattan Distance $d(x,y) = |x_1 - y_1| + |x_2 - y_2| + \ldots +$

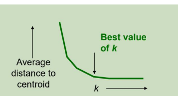
Cosine: $cosine(\mathbf{x},\mathbf{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$ **Jaccard:** $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ **Edit Distance:** distance that convert one string to another **Hierarchical Clustering:** repeatedly combine two nearest clusters **How to represent Cluster?** (1) Euclidean space: centroid = average of its data points (2) Non-Euclidean Space: Clustroid= point "closest" (smallest maximum/average/sum of squares distance) to other points



$$\min_c \sum_{x \in C} d(x,c)^2$$

Intercluster distance = minimum of distances between any two points, one from each clusters **Diameter**: The maximum distance of two points in the cluster **Aver distance** between points in the cluster **Density-based**: Diameter or Average distance / the number of points in the cluster **When to Terminate** (1) Pre-determined number of clusters (2) merging two clusters -> 'bad' cluster (diameter of merged cluster > threshold, diameter > average diameter by a wide margin / Density of clusters < threshold)

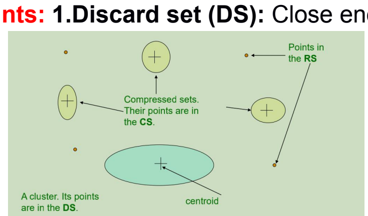
Complexity of hierarchical clustering: Naive: O(n3)/Priority Queue: O(n2logn) **K-means Algorithm:** Process: 1.place points to nearest clusters (centroid) 2. update centroid of k clusters 3. reassign points to closest centroid 4. repeat 2 and 3 util convergence **Select K:**



Complexity-K-means:

- Complexity is O(n*k*m)
  - n = number of points
  - k = number of clusters
  - m = number of iterations

BFR Algorithm: Variant of k-means to handle very large data sets in high dimensions./ points in clusters are normally distributed around a centroid in a Euclidean space **Memory** O(clusters) 1. Points are read from disk to memory in chunk 2. Process: (1) Select initial k centroids from the first chunk (Take k random points / Take a small random sample and cluster optimally / Pick a random point, and then k-1 more points (each as far from the previous points as possible) **3. Points:** **1.Discard set (DS):** Close enough to a centroid are summarized **2. Compression set (CS):** Summarized, but not assigned to a cluster **3.Retained set (RS):**



Isolated points **Summarizing Points** N(number of points) SUM(sum of points in ith dimension) SUMSQ(sum of squares of coordinates in ith dimension) **2d(number of dimensions)+1** -> any size cluster Centroid (Average in each dimension) SUMi/N Variance (SUMSQi/N) -(SUMi/N)^2

Mahalanobis Distance

- For a point $(x_1, \ldots, x_n)$ and centroid $(c_1, \ldots, c_n)$
  - $d(x,c) = \sqrt{\sum_{i=1}^{n}\left(\frac{x_i - c_i}{\sigma_i}\right)^2}$ Normalized Euclidea from Centroid

1. Compute MD (point & cluster centroid) 2. Choose cluster with least MD 3.Add point to cluster (MD < threshold)

Combine 2 CS into one: 1. Compute the variance of combined cluster 2. N, SUM and SUMSQ make calculation quickly 3. Combine if the combined variance < threshold

CURE Algorithm 1. Difference: (1) BFR and k-means assum cluster are normally distributed in each dimension [Axes are fixed, Ellipses at an angle are not OK] (2) CURE assumes Euclidean distance & allow clusters of any shape & Use a collection of representative points to represent clusters] **2. Process:** (1) Pass 1: a. Pick a random sample of points that fit in main memory (2) Cluster these points hierarchically(group nearest points/clusters) (3) Pick representative points: for each cluster, pick a sample of points, as dispersed as possible / For the sample, pick representatives by moving them 20% toward the centroid of the cluster (2) Pass 2: 1. Rescan the whole dataset and visit each point p in the dataset 2. Place it in the "closest cluster"

Recommender System

1. **Long tail:** a near-limitless selection **2. Types of Recommendations:** a. Editorial (list of favorites / essential items) b.Simple aggregates (Top10, Most popular, Recent Uploads) c.Personalized **3.Utility Function**: C x S -> R [C= sets of customers/ S = set of items / R = set of ratings] **4. Key Challenges:** a.Gathering "known" ratings for matrix b. Extrapolating unknown ratings from known ones c. Evaluating extrapolation methods (how to measure the (success/performance) of recommendation methods) 5. **Extrapolating Utilities**: a.Utility matrix U is sparse [no ratings / cold start] b. Recommender system seen as a function (1) Given -> User model (i.e. Ratings, preferences, demographics 人口统计) & Items (with or without description of item characteristics) (2) Find Relevance Score **Content-Based Recommender Systems 1.Main idea:** recommend to customer(C) **items** that are similar to previous **items** rated highly by C **2. Examples:**Movie recommendations / web, blogs, news **3.Item Profiles:** Profile is a set of features (value-> boolean or real-valued), use TF-IDF to pick important features **4.User Profiles:**a.vector to desribe user preferences b.Predict preference of x for items that has not rated

- Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate $utility(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

5. Pros: 1. No need for data on other users 2.Able to recommend to users with unique tastes 3. Able to recommend new & unpopular items 4. Able to provide explanations **6.Cons:** 1. Finding the appropriate features is hard 2. Cold start problem for new users 3.Overspecialization [Never recommend items outside user's content profile / People might have multiple interests] 4.Unable to exploit quality judgements of other users. **Collaborative Filtering**

1.Main idea: Estimate x's ratings based on the ratings of users in N whose ratings are 'similar' to x's ratings **2. Similar Users:** a. Jaccard similarity measure b. Cosine similarity c.Pearson correlation coefficient

$\mathrm{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r_x}, \boldsymbol{r_y}) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$

- S_{xy} = items rated by both users x and y

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

2. Rating Predictions

- Use average of their ratings $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$

weighted measures $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$ $\quad s_{xy} = sim(x, y)$

3.Collaborative Filtering: (1) Pros: Work for any kind of item (no feature selection needed) **(2) Cons:** 1. Code Start (Need enough users in the system to find a match) 2. Sparsity (Hard to find users that have rated the same items) 3. First rater (Cannot recommend an unrated item) 4. Popularity bias (tend to recommend popular item)

4.Hybrid Recommenders: Add contnet-based method to CF

- Root-mean-square error (RMSE)

users

$$\sqrt{\sum_{xi} (r_{xi} - r_{xi}^*)^2}$$

r_{xi} is predicted
r_{xi}^* is the true rating of x on i

5. Evaluation Predictions:

6.complexity: Find k most similar customers -> O(k*|C|) **Solutions:** 1. Near-neighbor search in high dimensions (LSH) 2. Clustering 3. Dimensionality reduction **7.Tips: Add Data** (Leverage all data)

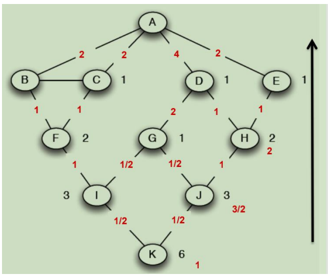
Analyzing Large Graphs

1.Community Detection:a.inter-group-sparse b.intra-group-dense

2.Edge Betweeness: a.how important an edge b.number of shortest path through an edge c.inter-community edge has higher betweenes

3.**Girvan-Newman Algorithm:** (1)Breadth first search of graph start from one node (2)Top down: root A=1, next label is sum of parent. (3) Bottom up: [Credit (Z) * label (Pi) divided by sum of labels of P1..,.Pk] [1 + credit on edge below it] (4) Repeat the calculation for every node

$$B(a,b) = \sum_{x,y \in V} \frac{|SP(x,y)\ that\ include\ (a,b)|}{|SP(x,y)|}$$



as the root and sum contribution (5) Divide by 2 to get final edge betweene

4.**Scalability issues:** m edges and n nodes-> Final betweenness score for every edge O(mn) & Recalculation takes O(m^2n)

5.Scaling up G-N Algorithm: 1.Determine which edges need recalculate betweenness when an edge is removed. (When e is removed, betweenness (e') needs to be recalculated only if e' is in the same connected component as e 2.Not much pruning if a component is large. **6.Link Analysis:** (1)Types of links: Referential -Click here and get back home (2) Informational-Click here to get more detail **7.PageRank:** Links as votes (A page is more important if it has more links) **8.Matrix Formulation:** (1) M (2) Rank vector r -> ri is the important score of page i. (3) Flow equation: r = M *r

9. Power iteration Method:

Column → out.
row → in

	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

- **Power iteration – simple iterative scheme**
  - Suppose there are N web pages
  - Initialize $\mathbf{r}^{(0)}$ = [1/N, …., 1/N]^T transpose
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
- **Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$**
  - $|\mathbf{x}|_1 = \sum_{i \in [1,N]} |x_i|$ is the L_1 norm
  - Can use any other vector norm e.g., Euclidean

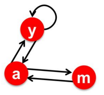
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

d_i …. out-degree of node i

10.Dead ends: A page has not out-links -> cause all pages to 0

11.Solution: Teleport -> Allow a surfer to jump to some random page from dead ends

- **Power Iteration:**
  Set $r_j = 1/N$
  1: $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  2: $r = r'$
  Goto 1



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2 + r_m$
$r_m = r_a/2$

- **Example:**

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ \\ \\ \end{bmatrix} \begin{bmatrix} 9/24 \\ \\ \\ \end{bmatrix} \quad 6/15$$

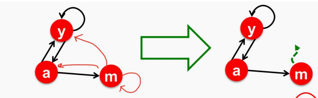
$r^{(0)}$ $r^{(1)}$ $\frac{1}{3}\times\frac{1}{2}+\frac{1}{3}\times\frac{1}{2}$ $\frac{1}{3}\times\frac{1}{2}+\frac{1}{2}\times\frac{1}{2}=\frac{5}{12}$ $\frac{1}{3}\times\frac{1}{2}+\frac{1}{3}\times 1$ $\frac{1}{3}\times\frac{1}{2}$ $\frac{1}{3}\times\frac{1}{2}+\frac{1}{6}$ $\frac{1}{2}\times\frac{1}{2}$ $\frac{5}{12}\times\frac{1}{2}+\frac{2}{6}\times\frac{1}{2}$

Iteration 0, 1, 2, …

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/6 \\ 1/6 \end{bmatrix} \begin{bmatrix} 1/4 \\ 1/6 \\ 1/12 \end{bmatrix} \begin{bmatrix} 5/24 \\ 1/8 \\ 1/12 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Iteration 0, 1, 2, …

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix} \begin{bmatrix} 1/4 \\ 1/6 \\ 7/12 \end{bmatrix} \begin{bmatrix} 5/24 \\ 1/8 \\ 2/3 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

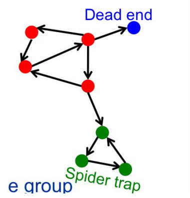
M teleport **$[1/N]_{NxN}$**

$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

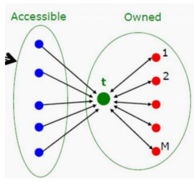
12.Problem: Spider Traps: A group of pages with no links out of the group **13.Solution:Teleport**

- **Measures generic popularity of a page**
  - Ignore or miss topic-specific authorities
  - **Solution:** Topic-specific PageRank
- **Uses a single measure of importance**
  - Other models of importance
  - **Solution:** Hubs-and-Authorities
- **Susceptible to link spam**
  - Artificial link topologies created in order to boost page rank
  - **Solution:** TrustRank

Limitations of PageRank:



15.Topic-Specific PageRank: Teleport set is restricted to a topic specific set: a.Decide on topics b.Pick a teleport set for that topics c. determine the topic (most relevant to a query) d. Use the pageRank vectors **16.TrustRank:(1) Link Farms:** Maximize the pagerank of page t **(2)TrustRank:** Topic-specific PageRank with a teleport set of trusted pages / Approximation isolation



$$y = x + \beta M \left[\frac{\beta y}{M} + \frac{1-\beta}{N}\right] + \frac{1-\beta}{N} \quad y = \frac{x}{1-\beta^2} + c\frac{M}{N} \quad \text{where } c = \frac{\beta}{1+\beta}$$

(3) Seed pages: make small and point to adequate trust rank

17.Trust Propogation: Each page has trust value (0-1), spam: page below trust threshold.

$\beta\, t_p / |o_p| \quad \text{for } 0 < \beta < 1$

Trust is additive: trust of p = sum of inlinked pages **Trust is attenuation:**decreases with the distance **Trust is splitting:**split by outlinks **18.Hubs and Authorities:1.HITS(Hypertext-induced Topic Selection) 2.Authority:**contains useful information **3.Hub:**link to authorities

- **HITS algorithm in vector notation**
  Set: $\boldsymbol{a_i} = \boldsymbol{h_i} = \frac{1}{\sqrt{n}}$
- **Repeat until convergence**
  $\boldsymbol{h} = \boldsymbol{A} \cdot \boldsymbol{a}$
  $\boldsymbol{a} = \boldsymbol{A^T} \cdot \boldsymbol{h}$
  Normalize $\boldsymbol{a}$ and $\boldsymbol{h}$
- **Now** $\boldsymbol{a} = \boldsymbol{A^T} \cdot \underbrace{(\underbrace{\boldsymbol{A} \cdot \boldsymbol{a}}_{\text{new } h})}_{\text{new } a}$

Convergence criterion:
$$\sum_i \left(h_i^{(t)} - h_i^{(t-1)}\right)^2 < \varepsilon$$
$$\sum_i \left(a_i^{(t)} - a_i^{(t-1)}\right)^2 < \varepsilon$$

a is updated (in 2 steps):
① $a = A^T (A\, a) = (A^T A)\, a$
h is updated (in 2 steps):
② $h = A\, (A^T h) = (A\, A^T)\, h$

4.PageRank vs HITS:1.both us link structure to rank page 2.PageRank computed for all web pages and stored prior to query / HITS operates on a small subgraph from web graph 3.PageRank model: the value of the link depends on the links into u / HITS model: depends on the value of the other links out of u (vulnerable to spam)