EECS E6690: SL for Bio & Info
Lecture 11: Dimensionality Reduction, Information Ranking, Association Analysis, and Social Networks

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm
1127 Seeley W. Mudd Building

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/∼predrag

Unsupervised Learning

There is no input-output relationship, $y = f(x)$, i.e. there is no **y**.

- We only have a bunch of points $(x_1, x_2, \ldots, x_n)$
- The problem has less structure
- We are trying to discover a structure - maybe more interesting

Typical questions and approaches

- Clustering - grouping data points
- Principal Component Analysis (PCA): used of preprocessing and visualization
- Ranking: e.g., Google's PageRank algorithm
- Association Rules - Market Basket Analysis
  discovering relationships between data points
- Community Detection or Graph Clustering: e.g., discovering communities on Facebook

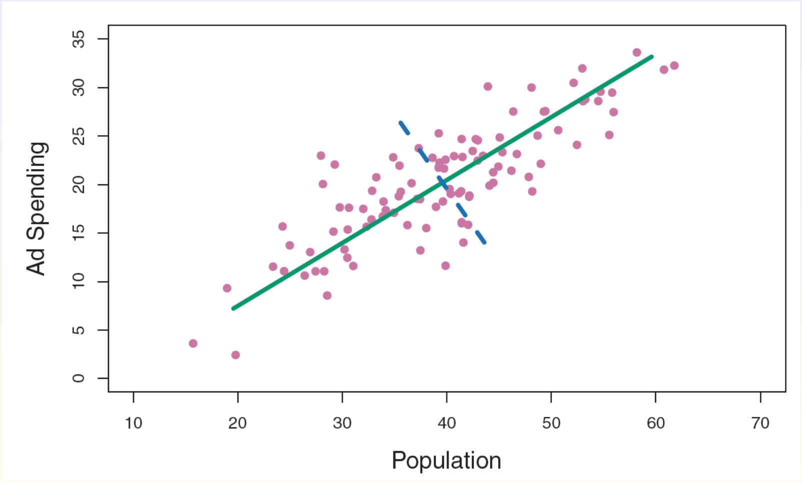
Last lecture: Principal Components Analysis (PCA)

Principal Components Analysis - unsupervised approach (no use of Y)

- Finding directions along which data is located - highly variable
- These directions define lines/subspaces that are close to the "data cloud"
- Can be used for visualization/understanding data
- Can be used as preprocessing for supervised learning

PCA: Example

Two PC-s: Solid line: First PC; Dashed line: Second PC



PCA: Good for High Dimensional Data - Large p

- Dimensionality reduction - e.g. gene expression data
- Assume we compute M principal components
- The best M-dimensional approximation to x_{ij}

$$x_{ij} = \sum_{m=1}^{M} z_{im}\phi_{jm}$$

How good is PC approximation?

- Proportion of Variance Explained (PVE)
- For each component, m, PVE is equal to

$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{ij} x_{ij}^2}$$

PCA is an Eigen-Decomposition Problem

- In general finding k-principal components is equivalent to finding a $k \times p$ matrix such that $$\boldsymbol{z}_i = W\boldsymbol{x}_i$$
- We also want linear recovery, i.e., to find a matrix U, such that $$\hat{\boldsymbol{x}}_i = U\boldsymbol{z}_i = UW\boldsymbol{x}_i \approx \boldsymbol{x}_i$$
- Hence, this reduces to an optimization problem $$\arg\min_{W,U} \sum_{i=1}^{n} \|\boldsymbol{x}_i - UW\boldsymbol{x}_i\|^2 \tag{1}$$

Theorem Let $A = \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^\top$ and let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ be the k leading eigenvectors of A. Then, the solution to the PCA problem is to set the columns of U to be $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ and to set $W = U^\top$.

Efficient Solution for High-Dimensional Data: $p \gg n$

The complexity of the previously described PCA solution is $O(p^3)$:

- $O(np^2)$ is the complexity of computing the matrix $A = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top$.
- $O(p^3)$ is the complexity of eigendecomposition of A.

Now, consider an alternative calculation:

- Rewrite $A = X^\top X$, where X is $n \times p$ matrix with ith row $\boldsymbol{x}_i^\top$
- Consider $K = XX^\top$, $n \times n$, dot-product matrix (linear kernel), whose (i, j) element is $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$
- Suppose $\boldsymbol{u}$ is an eigenvector of K:
$$K\boldsymbol{u} = \lambda \boldsymbol{u}$$
- Multiplying the preceding equality by $X^\top$: $X^\top X X^\top \boldsymbol{u} = \lambda X^\top \boldsymbol{u}$
- Implying: $A(X^\top \boldsymbol{u}) = \lambda(X^\top \boldsymbol{u})$
- Hence, $\frac{X^\top \boldsymbol{u}}{\|X^\top \boldsymbol{u}\|}$ is the eigenvector of A
- We can thus calculate the PCA solution from eigendecomposition of K with complexity $O(n^3 + n^2 p)$

Kernel PCA

- Expand basis $\boldsymbol{x} \to \boldsymbol{\phi}(\boldsymbol{x})$, where $\phi(\boldsymbol{x})$ is in some Hilbert space
- Now, PCA is based on orthogonal projections

$$\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}) \rangle$$

- Which, based on the preceding slide, can be done by performing eigen decomposition of the kernel matrix $K = \{\langle \boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\phi}(\boldsymbol{x}_j) \rangle\}$
- Moreover, recall that our approximation functions are of the form

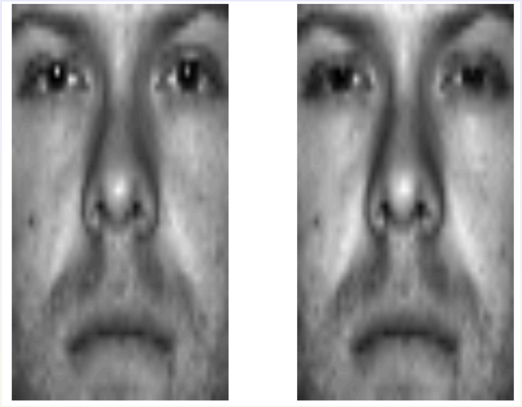
$$g_1(x) = \sum_{i=1}^{n} \phi_{i1} k(x, x_i)$$

- Hence, we can think of finding the first principle component of Kernel PCA as the variance maximization

$$\max_{g_1 \in \mathcal{H}_k} \text{Sample Var}(g_1(X)) \quad \text{subject to} \quad \|g_1\|_{\mathcal{H}_k} = 1$$

- Now the approximation "directions" are not lines anymore

Example: Dimensionality Reduction of Images



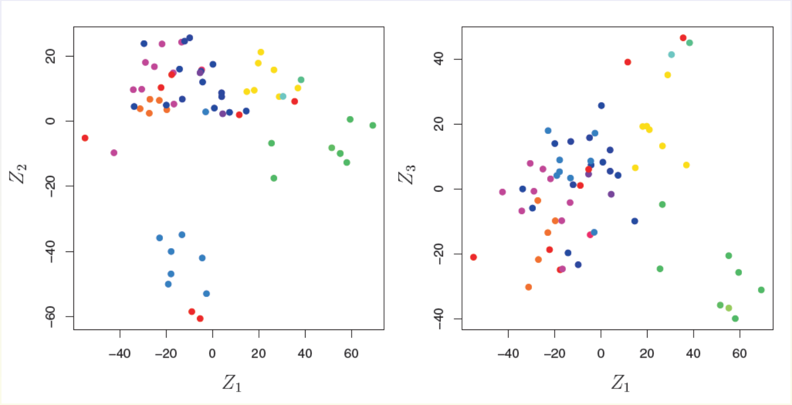
- Top left: 50×50 pixel image: $p = 2,500$ dimensions
- Top right: Reconstruction with $k = 10$ principal components: $10 \ll 2,500(!)$

PCA on the NCI60 Data

- First *scale* the data
  > pr.out=prcomp(nci.data, scale=TRUE)
- Then, **assign a color** to each of the 64 cancer cell lines
  ```
  Cols=function(vec){
  + cols=rainbow(length(unique(vec)))
  + return(cols[as.numeric(as.factor(vec))]) }
  ```
- We now can plot the principal component score vectors
  ```
  > par(mfrow=c(1,2))
  > plot(pr.out$x[,1:2], col=Cols(nci.labs), pch=19,
    xlab="Z1",ylab="Z2")
  > plot(pr.out$x[,c(1,3)], col=Cols(nci.labs), pch=19,
    xlab="Z1",ylab="Z3")
  ```

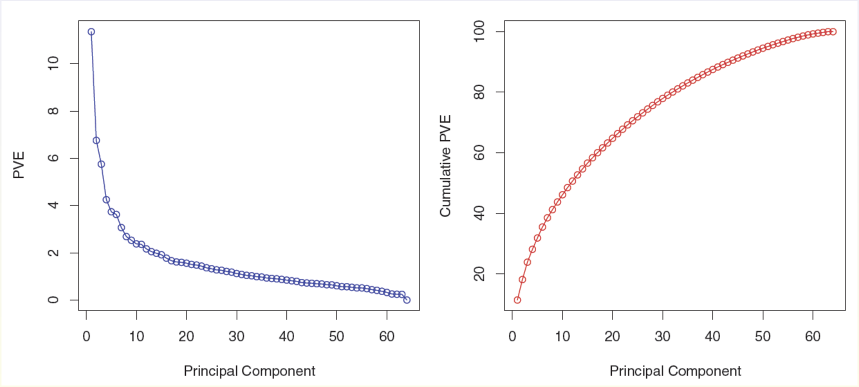
PCA on the NCI60 Data

- Observations belonging to a single cancer type tend to lie near each other in this low-dimensional space.
- Not possible to visualize data without PCA, $\binom{6832}{2}$ scatter plots (!)



Principal Value Explained (PVE) on NCI60 Data

- First 7 PC-s explain 40% of data
- 70 PC-s explain 100% of data
- Compare 70 to $p = 6832(!)$



Other Dim-Reduction Techniques: Random Projections

- Let W be a random $m \times p, m < p$ matrix, then the transformation

  $$\boldsymbol{x} \to W\boldsymbol{x}$$

  is a random projection.
- Simple compression technique that works remarkably well.
- It is based on Johnson-Lindenstrauss lemma, which shows that random projections do not distort Euclidian distance a lot, i.e.,

  $$\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \approx \|W\boldsymbol{x}_1 - W\boldsymbol{x}_2\|,$$

  which is equivalent to showing that $\|W\boldsymbol{x}\|/\|\boldsymbol{x}\| \approx 1$.

Lemma Fix $\boldsymbol{x} \in \mathbb{R}^p$. Let W be $m \times p$ random matrix with independent standard normal components. Then, for every $\epsilon \in (0, 3)$,

$$\mathbb{P}\left[\left|\frac{\|W\boldsymbol{x}/\sqrt{m}\|^2}{\|\boldsymbol{x}\|^2} - 1\right| > \epsilon\right] \leq 2e^{-\epsilon^2 m/6}.$$

Other Dim-Reduction Techniques: Random Projections

Proof: Without loss of generality assume $\|\boldsymbol{x}\| = 1$. Hence, it remains to prove

$$\mathbb{P}[(1-\epsilon)m \le \|W\boldsymbol{x}\|^2 \le (1+\epsilon)m] \ge 1 - 2e^{-\epsilon^2 m/6}$$

To this end, if $\boldsymbol{w}_i$ is the ith row of W, then $\langle \boldsymbol{w}_i, \boldsymbol{x}\rangle$ is normal with mean zero and

$$\sigma^2 = \sum_{j=1}^{p} x_j^2 = \|\boldsymbol{x}\|^2 = 1$$

Hence, $\|W\boldsymbol{x}\|^2$ has χ_m^2 distribution since

$$\|W\boldsymbol{x}\|^2 = \sum_{i=1}^{m} \langle \boldsymbol{w}_i, \boldsymbol{x}\rangle^2,$$

and $\langle \boldsymbol{w}_i, \boldsymbol{x}\rangle$ are independent standard normal variables. Thus, the claim of the lemma is equivalent to

$$\mathbb{P}[(1-\epsilon)m \le \chi_m^2 \le (1+\epsilon)m] \ge 1 - 2e^{-\epsilon^2 m/6},$$

which can be derived using Chernoff's bound, e.g., see Lemma B.12 in Shai Shalev-Shwartz and Shai Ben-David book.

Other Dim-Reduction Techniques: Compressed Sensing

Compressed Sensing:

- Also a linear technique
$$\boldsymbol{x} \to W\boldsymbol{x}$$
- Matrix W has a special property: Reconstructed Isoperimetric Property (RIP): A matrix W is (ϵ, s) RIP if for all $\|\boldsymbol{x}\| \neq 0$, such that $\|\boldsymbol{x}\|_0 \leq s$,
$$\left| \frac{\|W\boldsymbol{x}\|^2}{\|\boldsymbol{x}\|^2} - 1 \right| \leq \epsilon$$
- A random (!) $m \times p$ matrix is likely to satisfy the RIP condition provided that $m > s\log(p)$.
- Reconstruction solves a linear program, and it can be computed in polynomial time.

See Section 23.3 of Shai Shalev-Shwartz and Shai Ben-David book for more details.

Information Retrieval from Large Related Data: the Web

- Suppose we have a large amount of related data - e.g. the Web
- Relationship between data items, e.g. Web pages, can be expressed as a graph
- Information retrieval - Search: suppose we are looking for all data items (Web pages) that contain some information/keywords
- Typically the search will return a large number of results
- Hence, we need to rank them
- How?
- Example: The Google PageRank Algorithm

The Google PageRank Algorithm

- Idea: A page should be important if
  - It is pointed to by highly ranked pages
  - It is pointed/connected to by a lot of pages

Definition

- N - total number of pages
- Graph connectivity - incidence matrix
  $L_{ij} = 1$ if page j points to page i
- $c_j = \sum_{i=1}^{N} L_{ij}$ - number of pages that point to by page j - out-degree of j
- Than, PageRank assigns a rank, R_i, to page i according to

$$R_i = (1-d) + d \sum_{j=1} \frac{L_{ij}}{c_j} R_j,$$

where $0 < d < 1$ is a constant set to $d = 0.85$

PageRank Algorithm: Explanation

Let us look back at the equation

$$R_i = (1-d) + d \sum_{j=1}^{N} \frac{L_{ij}}{c_j} R_j,$$

where $0 < d < 1$ is a constant set to $d = 0.85$

- d - ensures that each page gets at least rank $1-d$
- d - also ensures that the preceding system of linear equations has a unique solution
- If some of the j-s that point to i have high rank, R_j, then R_i will be high, i.e.
  - Page i is important if some of the j-s are important
- $1/c_j$ is designed to prevent spamming, i.e., if j liberally points to a lot of pages, than its contribution to i's ranking should count less
- If i has a lot of neighbors, i.e., large $\sum_{j=1}^{N} L_{ij}$, than it's rank is large

PageRank: Solution

Let us introduce matrix notation

- $\boldsymbol{e}$ - vector of N ones
- $\boldsymbol{D}_c$ - diagonal matrix with elements c_j
- $\boldsymbol{R}$ - column vector of page ranks $(R_1, R_2, \ldots, R_N)$.
- Then, PageRank equations can be written as

$$\boldsymbol{R} = (1-d)\boldsymbol{e} + d\boldsymbol{L}\boldsymbol{D}_c^{-1}\boldsymbol{R}$$

- Introducing the normalization that the average page rank is one

$$\boldsymbol{e}^\top \boldsymbol{R} = N$$

we can write the preceding linear system as

$$\boldsymbol{R} = \left[(1-d)\frac{\boldsymbol{e}\boldsymbol{e}^\top}{N} + d\boldsymbol{L}\boldsymbol{D}_c^{-1}\right]\boldsymbol{R} = \boldsymbol{A}\boldsymbol{R}$$

where matrix $\boldsymbol{A}$ is the expression in square brackets

PageRank: Solution

Connection to Markov chains

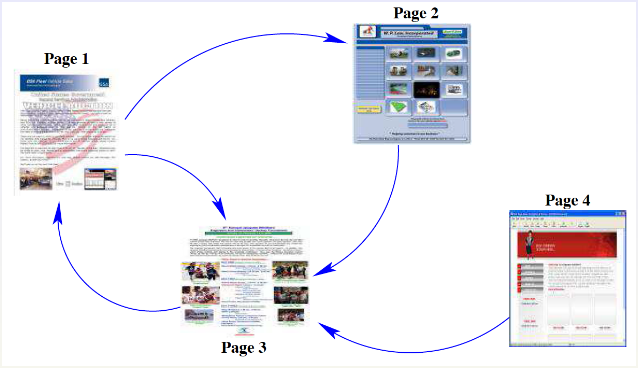
- $\boldsymbol{A}$ - is a stochastic matrix
  - Largest eigenvalue of $\boldsymbol{A}$ is 1
- $\boldsymbol{R}$ can be computed iteratively by power method, starting with $\boldsymbol{R} = \boldsymbol{R}_0$, and performing

$$\boldsymbol{R}_k \leftarrow \boldsymbol{A}\boldsymbol{R}_{k-1}; \quad \boldsymbol{R}_k \leftarrow N\frac{\boldsymbol{R}_k}{\boldsymbol{e}^\top \boldsymbol{R}_k}$$

Random surfer interpretation

- The surfer performs a random walk on the Web choosing among the outgoing links at random
- The surfer can also jump with probability $(1-d)$ to a random page on the Web
- Then, R_i represents the probability that a surfer is found at page i.

PageRank: Example



$$\boldsymbol{L} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{c} = \begin{bmatrix} 2 & 1 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} 1.49 & 0.78 & 1.58 & 0.15 \end{bmatrix}$$

PageRank: Stochastic Analysis

- Numerical solution for $\boldsymbol{R}$ doesn't reveal qualitative behavior
- Measurements showed that $\boldsymbol{R}$ follows power law behavior $\sim 1/i^{\alpha}$
- Why?

In Jelenković & Olvera-Cravioto (2010), we introduced a stochastic formulation

$$R = Q + \sum_{i \in \mathcal{N}} C_i R_i$$

where

- R is a typical, random, page on the Web
- $\mathcal{N}$ - typical, random, number of neighbors that are pointing to this page
- R_i - typical rank of these neighbors
- C_i - random variables taking into account d/c_j
- Q - random variable taking into account d

PageRank: Stochastic Solution

Main implications of Jelenković & Olvera-Cravioto (2010) and later work

- Explain why $\boldsymbol{R}$ follows power law behavior $\sim 1/i^{\alpha}$
- Known fact - Web graph is a power law graph, i.e. $|\mathcal{N}|$ - has a power law distribution
- R can either be large because
  - it is pointed to by a large number of lower-ranked pages
  - or, it is pointed to by a small number of lower-ranked pages
- Reveals a weakness: the factor $1/c_j$ doesn't prevent spamming

Improvements

- Prevent spamming
- Design other ranking schemes, e.g. personalized Web search

Association Rules: Market Basket Analysis

- Popular technique for mining commercial data bases
  - Amazon or Walmart customer data bases
- $X = (X_1, \dots, X_p)$
- p - total number of items in the store - very large
- Often $X_i \in \{0, 1\}$ - so $X = (X_1, \dots, X_p)$ represents the times that a customer bought
  - $\boldsymbol{x}_i = (x_{i1}, \dots, x_{ip})$ represents items that customer i bought "market basket"
- Those variables that frequently have joint values of one represent items that are frequently purchased together.
- General problem - find vectors v_i, such that $\mathbb{P}[X = v_i]$ is large
- This problem is impossibly difficult
- Can we compute the histogram of X?
  p large - never enough data

Association Rules: Simplification

- Instead of finding values of v for which $\mathbb{P}[X = v]$ is large
  - Find regions of the X-space with high probability relative to its size or support
  - Let s_j be a subset of values that X_j can take, then we try to find these subsets s_j for which

$$\mathbb{P}[\cap_{j=1}^{p}\{X_j \in s_j\}]$$

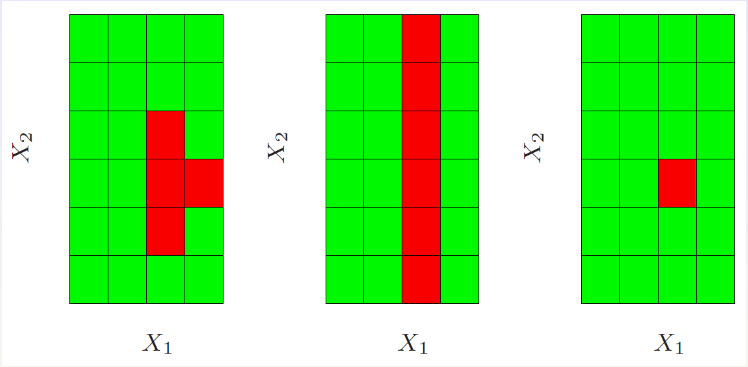
  - Even this is not feasible for $p \approx 10^4$ and $N \approx 10^8$
- Note that if $s_j =$entire set of values for X_j, then feature j is excluded
- Hence, we can only focus on a subset of items $\mathcal{J} \subset \{1, \dots, p\}$ and try to maximize

$$\mathbb{P}[\cap_{j \in \mathcal{J}}\{X_j \in s_j\}]$$

  or even simpler take each s_j to be a single item

$$\mathbb{P}[\cap_{j \in \mathcal{J}} 1_{\{X_j = v_j\}}]$$

Association Rules: Example



- The red squares indicate areas of high density.
- To simplify, we assume that the derived subset corresponds to either a single value of an input or all values.
- With this assumption we could find either the middle or right pattern, but not the left one.

Turn Everything to Binary Variables

- Introducing dummy variables can turn X into binary vectors Z
- Recall $\mathcal{S}_|$ - all values that X_j can take
- Let $K = \sum_{j=1}^{p} |\mathcal{S}_j|$
- Then $Z_k = 1$ is item $k, 1 \leq k \leq K$ is present in the basket
- $\mathcal{K} \subset \{1, \ldots, K\}$ - item set, then we maximize

$$\mathbb{P}[\prod_{k \in \mathcal{K}} Z_k = 1]$$

- The preceding probability can be estimated as

$$\mathbb{P}[\prod_{k \in \mathcal{K}} Z_k = 1] \approx \frac{1}{N} \sum_{i=1}^{N} \prod_{k \in \mathcal{K}} z_{ik}$$

  this is called the "support" or "prevalence", $T(\mathcal{K})$, of the item set $\mathcal{K}$
  z_{ik} is the value of Z_k for the i-th case/customer
- For a threshold t, find all item sets $\mathcal{K}_l$, for which $T(\mathcal{K}_l) > t$

The Apriori Algorithm

- Agrawal et al. (1995) - exploits
  - Choose a threshold t such that sets $\mathcal{K}_l$, for which $T(\mathcal{K}_l) > t$, have small number of items
  - If $\mathcal{L} \in \mathcal{K}$, then $T(\mathcal{L}) \geq T(\mathcal{K})$
- First pass: Find all single items whose support is bigger than the threshold (other items are discarded)
- Second pass: computes the support of all item sets of size two that can be formed from pairs of the single items surviving the first pass.
- Continue this procedure for sets of $|\mathcal{K}| = m, m = 3, 4, \ldots$ items, by only considering only those from the previous, $m - 1$, pass with those retained from the first pass.
- Passes over the data continue until all candidate rules from the previous pass have support less than the specified threshold.
- The Apriori algorithm requires only one pass over the data for each value of $|\mathcal{K}|$
- If the data are sufficiently sparse (or t is high enough), then the process will terminate in reasonable time even for huge data sets.

The Apriori Algorithm

- Each high support item set $\mathcal{K}$ returned by the Apriori algorithm is cast into a set of association rules.
- The items $Z_k, k \in \mathcal{K}$ are partitioned into two disjoint subsets, $\mathcal{A} \cup \mathcal{B} = \mathcal{K}$, and written
$$\mathcal{A} \Rightarrow \mathcal{B}$$
- The first item subset A is called the antecedent
- The second B the consequent
- The support of the rule $T(\mathcal{A} \Rightarrow \mathcal{B}) =$ fraction of observations in the union of the antecedent and consequent, which is just the support of the item set $\mathcal{K}$
- $T(\mathcal{A} \Rightarrow \mathcal{B}) = T(\mathcal{K})$ is an estimate of the probability of simultaneously observing both item sets $\mathbb{P}(\mathcal{A} \cap \mathcal{B})$ in a randomly selected market basket.

The Apriori Algorithm

- The confidence or predictability
$$C(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{T(\mathcal{A} \Rightarrow \mathcal{B})}{T(\mathcal{A})}$$
is an estimate of the conditional probability $\mathbb{P}(\mathcal{B}|\mathcal{A})$
- The "lift"
$$L(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{C(\mathcal{A} \Rightarrow \mathcal{B})}{T(\mathcal{B})}$$
is an estimate of
$$\frac{\mathbb{P}(\mathcal{A} \cap \mathcal{B})}{\mathbb{P}(\mathcal{A})\mathbb{P}(\mathcal{B})}$$

The Apriori Algorithm: Example

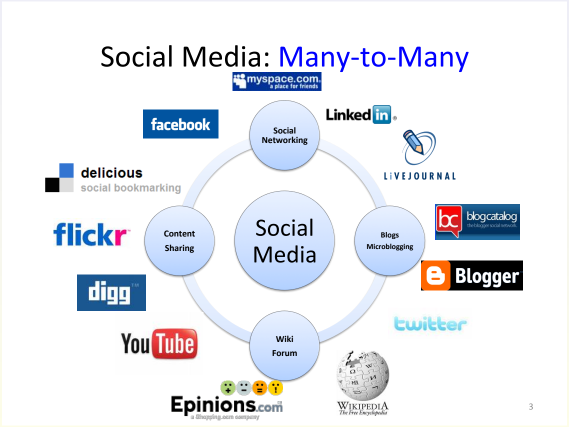
- $\mathcal{K}$ ={peanut butter, jelly, bread}
- consider the rule {peanut butter, jelly} $\Rightarrow$ {bread}.
- support value of 0.03 for this rule means that peanut butter, jelly, and bread appeared together in 3% of the market baskets.
- confidence of 0.82 for this rule implies that when peanut butter and jelly were purchased, 82% of the time bread was also purchased.
- If bread appeared in 43% of all market baskets then the rule {peanut butter, jelly} $\Rightarrow$ {bread} would have a lift of 1.95.

Goal: to find rules with high support and confidence

Social Networks

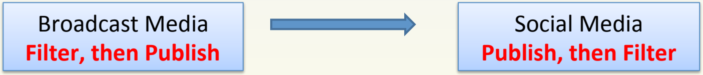
This part of the lecture is based on the book:
Community Detection and Mining in Social Media, L. Tang & H. Liu, 2010

- World Wide Web, Facebook, Twiter, Youtube, etc.



Characteristics of Social Media

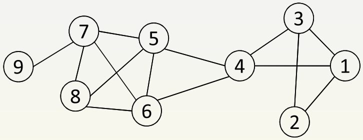
- Consumers become Producers
- Rich User Interaction
- User-Generated Contents
- Collaborative environment
- Collective Wisdom
- Long Tail



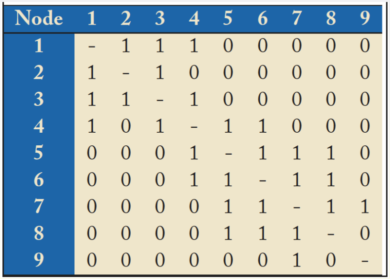
Social Media: Graph Representation

- Social Network: made of nodes (individuals or organizations) and edges that connect nodes in various relationships like friendship, kinship etc.

- Graph Representation



- Matrix Representation



Node	1	2	3	4	5	6	7	8	9
1	-	1	1	1	0	0	0	0	0
2	1	-	1	0	0	0	0	0	0
3	1	1	-	1	0	0	0	0	0
4	1	0	1	-	1	1	0	0	0
5	0	0	0	1	-	1	1	1	0
6	0	0	0	1	1	-	1	1	0
7	0	0	0	0	1	1	-	1	1
8	0	0	0	0	1	1	1	-	0
9	0	0	0	0	0	0	1	0	-

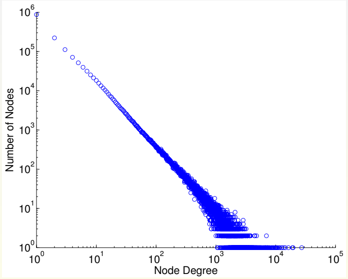
Graphs: Basic Concepts

- A: the adjacency matrix
- V: the set of nodes
- E: the set of edges
- v_i: a node v_i
- $e(v_i, v_j)$: an edge between node v_i and v_j
- N_i: the neighborhood of node v_i
- d_i: the degree of node v_i
- Geodesic distance: a shortest path between two nodes

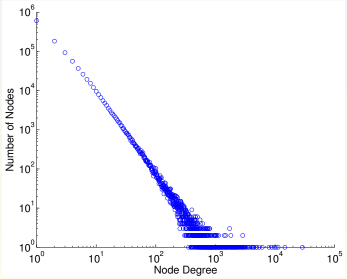
Properties of Large-Scale Social Networks

- Networks in social media are typically huge, involving millions of actors and connections.
- Large-scale networks in real world demonstrate similar patterns
  - Strong Community Structure
  - Small-world effect
  - Scale-free distributions: power laws - $1/i^{\alpha}$

Log-Log Plot of Power Law Distributions: $\log(1/i^{\alpha}) = -\alpha \log(i)$



Friendship Network in Flickr



Friendship Network in YouTube

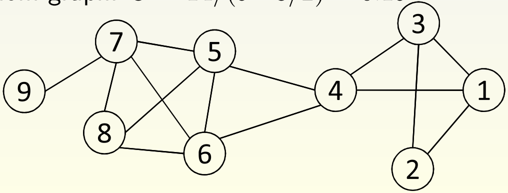
Measuring Community Structure

Clustering coefficient:

$$C_i = \frac{\text{\# of closed triplets}}{\text{\# of connected closed triplets}}$$

Example:

- $d_6 = 4, N_6 = \{4, 5, 7, 8\}$
- $k_6 = 4 : e(4,5), e(5,7), e(5,8), e(7,8)$
- $C_6 = 4/(4 * 3/2) = 2/3$
- Average clustering coefficient $C = (C_1 + C_2 + \ldots + C_n)/n$
- $C = 0.61$ for the example network
  In a random graph: $C = 14/(9 * 8/2) = 0.19$



Some Social Network Challenges

Network Modeling

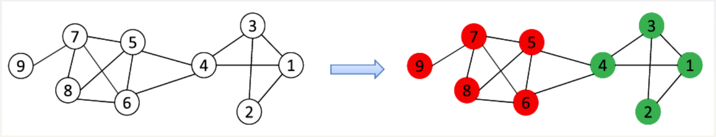
- Small-world effect (e.g., 6 degrees of separation)
- Power-law distribution (a.k.a. scale-free distribution)
- Community structure (high clustering coefficient)
- Common model: Preferential attachment
- Take a look at:
  Directed random graphs with given degree distributions, N. Chen & M. Olvera-Cravioto, Stochastic Systems, 3 (1), 147-186, 2013.

Community Detection - Graph Clustering

Community Detection - Graph Clustering

A community is a set of nodes between which the interactions are (relatively) frequent

- A.k.a., group, cluster, cohesive subgroups, modules



Applications: Recommendation based communities, Network Compression, Visualization of a huge network

Clustering Based on Vertex Similarity

This is familiar(!)

Define node similarity/distance:

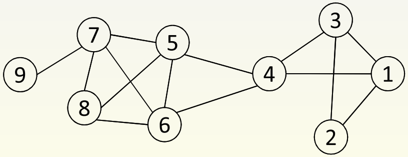
- Jaccard Similarity

$$d(v_i, v_j) = \text{Jaccard}(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

- Cosine similarity

$$d(v_i, v_j) = \text{Cosine}(v_i, v_j) = \frac{\sum_k A_{ik} A_{kj}}{\sqrt{\sum_k A_{ik}^2 \sum_k A_{ik}^2}}$$

$$\text{Jaccard}(4,6) = 1/7, \quad \text{Cosine}(4,6) = 1/4$$



Then, we can run any known clustering algorithm, e.g. K-means

Recall K-Means

What are centroids on a graph? - define properly

- Centroid of a graph: A vertex (or set vertices) u where the greatest distance $d(u,v)$ to other vertices v is minimal.

K-Means Algorithm

1. Initialization: Randomly assign a number, from 1 to K, to each of the nodes. (Or, select K nodes randomly to be centroids.)
2. Iterate until the cluster assignments stop changing:
   (a) For each of the K clusters, compute the cluster centroid.
   (b) Assign each observation to the cluster whose centroid is closest.

Divisive Hierarchical Clustering: Bottom up

Divisive clustering

- Partition nodes into several sets
- Each set is further divided into smaller ones

Example: recursively remove the weakest tie

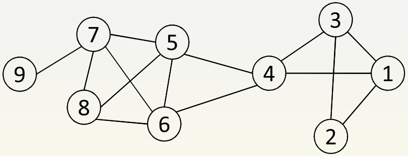
- Find the edge with the least strength
- Remove the edge and update the corresponding strength of each edge

Recursively apply the above two steps until a network is discomposed into desired number of connected components.
Each component forms a community.

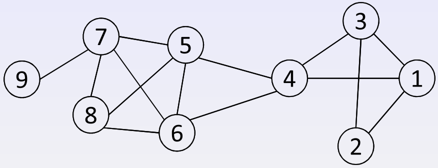
Strength of a Tie: Edge Betweenness

Edge betweenness: the # of shortest paths that pass along with the edge

The edge betweenness of $e(1,2)$ is 4, as:
all the shortest paths from 2 to {4, 5, 6, 7, 8, 9} have to either pass $e(1,2)$ or $e(2,3)$, and $e(1,2)$ is the shortest path between 1 and 2



Example



Initial betweenness value

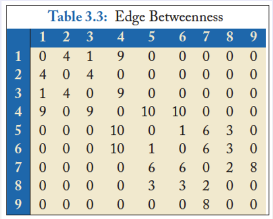
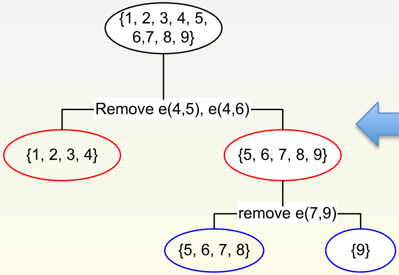


Table 3.3: Edge Betweenness

	1	2	3	4	5	6	7	8	9
1	0	4	1	9	0	0	0	0	0
2	4	0	4	0	0	0	0	0	0
3	1	4	0	9	0	0	0	0	0
4	9	0	9	0	10	10	0	0	0
5	0	0	0	10	0	1	6	3	0
6	0	0	0	10	1	0	6	3	0
7	0	0	0	0	6	6	0	2	8
8	0	0	0	0	3	3	2	0	0
9	0	0	0	0	0	0	8	0	0

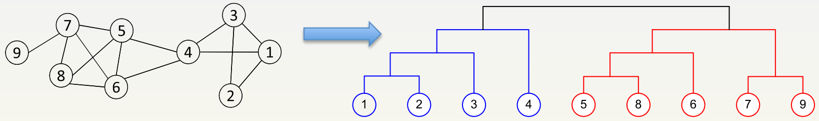


After remove e(4,5), the betweenness of e(4, 6) becomes 20, which is the highest;

After remove e(4,6), the edge e(7,9) has the highest betweenness value 4, and should be removed.

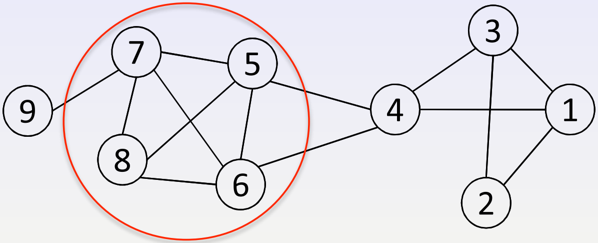
Agglomerative Hierarchical Clustering

- Initialize each node as a community
- Merge communities successively into larger communities following a certain criterion



Other Ideas: Using Cliques

Clique: a maximum complete subgraph in which all nodes are adjacent



- NP-hard to find the maximum clique in a network
- Straightforward implementation to find cliques is very expensive in time complexity
- There are heuristics: Find a clique of size k, and prune all nodes with $< k - 1$ neighbors
- Idea for clustering: Use cliques as seeds to form larger communities

Approximating Cliques: Density Based Groups

- The group-centric criterion requires the whole group to satisfy a certain condition
  - E.g., the group density $\geq$ a given threshold
- A subgraph $G_s(V_s, E_s)$ is a *γ-dense quasi-clique* if

$$\frac{|E_s|}{|V_s|(|V_s|-1)/2} \geq \gamma$$

- A similar strategy to that of cliques can be used
  - Find a maximal quasi-clique, say, of size k
  - Remove nodes with degree $< \gamma k$

Dimensionality Reduction on Graphs

- Map nodes into a low-dimensional space such that:
  - The proximity between nodes is preserved in the new space
  - Solutions usually involve eiigen-decomposition
  - Then perform clustering, e.g. k-means

Reading on Community Detection and Social Networks

- Book: Community Detection and Mining in Social Media, Lei Tang & Huan Liu, 2010
- Free download: `https://www.morganclaypool.com/doi/abs/10.2200/S00298ED1V01Y201009DMK003`
- Other book resources: `http://dmml.asu.edu/cdm/`

Comprehensive R-Package for Graph Analysis

- iGraph - `http://igraph.org/r/#docs`
- R manual: `http://igraph.org/r/doc/igraph.pdf`
- Various tutorials for iGraph in R available online

Reading on PageRank

ESL:	Section 14.10
Paper	P.R. Jelenković and M. Olvera-Cravioto, Information ranking and power laws on trees, *Advances in Applied Probability*, 42 (4), pp. 1057-1093, 2010.

Reading on Association Rules

ESL:	Section 14.2
Paper	Agrawal et al., Fast discovery of association rules, Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1995.

Dimensionality Reduction: Chapter 23 in:

- Shai Shalev-Shwartz and Shai Ben-David, Understanding Machine Learning: From Theory to Algorithms, 2014. `https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf`

Final Project

- Groups of 4 students
- **Deliverables:**
  - **Paper** - 15 - 20 pages
  - **10min Presentation** with about 10 slides
- **Due: likely Dec 20, 11:59pm**
- One slot for presentations on **Tue, Dec 13, 4:10-6:40pm**
  Other slots to be scheduled on **Friday, Dec 16**
- **Suggested Data Repositories**:
  - UC Irvine Machine Learning Repository
    `https://archive.ics.uci.edu/ml/datasets.html`
  - Datasets supported by Bioconductor:
    `http://www.bioconductor.org/packages/release/data/experiment/`
  - GEO (Gene Expression Omnibus) Data Repository
    `https://www.ncbi.nlm.nih.gov/geo/`

Final Project

Final Paper Outline:

1. Introduction: e.g., describe the application area, problems considered, etc
2. Data set(s) and paper(s): e.g., describe data in detail, what was done in the paper(s), common stat/machine learning tools, etc
3. Reproduce the results from the paper(s)
4. Try different techniques learned in class, or propose new ones
5. Discussion and conclusion: e.g., compare different techniques, pros and cons, future work, etc

General

- Document software well
- Use R, unless there is a compelling reason to use Python, e.g., you implemented a new algorithm
- Academic Honesty - do not plagiarize, as the papers will be submitted via Turnitin, which automatically checks for plagiarization.

Have fun and GOOD LUCK!