CS4220: Knowledge Discovery Methods for Bioinformatics
Unit 1: Essence of Knowledge Discovery
(Part C: Data Mining)

Wong Limsoo
Outline

• Clustering, aka unsupervised learning
• Association rule mining
• Classification, aka supervised learning
CLUSTERING
Objective of cluster analysis

- Find groups of objects s.t. objects in a group are
  - Similar (or related) to one another
  - Diff from (or unrelated to) objects in other groups

Intra-cluster distances are minimized

Inter-cluster distances are maximized

Cohesive, compact

Distinctive, apart
The notion of a “cluster” can be ambiguous

How many clusters?

Six Clusters

Two Clusters

Four Clusters
Supervised vs unsupervised learning

- **Supervised learning (aka classification)**
  - Training data (observations, measurements, etc.) are accompanied by class
  - New data is classified based on training data

- **Unsupervised learning (aka clustering)**
  - Class labels of training data are unknown
  - Given a set of measurements, observations, etc., aim to establish existence of classes in the data
Clustering techniques

• **Partitional clustering: K-means**
  – Division of data objects into non-overlapping subsets (clusters) s.t. each data object is in exactly one subset

• **Hierarchical clustering: Agglomerative approach**
  – A set of nested clusters organized as a hierarchical tree

• **Subspace clustering and bi-/co-clustering**
  – Simultaneous clustering on a subset of tuples and a subset of attributes
Partitional clustering: K-means

- Each cluster has a centroid
- Each point is assigned to a cluster based on closest centroid
- # of clusters, K, must be specified

1: Select $K$ points as the initial centroids.
2: repeat
3: Form $K$ clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: until The centroids don’t change
Details of K-means

• Initial centroids are often chosen randomly
  – Clusters produced vary from one run to another
• Centroid is the “mean” of points in the cluster
• “Closeness” is measured by Euclidean distance, cosine similarity, correlation, etc
• K-means usually converges in a few iterations
  – Often the stopping condition is changed to “until relatively few points change clusters”
• Complexity is O(n * K * i * d)
  – n = # of points, K = # of clusters, i = # of iterations, d = # of attributes
Example iterations by K-means
Two different K-means clusterings

Original Points

Optimal Clustering

Sub-optimal Clustering
Evaluating K-means clusters

• **Sum of Squared Error (SSE) is commonly used**
  – Error of a point is its distance to nearest centroid
  – Square these errors and sum them to get SSE

\[
SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)
\]

where \( C_i \) is a cluster, \( m_i \) is its centroid

• **Can reduce SSE by increasing K, the # of clusters**
• **A good clustering with smaller K can have a lower SSE than a poor clustering with higher K**
Importance of initial centroids

Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5
Solutions to initial-centroid problem

- **Multiple runs**
  - Helps, but probability is not on your side
- **Use hierarchical clustering to determine initial centroids**
- **Select >k initial centroids and then select the most widely separated among these initial centroids**
- **Use more advanced algos, like “Bisecting K-Means”, that are not as susceptible to initialization issues**
Limitations of K-means

• Has problems when clusters are of differing
  – Sizes
  – Densities
  – Non-globular shapes

• Also has problems when data contain outliers
Overcoming K-means’ limitations

- One solution is to use many clusters
  - Find parts of clusters
  - But need to put them together
Hierarchical clustering

Hierarchical clustering

- Organize similar data into groups
- Form groups into a hierarchical tree structure, termed a Dendrogram

- Offer useful visual descriptions of data

- Two approaches
  - Agglomerative
    - Build the tree by finding most related objects first
  - Divisive
    - Build the tree by finding most dissimilar objects first.
Distance matrix

- Square, symmetrical

- Element value is based on a similarity function, e.g., Euclidian distance

- Sometimes, it’s called a Similarity Matrix or a Proximity Matrix

Distance (similarity) Matrix

\[ P(i, j) = \text{dist}(p_i, p_j) \]
Agglomerative hierarchical clustering

- **Basic algo is straightforward**

  Compute proximity matrix
  Let each data point be a cluster
  Repeat
    - Merge the two closest clusters
    - Update the proximity matrix
  Until only a single cluster remains

- **Key is computing proximity of two clusters**
  - Diff approaches to defining distance betw clusters
distinguish the diff algos
Starting situation

- Start with clusters of individual points and a proximity matrix.

Proximity Matrix:

```
  p1  p2  p3  p4  p5  ...
p1
p2
p3
p4
p5
```

Starting points:

- p1
- p2
- p3
- p4
- p5
- ...

- p9
- p10
- p11
- p12
Intermediate situation

- After some merging steps, we have some clusters
Intermediate situation

- We want to merge two closest clusters (C2, C5) and update the proximity matrix.
Defining inter-cluster similarity

- **Other methods use an objective function**
  - Ward’s method uses squared error
Finally, get a resulting dendrogram
Strengths of hierarchical clustering

• No need to assume any particular # of clusters
  – Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

• They may correspond to meaningful taxonomies
  – Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)
Divisive hierarchical clustering

- Start with one, all-inclusive cluster
- At each step, split a cluster until each cluster contains a point (or there are k clusters)

**Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm**

1: Compute a minimum spanning tree for the proximity graph.
2: repeat
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: until Only singleton clusters remain
In case you don’t know what a MST is…

- To build a MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points \((p, q)\) s.t. \(p\) is in the current tree but \(q\) is not
  - Add \(q\) to the tree and put an edge between \(p\) and \(q\)
Subspace clustering

• Cluster boundaries clear only wrt the subspaces

Bi- or co-clustering

• Simultaneous clustering on a subset of attributes and a subset of tuples
High-dimensional data

- Many applications need clustering on high-dimensional data
  - Text documents
  - Microarray data

- Major challenges:
  - Many irrelevant dimensions may mask clusters
  - Distance measure becomes meaningless
    - The “equi-distance” phenomenon
  - Clusters may exist only in some subspaces
Curse of dimensionality

- Data in only one dimension is relatively packed
- Adding a dimension “stretches” the points across that dimension, making them further apart
- Adding more dimensions makes the points further apart
  - High-dimensional data is sparse
  ⇒ Distance measure becomes meaningless, as most data points become equi-distance to each other

Image credit: Parsons et al. *KDD Explorations*, 2004
Why subspace clustering?

- Clusters may exist only in some subspaces
- Subspace-clustering: find clusters in all the subspaces

Exercise: Which dimension combinations are best for identifying which clusters?

Image credit: Parsons et al. *KDD Explorations*, 2004
However, inspect your subspace clusters carefully!

Image credit: Eamonn Keogh
Time for Exercise #1

- The picture shows random sets of 3x3 cells selected from randomly generated nxn samples for various n. The darker the cell, the higher the value it contains.

- Based on this picture, discuss the effect of high dimensionality on clustering and feature selection.

Image of a slide from a talk by Koji Tsuda.
CLIQUE (Clustering In QUEst)

• Automatically identify subspaces of a high dimensional data space that allow better clustering than original space

CLIQUE: The Major Steps

• Partition the data space
• Identify subspaces that contain clusters
  – Use the “Apriori Principle”
    • Find dense units in all subspaces
    • Form connected dense units in all subspaces

• Generate minimal description for the clusters
  – Determine maximal regions that cover a cluster of connected dense units
  – Determination of minimal cover for each cluster
Biclustering

- Please read these two papers yourself 😊
  - Cheng & Church. “Biclustering of expression data”. *ISMB 2000*

Biclusters = small boxes of homogeneity

A small box =
A subset of attributes X
A subset of tuples
A special case of biclustering: Biclique detection

- When the table is a binary matrix of 0s and 1s
- Convert the table into a bipartite graph

Then, a max biclique corresponds to a bicluster

A good algo for max biclique can be found at
What have we learned?

- **Partitional clustering**
  - K-means

- **Hierarchical clustering**
  - Agglomerative approach
  - Divisive approach

- **Subspace clustering and bi-/co-clustering**, albeit rather briefly!

- **How to evaluate quality of clusters**
  - SSE

- **A general strategy for some difficult-to-cluster situations**
  - Differing sizes
  - Differing densities
  - Non-globular
References

• **Must read**

• **Good to read**
  – Agrawal et al. “Automatic Subspace Clustering of High Dimensional Data”. *Data Mining & Knowledge Discovery*, 11(1):5-33, 2005
For those who want to go further...

- Much progress has been made in scalable clustering methods
  - Partitioning: k-means, k-medoids, CLARANS
  - Hierarchical: BIRCH, ROCK, CHAMELEON
  - Density-based: DBSCAN, OPTICS, DenClue
  - Grid-based: STING, WaveCluster, CLIQUE
  - Model-based: EM, Cobweb, SOM
  - Frequent pattern-based: pCluster
  - Constraint-based: COD, constrained-clustering
ASSOCIATION RULE MINING
Market-basket analysis

• What do my customers buy?
• Which products are bought together?

• Find associations and correlations between the different items that customers buy

Source: A. Puig
**Association rule mining**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>bread, jelly, peanut-butter</td>
</tr>
<tr>
<td>T2</td>
<td>bread, peanut-butter</td>
</tr>
<tr>
<td>T3</td>
<td>bread, milk, peanut-butter</td>
</tr>
<tr>
<td>T4</td>
<td>beer, bread</td>
</tr>
<tr>
<td>T5</td>
<td>beer, milk</td>
</tr>
</tbody>
</table>

- **Frequent itemsets**
  - Items that often appear together
    - \{bread, peanut-butter\}
- **Association rules**
  - bread \(\Rightarrow\) peanut-butter

- Transaction db \(T = \{t_1, ..., t_n\}\) is a set of trans
- Each trans \(t_k\) is an itemset \(I = \{i_1, ..., i_m\}\)
- Findfreq patterns, associations, ... among sets of items in \(T\)
- Represent these relationships as association rules \(X \Rightarrow Y\)
What is an interesting rule?

• **Support count, \( \sigma \)**
  – # of occurrence of an itemset
  – \( \sigma(\{\text{bread, peanut-butter}\}) = 3 \)

• **Support, \( s \)**
  – Fraction of transactions containing that itemset
  – \( s(\{\text{bread, peanut-butter}\}) = 3/5 \)

• **Frequent itemset**
  – An itemset whose support \( \geq \) a threshold \( \minsup \)
What is an interesting rule?

- **Association rule**
  - $X \Rightarrow Y$

- **Support, $s$**
  - # of trans containing $X, Y$

- **Confidence, $c$**
  - How often $Y$ occurs in trans containing $X$

$s = \frac{\sigma(X \cup Y)}{\# \text{ of trans.}}$, $c = \frac{\sigma(X \cup Y)}{\sigma(X)}$

Source: A. Puig
Apriori

• Apriori is the classic assoc rule mining algo

• Mines assoc rules in two steps
  1. Generate all freq itemsets with support $\geq \text{minsup}$
  2. Generate assoc rules using these freq itemsets

Let’s work on Step 1 first…
Step 1 of Apriori:
Generate freq itemsets with support $\geq$ minsup

- Given $d$ items. There are $2^d$ possible itemsets
- Do we need to generate them all?

Source: A. Puig
Anti-monotonicity

- Downward Closure Property:

  Any subset of a frequent itemset is frequent

⇒ If an itemset is not frequent, none of its supersets can be frequent

⇒ If an itemset is not frequent, there is no need to explore its supersets
Step 1 of Apriori:
Generate freq itemsets with support ≥ minsup

- By anti-monotonicity, if B’s support < minsup, we can prune all its supersets. I.e., no need to generate these itemsets

Source: A. Puig
Apriori’s Step 1 in Pseudo Codes

- \( k = 1 \)
- Generate frequent itemsets of length 1
- Repeat until no frequent itemsets are found
  - \( k := k + 1 \)
  - Generate itemsets of size \( k \) from the \( k-1 \) frequent itemsets
  - Compute the support of each candidate by scanning DB

---

**Algorithm Apriori(\( T \))**

\[
\begin{align*}
C_1 & \leftarrow \text{init-pass}(T); \\
F_1 & \leftarrow \{ f \mid f \in C_1, \ f.\text{count}/n \geq \text{mins}\}\sup \}; \\
\text{for} \ (k = 2; F_{k-1} \neq \emptyset; k++) \ \text{do} & \\
\quad C_k & \leftarrow \text{candidate-gen}(F_{k-1}); \\
\quad \text{for each transaction } t \in T \ \text{do} & \\
\quad \quad \text{for each candidate } c \in C_k \ \text{do} & \\
\quad \quad \quad \text{if } c \ \text{is contained in } t \ \text{then} & \\
\quad \quad \quad \quad c.\text{count}++; & \\
\quad \quad \end{align*}
\]

\[
\begin{align*}
F_k & \leftarrow \{ c \in C_k \mid c.\text{count}/n \geq \text{mins}\}\sup \}; \\
\text{return } F & \leftarrow \bigcup_{k} F_{k};
\end{align*}
\]

---

**Function candidate-gen(F_{k-1})**

\[
\begin{align*}
C_k & \leftarrow \emptyset; \\
\text{forall } f_1, f_2 & \in F_{k-1} \\
\quad \text{with } f_1 = \{i_1, \ldots, i_{k-2}, i_{k-1}\} & \\
\quad \text{and } f_2 = \{i_1, \ldots, i_{k-2}, i_{k-1}'\} & \\
\quad \text{and } i_{k-1} < i_{k-1}' \ \text{do} & \\
\quad c & \leftarrow \{i_1, \ldots, i_{k-1}, i_{k-1}';\}; \\
\quad C_k & \leftarrow C_k \cup \{c\}; \\
\quad \text{for each } (k-1)\text{-subset } s \ of \ c \ \text{do} & \\
\quad \quad \text{if } s \not\in F_{k-1} \ \text{then} & \\
\quad \quad \quad \text{delete } c \ \text{from } C_k; & \\
\end{align*}
\]

\[
\begin{align*}
\text{return } C_k; & \\
\end{align*}
\]

---

anti-monotonicity is used here
Example run of Apriori’s step 1

Source: A. Puig
Apriori

• **Apriori is the classic assoc rule mining algo**

• **Mines assoc rules in two steps**
  1. Generate all freq itemsets with support ≥ minsup
  2. Generate assoc rules using these freq itemsets

Now that we have settled Step 1, Let’s work on Step 2 next…
Step 2 of Apriori:
Generate association rules using freq itemsets

- **Given a frequent itemset L**
  - Find all non-empty subsets \( F \) of \( L \)
  - Output each rule \( F \Rightarrow \{L-F\} \) that satisfies the threshold on confidence

- **Example: \( L = \{A, B, C\} \)**
  - The candidate itemsets are: \( AB \Rightarrow C \), \( AC \Rightarrow B \), \( BC \Rightarrow A \), \( A \Rightarrow BC \), \( B \Rightarrow AC \), \( C \Rightarrow AB \)
  - In general, there are \( 2^{|L|} \) – 2 candidates!
Can we be more efficient?

- Confidence of rules generated from the same itemset does have the anti-monotone property
  - \( c(ABC \Rightarrow D) \geq c(AB \Rightarrow CD) \geq c(A \Rightarrow BCD) \)

- We can apply this property to prune rule generation

Exercise: Prove this.

Source: A. Puig
Shortcomings of Apriori

• Apriori scans the db multiple times
• There is often a high # of candidates
• Support counting for candidates takes a lot of time

• Newer methods try to improve on these points
  – Reduce the # of scans of the db
  – Reduce the # of candidates
  – Count the support of candidates more efficiently
FP-Growth

- **Build in one scan a data structure, FP-Tree**

- **Use it for fast support counting**
  - To count the support of an itemset \{FCM\}, follow the “dotted” links on M. At each node M:n, note its support n & visit its prefix chain; if FCM is found in the prefix, add n to the support

Source: A. Puig

Han et al. “Mining frequent patterns without candidate generation”. SIGMOD 2000, pp.1–12
Gr-Growth

- Build FP-Tree on the db
- Visit itemsets non-redundantly by following the right-to-left top-to-bottom SE-Tree order
- When visiting an itemset
  - Use the FP-tree to count its support efficiently
  - If it is frequent, output it, & visit its supersets
  - Otherwise skip visiting its supersets
How do you mine association rules across multiple tables?

Multidimensional association rules can be mined using the same method by transforming the problem. The items and the corresponding item values are encoded into a tuple. This results again in a finite number of possible (modified) item values, and therefore the same techniques as for single-dimensional rules apply.

Source: Karl Aberer
What have we learned?

- Frequent itemsets & association rules
- Support & confidence
- Apriori, a classic association rule mining algo
  - Anti-monotonicity
  - Search space pruning
- Advanced methods, albeit rather briefly
  - FP-Growth
  - Gr-Growth
  - Multidimensional association rule mining
References

• **Must read**

• **Good to read**
  – Agrawal et al. “Mining association rules between sets of items in large databases”. *SIGMOD 1993*, 207-216
  – Han et al. “Mining frequent patterns without candidate generation”. *SIGMOD 2000*, pp.1–12
  – Li et al. “Mining Statistically Important Equivalence Classes and Delta-Discriminative Emerging Patterns”. *KDD 2007*, pp. 430-439
For those who want to go further …

• Association rule mining has been extended in many interesting directions
  – Mining multilevel association
  – Mining multidimensional association
  – Mining quantitative association
    • R. Srikant and R. Agrawal. “Mining quantitative association rules in large relational tables”. SIGMOD 1996
  – Hypothesis exploration, testing, and analysis
CLASSIFICATION
Classification, aka supervised learning

**Model construction**
- For describing a set of predetermined classes
  - The model is represented as classification rules, decision trees, or mathematical formulae

**Model usage**
- For classifying future or unknown objects
- Estimate accuracy of the model
  - The known label of test sample is compared with the classification result from the model
- If accuracy is acceptable, use the model to classify data tuples whose class labels are unknown
Model construction

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

Classification Algorithms

IF rank = ‘professor’ OR years > 6
THEN tenured = ‘yes’
Use the model for prediction

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Merlisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

Unseen Data
(Jeff, Professor, 4)

Tenured?
Yes
Steps of model construction

• Training data gathering
• Feature generation
  – k-grams, colour, texture, domain know-how, ...
• Feature selection
  – Entropy, $\chi^2$, t-test, domain know-how...
• Feature integration
  – SVM, ANN, PCL, CART, C4.5, kNN, ...

You should have already learned this stuff from CS2220. Here is just a quick revision…
Feature selection

• **Purpose**
  – Measure the diff betw two classes, and rank the features according to the degree of the difference
  – Get rid of noisy & irrelevant features

• **Approaches**
  – Statistical tests
    • E.g., t-test, $\chi^2$-test
  – Information theory
    • E.g., Gini index, entropy, info gain
Feature integration

• I hope you still remember the various classifiers you came across in CS2220
  – Decision Trees
  – Decision Trees Ensembles
    • E.g., Bagging
  – K-Nearest Neighbour
  – Support Vector Machines
  – Bayesian Approach
  – Hidden Markov Models

If not, it is time to dig out your old CS2220 lecture slides & notes 😊
Measures of classifier performance

<table>
<thead>
<tr>
<th></th>
<th>predicted as positive</th>
<th>predicted as negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>negative</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

Accuracy = \[
\frac{\text{# correct predictions}}{\text{# predictions}} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Sensitivity = \[
= \frac{\text{# correct +ve predictions}}{\text{# +ve}} = \frac{TP}{TP + FN}
\]

Specificity = \[
= \frac{\text{# correct -ve predictions}}{\text{# -ve}} = \frac{TN}{TN + FP}
\]

Precision = \[
= \frac{\text{# correct +ve predictions}}{\text{# +ve predictions}} = \frac{TP}{TP + FP}
\]
Time for Exercise #2

- Accuracy is not a good measure if the (class) distribution of test data has bias

- Sensitivity (SE), specificity (SP), & precision (PPV) are better; but they must be used together

- How to combine SE, SP, and/or PPV?
Evaluation

• Accuracy, sensitivity, precision, etc of a classifier are generally evaluated based on blind test sets

• If adequate blind test set is unavailable, evaluate the expected performance of the learning algorithm instead
  – Sampling and apply Central Limit Theorem (CLT)
  – Cross validation
  – P-value
Evaluation by sampling & CLT

By CLT, ave accuracy of $h_1, h_2, \ldots, h_k$ is the expected accuracy of the classifier produced by the based inducer on the original samples.
Evaluation by cross validation

- Divide samples into $k$ roughly equal parts
- Each part has similar proportion of samples from different classes
- Use each part to test other parts
- Total up the accuracy
### Time for Exercise #3

#### Table: Distribution, Mean, Standard Deviation, and Sample Size

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Distribution A</th>
<th>Distribution B</th>
<th>Mean A</th>
<th>Mean B</th>
<th>Standard Deviation A</th>
<th>Standard Deviation B</th>
<th>Sample Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Normal</td>
<td>Normal</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(2)</td>
<td>Normal</td>
<td>Normal</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>(3)</td>
<td>Normal</td>
<td>Normal</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

#### Diagram: Sample Size, Effect Size, and log10(p)

- **Sample size**: 10, 30, 100
- **Effect size**: -3 to 0
- **log10(p)**: -50 to 0
Time for Exercise #3

- You are analyzing a gene expression dataset with two phenotypes, say cancer vs normal, to identify genes that behave differently between the two phenotypes.

- To help you make various decisions, you decide to do some simulations under three scenarios. In scenario (1), a gene is simulated to have the same expression distribution---specifically, $N(0,1)$, the normal distribution with mean 0 and standard deviation 1---in classes A and B. In scenario (2), the gene has the distribution $N(0,1)$ in class A and $N(0.5,1)$ in B. In scenario (3), the gene has the distribution $N(0,1)$ in class A and $N(2,1)$ in B. The scenarios are simulated 10,000 times and the (observed) effect size---defined as the (observed) mean of class A minus the (observed) mean of class B---and (observed) t-statistic p-value are computed for each simulation and plotted in slide #74.

- Discuss the threshold to use on the observed effect size when sample size is 10.
- Discuss the threshold to use on the observed effect size when sample size is 100.
- Discuss the threshold to use on the observed p-value when sample size is 10.
- Discuss the threshold to use on the observed p-value when sample size is 100.
References

• **Must read**

• **Good to read**
Acknowledgements

• The set of slides on clustering was mostly adapted from Jinyan Li

• The set of slides on association rule mining was mostly adapted from Albert Puig