CS490D:
Introduction to Data Mining
Prof. Chris Clifton

March 8, 2004
Midterm Review

Midterm Wednesday, March 10, in
class. Open book/notes.

Seminar Thursday:
Support Vector Machines

• Massive Data Mining via Support Vector Machines

• Hwanjo Yu, University of Illinois
  – Thursday, March 11, 2004
  – 10:30-11:30
  – CS 111

• Support Vector Machines for:
  – classifying from large datasets
  – single-class classification
  – discriminant feature combination discovery
Course Outline

www.cs.purdue.edu/~clifton/cs490d

1. Introduction: What is data mining?
   - What makes it a new and unique discipline?
   - Relationship between Data Warehousing, On-line Analytical Processing, and Data Mining
2. Data mining tasks - Clustering, Classification, Rule learning, etc.
3. Data mining process: Data preparation/cleansing, task identification
   - Introduction to WEKA
4. Association Rule mining
5. Association rules - different algorithm types
6. Classification/Prediction
7. Classification - tree-based approaches
8. Classification - Neural Networks Midterm
9. Clustering basics
10. Clustering - statistical approaches
11. Clustering - Neural-net and other approaches
12. More on process - CRISP-DM
    - Preparation for final project
13. Text Mining
14. Multi-Relational Data Mining
15. Future trends Final

Text: Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, August 2000.

Data Mining: Classification Schemes

- General functionality
  - Descriptive data mining
  - Predictive data mining
- Different views, different classifications
  - Kinds of data to be mined
  - Kinds of knowledge to be discovered
  - Kinds of techniques utilized
  - Kinds of applications adapted
What Can Data Mining Do?

- Cluster
- Classify
  - Categorical, Regression
- Summarize
  - Summary statistics, Summary rules
- Link Analysis / Model Dependencies
  - Association rules
- Sequence analysis
  - Time-series analysis, Sequential associations
- Detect Deviations
What is Data Warehouse?

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained separately from the organization’s operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis.
- “A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management’s decision-making process.”—W. H. Inmon
- Data warehousing:
  - The process of constructing and using data warehouses

Example of Star Schema

- time
  - time_key
day
day_of_the_week
month
quarter
year

- branch
  - branch_key
branch_name
branch_type

- Sales Fact Table
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
  - dollars_sold
  - avg_sales

- item
  - item_key
item_name
brand
type
supplier_type

- location
  - location_key
city
state_or_province
country

CS490D Midterm Review
From Tables and Spreadsheets to Data Cubes

- A data warehouse is based on a multidimensional data model which views data in the form of a data cube.
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions.
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year).
  - Fact table contains measures (such as dollars_sold) and keys to each of the related dimension tables.
- In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

Cube: A Lattice of Cuboids

- 0-D (apex) cuboid
- 1-D cuboids
- 2-D cuboids
- 3-D cuboids
- 4-D (base) cuboid
A Sample Data Cube

Warehouse Summary

- **Data warehouse**
- A multi-dimensional model of a data warehouse
  - Star schema, snowflake schema, fact constellations
  - A data cube consists of dimensions & measures
- **OLAP operations:** drilling, rolling, slicing, dicing and pivoting
- **OLAP servers:** ROLAP, MOLAP, HOLAP
- Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Multiway array aggregation
  - Bitmap index and join index implementations
- Further development of data cube technology
  - Discovery-drive and multi-feature cubes
  - From OLAP to OLAM (on-line analytical mining)
Data Preprocessing

• Data in the real world is dirty
  – incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    • e.g., occupation=""
  – noisy: containing errors or outliers
    • e.g., Salary="-10"
  – inconsistent: containing discrepancies in codes or names
    • e.g., Age="42" Birthday="03/07/1997"
    • e.g., Was rating “1,2,3”, now rating “A, B, C”
    • e.g., discrepancy between duplicate records

Multi-Dimensional Measure of Data Quality

• A well-accepted multidimensional view:
  – Accuracy
  – Completeness
  – Consistency
  – Timeliness
  – Believability
  – Value added
  – Interpretability
  – Accessibility

• Broad categories:
  – intrinsic, contextual, representational, and accessibility.
Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data transformation**
  - Normalization and aggregation
- **Data reduction**
  - Obtains reduced representation in volume but produces the same or similar analytical results
- **Data discretization**
  - Part of data reduction but with particular importance, especially for numerical data

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree
How to Handle Noisy Data?

- Binning method:
  - first sort data and partition into (equi-depth) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)
- Regression
  - smooth by fitting the data into regression functions

Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
- Attribute/feature construction
  - New attributes constructed from the given ones
Data Reduction Strategies

- A data warehouse may store terabytes of data
  - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
  - Data cube aggregation
  - Dimensionality reduction — remove unimportant attributes
  - Data Compression
  - Numerosity reduction — fit data into models
  - Discretization and concept hierarchy generation

Principal Component Analysis

- Given N data vectors from k-dimensions, find $c \leq k$ orthogonal vectors that can be best used to represent data
  - The original data set is reduced to one consisting of N data vectors on c principal components (reduced dimensions)
- Each data vector is a linear combination of the c principal component vectors
- Works for numeric data only
- Used when the number of dimensions is large
Discretization

- Three types of attributes:
  - Nominal — values from an unordered set
  - Ordinal — values from an ordered set
  - Continuous — real numbers
- Discretization:
  - divide the range of a continuous attribute into intervals
  - Some classification algorithms only accept categorical attributes.
  - Reduce data size by discretization
  - Prepare for further analysis

Data Preparation Summary

- Data preparation is a big issue for both warehousing and mining
- Data preparation includes
  - Data cleaning and data integration
  - Data reduction and feature selection
  - Discretization
- A lot of methods have been developed but still an active area of research
Association Rule Mining

• Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
  – Frequent pattern: pattern (set of items, sequence, etc.) that occurs frequently in a database [AIS93]

• Motivation: finding regularities in data
  – What products were often purchased together? — Beer and diapers?!
  – What are the subsequent purchases after buying a PC?
  – What kinds of DNA are sensitive to this new drug?
  – Can we automatically classify web documents?

Basic Concepts:
Association Rules

<table>
<thead>
<tr>
<th>Transaction-id</th>
<th>Items bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, B, C</td>
</tr>
<tr>
<td>20</td>
<td>A, C</td>
</tr>
<tr>
<td>30</td>
<td>A, D</td>
</tr>
<tr>
<td>40</td>
<td>B, E, F</td>
</tr>
</tbody>
</table>

• Itemset $X=\{x_1, \ldots, x_n\}$
• Find all the rules $X \rightarrow Y$ with min confidence and support
  – support, $s$, probability that a transaction contains $X \cup Y$
  – confidence, $c$, conditional probability that a transaction having $X$ also contains $Y$.

Let $min\_support = 50\%$, $min\_conf = 50\%$:

$A \rightarrow C$ (50\%, 66.7\%)
$C \rightarrow A$ (50\%, 100\%)
### The Apriori Algorithm—An Example

#### Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, E</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

#### 1st scan

- **C₁**
  - Itemset: (A), (B), (C), (D), (E)
  - Support: 2, 3, 3, 1, 3

#### 2nd scan

- **C₂**
  - Itemset: (A, B), (A, C), (A, E), (B, C), (B, E), (C, E)
  - Support: 1, 2, 1, 2, 3, 2

#### 3rd scan

- **C₃**
  - Itemset: (B, C, E)

#### Itemset and Support

- **L₁**
  - Itemset: (A), (B), (C), (E)
  - Support: 2, 3, 3, 3

- **L₂**
  - Itemset: (A, B), (A, C), (A, E), (B, C), (B, E), (C, E)
  - Support: 1, 2, 1, 2, 3, 2

- **L₃**
  - Itemset: (B, C, E)
  - Support: 2

### FP-Tree Algorithm

#### TID Items bought (ordered) frequent items

<table>
<thead>
<tr>
<th>TID</th>
<th>Items bought</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f, a, c, d, g, i, m, p}</td>
<td>{f, c, a, m, p}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c, f, l, m, o}</td>
<td>{f, c, a, b, m}</td>
</tr>
<tr>
<td>300</td>
<td>{b, f, h, j, o, w}</td>
<td>{f, b}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, k, s, p}</td>
<td>{c, b, p}</td>
</tr>
<tr>
<td>500</td>
<td>{a, f, c, e, i, l, p, m, n}</td>
<td>{f, c, a, m, p}</td>
</tr>
</tbody>
</table>

**min_support = 3**

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

#### Header Table

- Item frequency head
  - f: 4
  - c: 4
  - a: 3
  - b: 3
  - m: 3
  - p: 3

#### F-list

- f-c-a-b-m-p
Constrained Frequent Pattern Mining: A Mining Query Optimization Problem

- Given a frequent pattern mining query with a set of constraints $C$, the algorithm should be
  - sound: it only finds frequent sets that satisfy the given constraints $C$
  - complete: all frequent sets satisfying the given constraints $C$ are found
- A naïve solution
  - First find all frequent sets, and then test them for constraint satisfaction
- More efficient approaches:
  - Analyze the properties of constraints comprehensively
  - Push them as deeply as possible inside the frequent pattern computation.

Classification: 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>

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’
Classification: Use the Model in 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>

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent:
  \[ P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i) \]
- The product of occurrence of say 2 elements \( x_1 \) and \( x_2 \), given the current class is \( C \), is the product of the probabilities of each element taken separately, given the same class \( P([y_1,y_2],C) = P(y_1,C) * P(y_2,C) \)
- No dependence relation between attributes
- Greatly reduces the computation cost, only count the class distribution.
- Once the probability \( P(X|C_i) \) is known, assign \( X \) to the class with maximum \( P(X|C_i)*P(C_i) \)
Bayesian Belief Network

The conditional probability table for the variable LungCancer:
Shows the conditional probability for each possible combination of its parents

\[
P(z_1, \ldots, z_n) = \prod_{i=1}^{n} P(z_i | \text{Parents}(Z_i))
\]

Decision Tree

age?

<=30

30..40

>40

student?

yes

credit rating?

excellent

fair
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning — majority voting is employed for classifying the leaf
  - There are no samples left

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- \( S \) contains \( s_i \) tuples of class \( C_i \) for \( i = \{1, \ldots, m\} \)
- Information measures info required to classify any arbitrary tuple
  \[
  I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log \frac{s_i}{s}
  \]
- Entropy of attribute \( A \) with values \( \{a_1, a_2, \ldots, a_v\} \)
  \[
  E(A) = \sum_{j=1}^{v} \frac{s_j}{s} I(s_1, s_2, \ldots, s_m)
  \]
- Information gained by branching on attribute \( A \)
  \[
  Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)
  \]
Definition of Entropy

- Entropy \( H(X) = \sum_{x \in A_X} -P(x) \log_2 P(x) \)

- Example: Coin Flip
  - \( A_X = \{ \text{heads, tails} \} \)
  - \( P(\text{heads}) = P(\text{tails}) = \frac{1}{2} \)
  - \( \frac{1}{2} \log_2 \frac{1}{2} = 1 - \frac{1}{2} \)
  - \( H(X) = 1 \)

- What about a two-headed coin?
- Conditional Entropy: \( H(X | Y) = \sum_{y \in A_Y} P(y) H(X | y) \)

---

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- \( I(p, n) = I(9, 5) = 0.940 \)
- Compute the entropy for age:

<table>
<thead>
<tr>
<th>age</th>
<th>( p_i )</th>
<th>( n_i )</th>
<th>( I(p_i, n_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>2</td>
<td>3</td>
<td>0.971</td>
</tr>
<tr>
<td>30...40</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
<td>2</td>
<td>0.971</td>
</tr>
</tbody>
</table>

\( E(\text{age}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) \)
\( + \frac{5}{14} I(3,2) = 0.694 \)

\( \frac{5}{14} I(2,3) \) means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

\( Gain(\text{age}) = I(p, n) - E(\text{age}) = 0.246 \)

Similarly,

\( Gain(\text{income}) = 0.029 \)
\( Gain(\text{student}) = 0.151 \)
\( Gain(\text{credit rating}) = 0.048 \)
Overfitting in Decision Trees

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Artificial Neural Networks: A Neuron

- The \( n \)-dimensional input vector \( x \) is mapped into variable \( y \) by means of the scalar product and a nonlinear function mapping

\[
\sum_{i=1}^{n} x_i w_i + \mu_k 
\]

Output \( y \)
Artificial Neural Networks: Training

- The ultimate objective of training
  - obtain a set of weights that makes almost all the tuples in the training data classified correctly

- Steps
  - Initialize weights with random values
  - Feed the input tuples into the network one by one
  - For each unit
    - Compute the net input to the unit as a linear combination of all the inputs to the unit
    - Compute the output value using the activation function
    - Compute the error
    - Update the weights and the bias

SVM – Support Vector Machines

Small Margin  Large Margin

Support Vectors
Non-separable case

When the data set is non-separable as shown in the right figure, we will assign weight to each support vector which will be shown in the constraint.

\[ x^T \beta + \beta_0 = 0 \]

Non-separable Cont.

1. Constraint changes to the following:
   \[ y_i (x_i^T \beta + \beta_0) > C (1 - \xi_i) \]
   Where
   \[ \forall i, \xi_i > 0, \sum_{i=1}^{N} \xi_i < \text{const.} \]

2. Thus the optimization problem changes to:

   \[
   \text{Min} \| \beta \| \text{subject to } \begin{cases} 
   y_i (x_i^T \beta + \beta_0) > 1 - \xi_i, & i = 1, \ldots, N. \\
   \forall i, \xi_i > 0, \sum_{i=1}^{N} \xi_i < \text{const.} 
   \end{cases}
   \]
This classification problem clearly do not have a good optimal linear classifier.

Can we do better?
A non-linear boundary as shown will do fine.

• The idea is to map the feature space into a much bigger space so that the boundary is linear in the new space.
• Generally linear boundaries in the enlarged space achieve better training-class separation, and it translates to non-linear boundaries in the original space.
Mapping

- Mapping $\Phi : \mathbb{R}^d \mapsto H$
  - Need distances in $H$: $\Phi(x_i) \cdot \Phi(x_j)$
- Kernel Function: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$
  - Example: $K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2}$
- In this example, $H$ is infinite-dimensional

The $k$-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real-valued.
- For discrete-valued, the $k$-NN returns the most common value among the $k$ training examples nearest to $x_q$.
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples.
Case-Based Reasoning

- **Also uses**: lazy evaluation + analyze similar instances
- **Difference**: Instances are not “points in a Euclidean space”
- **Example**: Water faucet problem in CADET (Sycara et al'92)
- **Methodology**
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- **Research issues**
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

Regress Analysis and Log-Linear Models in Prediction

- **Linear regression**: \( Y = \alpha + \beta X \)
  - Two parameters, \( \alpha \) and \( \beta \) specify the line and are to be estimated by using the data at hand.
  - using the least squares criterion to the known values of \( Y_1, Y_2, \ldots, X_1, X_2, \ldots \)
- **Multiple regression**: \( Y = b_0 + b_1 X_1 + b_2 X_2 \).
  - Many nonlinear functions can be transformed into the above.
- **Log-linear models**:
  - The multi-way table of joint probabilities is approximated by a product of lower-order tables.
  - Probability: \( p(a, b, c, d) = \alpha_{ab} \beta_{ac} \gamma_{ad} \delta_{bcd} \)
Bagging and Boosting

- General idea
  - Training data
    - Classification method (CM)
      - Classifier C
  - Altered Training data
    - Classifier C1
  - Altered Training data
    - Classifier C2
  - Altered Training data
  - Aggregation ....
    - Classifier C*

Test Taking Hints

- Open book/notes
  - Pretty much any non-electronic aid allowed
- See old copies of my exams (and solutions) at my web site
  - CS 526
  - CS 541
  - CS 603
- Time will be tight
  - Suggested “time on question” provided
Seminar Thursday: Support Vector Machines

- Massive Data Mining via Support Vector Machines
- Hwanjo Yu, University of Illinois
  - Thursday, March 11, 2004
  - 10:30-11:30
  - CS 111
- Support Vector Machines for:
  - classifying from large datasets
  - single-class classification
  - discriminant feature combination discovery