Data Mining – An Introduction

Johannes Gehrke
Department of Computer Science
Cornell University
johannes@cs.cornell.edu
http://www.cs.cornell.edu/johannes

Classification
Goal: Learn a function that assigns a record to one of several predefined classes.

Definition
Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.

Example pattern (Census Bureau Data):
If (relationship = husband) and (gender = male), 99.6%

Classification Example
- Example training database
  - Two predictor attributes:
    - Age and Car-type (Sport, Minivan and Truck)
    - Age is ordered, Car-type is categorical attribute
  - Class label indicates whether person bought product
  - Dependent attribute is categorical

<table>
<thead>
<tr>
<th></th>
<th>Age</th>
<th>Car</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>M</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>M</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>T</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>S</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>S</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Regression Example
- Example training database
  - Two predictor attributes:
    - Age and Car-type (Sport, Minivan and Truck)
  - Spent indicates how much person spent during a recent visit to the web site
  - Dependent attribute is numerical

<table>
<thead>
<tr>
<th></th>
<th>Age</th>
<th>Car</th>
<th>Spent</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>M</td>
<td>$200</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>M</td>
<td>$150</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>T</td>
<td>$300</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>S</td>
<td>$220</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>S</td>
<td>$400</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>T</td>
<td>$80</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>M</td>
<td>$100</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>M</td>
<td>$125</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>M</td>
<td>$500</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>S</td>
<td>$420</td>
<td></td>
</tr>
</tbody>
</table>
Types of Variables (Review)

- **Numerical**: Domain is ordered and can be represented on the real line (e.g., age, income)
- **Nominal or categorical**: Domain is a finite set without any natural ordering (e.g., occupation, marital status, race)
- **Ordinal**: Domain is ordered, but absolute differences between values is unknown (e.g., preference scale, severity of an injury)

---

Regression Problem

- If $Y$ is numerical, the problem is a *regression problem*.
- $Y$ is called the dependent variable, $d$ is called a *regression function*.
- Take $r$ be record randomly drawn from $P$. Define mean squared error rate of $d$: $RT(d,P) = E(r.Y - d(r.X_1, ..., r.X_d))^2$
- Problem definition: Given dataset $D$ that is a random sample from probability distribution $P$, find regression function $d$ such that $RT(d,P)$ is minimized.

---

Definitions

- Random variables $X_1, ..., X_i$ (*predictor variables*) and $Y$ (*dependent variable*)
- $X_i$ has domain $\text{dom}(X_i)$, $Y$ has domain $\text{dom}(Y)$
- $P$ is a probability distribution on $\text{dom}(X_1) \times ... \times \text{dom}(X_i) \times \text{dom}(Y)$
- Training database $D$ is a random sample from $P$
- A predictor $d$ is a function $d: \text{dom}(X_1) \times ... \times \text{dom}(X_i) \rightarrow \text{dom}(Y)$

---

Goals and Requirements

- **Goals:**
  - To produce an accurate classifier/regression function
  - To understand the structure of the problem
- **Requirements on the model:**
  - High accuracy
  - Understandable by humans, interpretable
  - Fast construction for very large training databases

---

Classification Problem

- If $Y$ is categorical, the problem is a *classification problem*, and we use $C$ instead of $Y$. $|\text{dom}(C)| = J$.
- $C$ is called the *class label*, $d$ is called a *classifier*.
- Take $r$ be record randomly drawn from $P$.
  Define the *classification rate* of $d$: $\text{RT}(d,P) = P(d(r.X_1, ..., r.X_i) \neq r.C)$
- Problem definition: Given dataset $D$ that is a random sample from probability distribution $P$, find classifier $d$ such that $RT(d,P)$ is minimized.

---

What are Decision Trees?

- Decision tree is a classifier in which the *class label* $C$ is given at the leaves and *predictor variables* $X_i$ are given at internal nodes.
- A *decision tree* is a rooted tree.
- A path from root to leaf corresponds to a vector of values $d(r.X_1, ..., r.X_i)$.
- Each node has an associated *attribute* $A$.
- Each edge is associated with an *attribute value* $v$.
- Each leaf node has an associated *class label* $C$.

---

Example of a decision tree:

```
          Age
         /  \  
       <30  >=30
     /     |      
Minivan YES

Sports, Truck

NO
```

---

Copyright © 2000 by Johannes Gehrke
**Decision Trees**

- A decision tree $T$ encodes $d$ (a classifier or regression function) in form of a tree.
- A node $t$ in $T$ without children is called a leaf node. Otherwise $t$ is called an internal node.

**Leaf Nodes**

Consider leaf node $t$

- Classification problem: Node $t$ is labeled with one class label $c$ in $\text{dom}(C)$
- Regression problem: Two choices
  - Piecewise constant model: $t$ is labeled with a constant $y$ in $\text{dom}(Y)$.
  - Piecewise linear model: $t$ is labeled with a linear model $Y = y_1 + \Sigma a_i x_i$

**Internal Nodes**

- Each internal node has an associated splitting predicate. Most common are binary predicates.
- Example predicates:
  - $\text{Age} \leq 20$
  - Profession in \{student, teacher\}
  - $5000*\text{Age} + 3*\text{Salary} - 10000 > 0$

**Example**

- **Encoded classifier:**
  - If ($\text{age} < 30$ and \text{carType}=$\text{Minivan}$) Then YES
  - If ($\text{age} < 30$ and (\text{carType}=\text{Sports} or \text{carType}=\text{Truck})) Then NO
  - If ($\text{age} \geq 30$) Then NO

**Internal Nodes: Splitting Predicates**

- Binary Univariate splits:
  - Numerical or ordered $X$: $X \leq c$, $c$ in $\text{dom}(X)$
  - Categorical $X$: $X$ in $A$, $A$ subset $\text{dom}(X)$
- Binary Multivariate splits:
  - Linear combination split on numerical variables: $\Sigma a_i x_i \leq c$
  - $k$-ary ($k>2$) splits analogous

**Evaluation of Misclassification Error**

Problem:

- In order to quantify the quality of a classifier $d$, we need to know its misclassification rate $RT(d, P)$.
- But unless we know $P$, $RT(d, P)$ is unknown.
- Thus we need to estimate $RT(d, P)$ as good as possible.
Resubstitution Estimate

The *Resubstitution estimate* \( R(d,D) \) estimates \( RT(d,P) \) of a classifier \( d \) using \( D \):
- Let \( D \) be the training database with \( N \) records.
- \( R(d,D) = \frac{1}{N} \sum I(d(r.X) \neq r.C) \)
- Intuition: \( R(d,D) \) is the proportion of training records that is misclassified by \( d \)
- Problem with resubstitution estimate: Overfitting; classifiers that overfit the training dataset will have very low resubstitution error.

Test Sample Estimate

- Divide \( D \) into \( D_1 \) and \( D_2 \)
- Use \( D_1 \) to construct the classifier \( d \)
- Then use resubstitution estimate \( R(d,D_2) \)
  to calculate the estimated misclassification error of \( d \)
- Unbiased and efficient, but removes \( D_2 \)
  from training dataset \( D \)

Cross-Validation: Example

Cross-Validation

- Misclassification estimate obtained through cross-validation is usually nearly unbiased
- Costly computation (we need to compute \( d_1, d_2, \ldots, d_v \)); computation of \( d_i \) is nearly as expensive as computation of \( d \)
- Preferred method to estimate quality of learning algorithms in the machine learning literature

V-fold Cross Validation

Procedure:
- Construct classifier \( d \) from \( D \)
- Partition \( D \) into \( V \) datasets \( D_{v_1}, \ldots, D_{v_V} \)
- Construct classifier \( d_i \) using \( D \setminus D_{v_i} \)
- Calculate the estimated misclassification error \( R(d_i,D_{v_i}) \) of \( d_i \) using test sample \( D_{v_i} \)
- Final misclassification estimate:
  - Weighted combination of individual misclassification errors:
    \[ R(d,D) = \frac{1}{V} \sum R(d, D_{v_i}) \]

Decision Tree Construction

- Top-down tree construction schema:
  - Examine training database and find best splitting predicate for the root node
  - Partition training database
  - Recurse on each child node
Top-Down Tree Construction

**BuildTree** (Node \( t \), Training database \( D \), Split Selection Method \( S \))

1. Apply \( S \) to \( D \) to find splitting criterion
2. if \( (t \) is not a leaf node)
   3. Create children nodes of \( t \)
   4. Partition \( D \) into children partitions
   5. Recurse on each partition
5. endif

Split Selection Method (Contd.)

- Categorical attributes: How to group?
  - Sport: 🏃️‍♂️ 🏁 🚗 Minivan: 🚗 🚗
  - (Sport, Truck) --- (Minivan) 🚗 🚗 🚗
  - (Sport) --- (Truck, Minivan) 🚗 🚗 🚗
  - (Sport, Minivan) --- (Truck) 🚗 🚗 🚗 🚗

Decision Tree Construction

- Three algorithmic components:
  - Split selection (CART, C4.5, QUEST, CHAID, CRUISE, …)
  - Pruning (direct stopping rule, test dataset pruning, cost-complexity pruning, statistical tests, bootstrapping)
  - Data access (CLOUDS, SLIQ, SPRINT, RainForest, BOAT, UnPivot operator)

Pruning Method

- For a tree \( T \), the misclassification rate \( R(T, P) \) and the mean-squared error rate \( R(T, P) \) depend on \( P \), but not on \( D \).
- The goal is to do well on records randomly drawn from \( P \), not to do well on the records in \( D \).
- If the tree is too large, it overfits \( D \) and does not model \( P \). The pruning method selects the tree of the right size.

Split Selection Method

- Numerical or ordered attributes: Find a split point that separates the (two) classes

```
  30  35
  🏃️‍♂️ 🏁 🚗
(Yes: • No: •)
```

Data Access Method

- Recent development: Very large training databases, both in-memory and on secondary storage
- Goal: Fast, efficient, and scalable decision tree construction, using the complete training database.
Split Selection Methods

- Multitude of split selection methods in the literature
- In this workshop:
  - CART

Intuition: Impurity Function

<table>
<thead>
<tr>
<th>X1</th>
<th>X2</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>No</td>
</tr>
</tbody>
</table>

- X1 <= 1 (50%, 50%)  
- X2 <= 1 (50%, 50%)

Impurity Function

- Let \( p(j|t) \) be the proportion of class \( j \) training records at node \( t \)
- Node impurity measure at node \( t \):
  \[ i(t) = \phi(p(1|t), ..., p(3|t)) \]
- \( \phi \) is symmetric
- Maximum value at arguments \( (J^1, ..., J^3) \) (maximum impurity)
- \( \phi(1,0,0) = ... = \phi(0,...,0,1) = 0 \) (node has records of only one class; "pure" node)

CART Split Selection Method

Motivation: We need a way to choose quantitatively between different splitting predicates
- Idea: Quantify the impurity of a node
- Method: Select splitting predicate that generates children nodes with minimum impurity from a space of possible splitting predicates

Example

- Root node \( t \):
  - \( p(1|t) = 0.5 \), \( p(2|t) = 0.5 \)
  - Left child node \( t' \):
    - \( p(1|t') = 0.83 \), \( p(2|t') = -0.17 \)
- Impurity of root node:
  - \( \phi(0.5, 0.5) \)
- Impurity of left child node:
  - \( \phi(0.83, 0.17) \)
- Impurity of right child node:
  - \( \phi(0.0, 1.0) \)
Goodness of a Split

Consider node \( t \) with impurity \( \phi(t) \)
The reduction in impurity through splitting predicate \( s \) (i.e., \( t \) splits into children nodes \( t_L \) and \( t_R \) with impurity \( \phi(t_L) \) and \( \phi(t_R) \)) is:

\[
\Delta_{\phi}(s, t) = \phi(t) - p_L \phi(t_L) - p_R \phi(t_R)
\]

Nonnegative Decrease in Impurity

Theorem: Let \( \phi(p_1, ..., p_J) \) be a strictly concave function on \( j = 1, ..., J, \Sigma_j p_j = 1 \).
Then for any split \( s \):

\[
\Delta_{\phi}(s, t) \geq 0
\]

With equality if and only if:

\[
p(j|t_L) = p(j|t_R) = p(j|t), \quad j = 1, ..., J
\]

Note: Entropy and gini-index are concave.

Example (Contd.)

- Impurity of root node:
  \( \phi(0.5, 0.5) \)
- Impurity of whole tree:
  0.6* \( \phi(0.83, 0.17) \)
  + 0.4* \( \phi(0,1) \)
- Impurity reduction:
  \( \phi(0.5, 0.5) \)
  - 0.6* \( \phi(0.83, 0.17) \)
  - 0.4* \( \phi(0,1) \)

CART Univariate Split Selection

- Use gini-index as impurity function
- For each numerical or ordered attribute \( X \), consider all binary splits \( s \) of the form
  \( X \leq x \)
  where \( x \) in \( \text{dom}(X) \)
- For each categorical attribute \( X \), consider all binary splits \( s \) of the form
  \( X \in A \)
  where \( A \) subset \( \text{dom}(X) \)
- At a node \( t \), select split \( s^* \) such that
  \( \Delta_{\phi}(s^*, t) \) is maximal over all \( s \) considered

Split Selection Methods

Use impurity functions that are concave:

\[ \phi' < 0 \]

Example impurity functions

- Entropy:
  \( \phi(t) = -\Sigma p(j|t) \log(p(j|t)) \)
- Gini index:
  \( \phi(t) = \Sigma p(j|t)^2 \)