Data Mining I: Introduction

Computer Science 105
Boston University
David G. Sullivan, Ph.D.

References for This Part of the Course

• Roiger & Geatz, *Data Mining: A Tutorial-Based Primer* (Addison-Wesley, 2003)

What is Data Mining?

• Informally, it's the process of using a computer program to find patterns or relationships in data.

• Examples:
  • looking for combinations of symptoms that are reliable indicators of a given disease
  • mining a grocery store's customer-purchase data
    • which two products below were found to be frequently purchased together?
      - beer  cereal  diapers
      - milk  soft drinks  toilet paper
    • how could the store make use of this result?

Finding Patterns

• Something that human beings have always done!
  • example: how do we learn to identify a dog?
Finding Patterns (cont.)

- In data mining:
  - the data is stored in electronic form
  - the process is automated (at least in part) by using a computer program
  - the program "mines" the data – "sifting through" it in an attempt to find something useful/valuable

Data Mining vs. Data Query

- Database queries in SQL are not the same thing as data mining.
- Queries allow us to extract factual information.
  - "shallow knowledge"
- In data mining, we attempt to extract patterns and relationships that go beyond mere factual information.
  - "hidden knowledge"
Machine Learning

• In data mining, we apply an algorithm that "learns" something about the data.

• These algorithms are referred to as *machine-learning algorithms*.

• There are several different types of machine learning that are used in data mining.
  • classification learning
  • association learning
  • numeric estimation
  • clustering

• We will limit ourselves to the first three types.

Classification Learning

• Classification learning involves learning how to classify objects/entities on the basis of their characteristics.
  • example: how to classify credit-card purchases as fraudulent or non-fraudulent

• Input to the algorithm = a set of data describing objects that have already been classified.
  • known as *training data or training examples*

• Output = a *model* that can be used to classify other objects.
  • different algorithms produce different types of models
Example: Medical Diagnosis

- Given a set of symptoms, we want to be able to determine a correct diagnosis for a patient with cold-like symptoms.

- Sample training data (table 1-1 of Roiger & Geatz):

<table>
<thead>
<tr>
<th>Patient ID#</th>
<th>Sore Throat</th>
<th>Swollen Glands</th>
<th>Congestion</th>
<th>Headache</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Strep throat</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Strep throat</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Strep throat</td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Cold</td>
</tr>
</tbody>
</table>

- Can you see any patterns that would help you diagnose patients with one or more of these symptoms?

Example: Medical Diagnosis (cont.)

- One possible model that could be used for classifying other patients is a set of rules like the following:

  \[ \text{if Swollen Glands == Yes} \quad \text{then Diagnosis = Strep Throat} \]
  \[ \text{if Swollen Glands == No and Fever == Yes} \quad \text{then Diagnosis = Cold} \]
  \[ \text{if Swollen Glands == No and Fever == No} \quad \text{then Diagnosis = Allergy} \]
Example: Medical Diagnosis (cont.)

<table>
<thead>
<tr>
<th>Patient ID#</th>
<th>Sore Throat</th>
<th>Fever</th>
<th>Swollen Glands</th>
<th>Congestion</th>
<th>Headache</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Strep throat</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Cold</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Strep throat</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Cold</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Allergy</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Strep throat</td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
</tr>
</tbody>
</table>

• Another possible type of model is known as a decision tree:
  • start at the top and work down until you reach a box containing a classification
  • what diagnosis would the tree give for patient 11 above?

Some Terminology

• Each row in a collection of training data is known as an example or instance.

• Each column is referred to as an attribute.

• The attributes can be divided into two types:
  • the output attribute – the one we want to determine/predict
  • the input attributes – everything else

  input attributes → model → output attribute

• In our example:

  fever
  swollen glands
  headache

  rules or
tree
or…

  diagnosis
Types of Attributes

- *Nominal* attributes have values that are "names" of categories.
  - there is a small set of possible values
    
    | attribute       | possible values                      |
    |-----------------|--------------------------------------|
    | Fever           | {Yes, No}                            |
    | Diagnosis       | {Allergy, Cold, Strep Throat}        |

- In classification learning, the output attribute is always *nominal*.

- *Numeric* attributes have values that come from a range of #s.
  
<table>
<thead>
<tr>
<th>attribute</th>
<th>possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Temp</td>
<td>any value in 96.0-106.0</td>
</tr>
<tr>
<td>Salary</td>
<td>any value in $15,000-250,000</td>
</tr>
</tbody>
</table>

- you can *order* their values
  - $210,000 > $125,000
  - $98.6 < $101.3

Types of Attributes (cont.)

- What about this one?
  
<table>
<thead>
<tr>
<th>attribute</th>
<th>possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Type</td>
<td>{0, 1, 2, 3}</td>
</tr>
</tbody>
</table>

- If numbers are used as IDs or names of categories, the corresponding attribute is actually nominal.

- Note that it doesn't make sense to order the values of such attributes.
  - example: product type 2 > product type 1 doesn't have any meaning
Numeric Estimation

- Like classification learning, but for a *numeric* output attribute.
  - example: a charity that needs to decide who should be sent a fundraising appeal letter

\[
\text{model} \rightarrow \text{probability of response} \quad \text{(a number between 0 and 1)}
\]

- The model learned by the algorithm often takes the form of an equation.
  
  \[
  \text{probability of reply} = 0.424 \text{attr1} - 0.072 \text{attr2} + \ldots
  \]
  
  where \( \text{attr1}, \text{attr2}, \ldots \) are attributes

- Linear regression is a form of numeric estimation.

Association Learning

- In association learning, the algorithm looks for relationships between sets of attributes in the training examples.
  - produces a set of rules
  - for example:
    
    \[
    \begin{align*}
    \text{if Congestion} &= \text{Yes} \\
    \text{then Headache} &= \text{Yes} \\
    \text{if Sore Throat} &= \text{Yes} \quad \text{and Swollen Glands} = \text{No} \\
    \text{then Congestion} &= \text{Yes} \quad \text{and Fever} = \text{No}
    \end{align*}
    \]

- Unlike classification learning and numeric estimation, association learning does *not* focus on predicting a particular attribute.
  - no distinction between input and output attributes
Association Learning (cont.)

- One form of association learning is market-basket analysis, which finds associations between items that people buy.
  - classic example: beer and diapers on Thursdays!

- Association learning is often more difficult than classification learning. Why do you think that is?

Example: Data About Investors

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Account Type</th>
<th>Margin Account</th>
<th>Transaction Method</th>
<th>Trades/ Month</th>
<th>Sex</th>
<th>Age</th>
<th>Favorite Recreation</th>
<th>Annual Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>1005</td>
<td>Joint</td>
<td>No</td>
<td>Online</td>
<td>12.5</td>
<td>F</td>
<td>30–39</td>
<td>Tennis</td>
<td>40–59K</td>
</tr>
<tr>
<td>1013</td>
<td>Custodial</td>
<td>No</td>
<td>Broker</td>
<td>0.5</td>
<td>F</td>
<td>50–59</td>
<td>Skiing</td>
<td>80–99K</td>
</tr>
<tr>
<td>1245</td>
<td>Joint</td>
<td>No</td>
<td>Online</td>
<td>3.6</td>
<td>M</td>
<td>20–29</td>
<td>Golf</td>
<td>20–39K</td>
</tr>
<tr>
<td>2110</td>
<td>Individual</td>
<td>Yes</td>
<td>Broker</td>
<td>22.3</td>
<td>M</td>
<td>30–39</td>
<td>Fishing</td>
<td>40–59K</td>
</tr>
<tr>
<td>1001</td>
<td>Individual</td>
<td>Yes</td>
<td>Online</td>
<td>5.0</td>
<td>M</td>
<td>40–49</td>
<td>Golf</td>
<td>60–78K</td>
</tr>
</tbody>
</table>

(Table 1-3 in Roiger & Geatz)

- Why might we want to perform classification learning on this type of data?
  - what are some possible class attributes?
    - 
    - 

- What about numeric estimation?
Summary of Machine-Learning Approaches

- **classification learning**: takes a set of already classified training examples and learns a model that can be used to classify previously unseen examples

  ```
  if Swollen Glands = Yes
  then Diagnosis = Strep Throat
  
  if Swollen Glands = No and Fever = Yes
  then Diagnosis = Cold
  ...
  ```

- **numeric estimation**: like classification learning, but the output attribute is numeric
  - the model is typically in the form of an equation

  ```
  Patient Sore Swollen ID# Throat Fever Glands Congestion Headache Diagnosis
  1 Yes Yes Yes Yes Yes Strep throat
  2 No No No Yes Yes Allergy
  3 Yes Yes No Yes No Cold
  4 Yes No Yes No No Strep throat
  5 No Yes No Yes No Cold
  6 No No No Yes No Allergy
  ... ...
  ```

- **association learning**: takes a set of training examples and discovers associations among attributes
  - we don't specify a single class/output attribute

  ```
  if Congestion = Yes
  then Headache = Yes
  
  if Sore Throat = Yes and Swollen Glands = No
  then Congestion = Yes and Fever = No
  ```

  ```
  ... ...
  ```
Evaluating the Results

- For most non-trivial, real-world data sets, no learned model is likely to work perfectly on all possible examples.
  - the concepts being modelled are complicated
  - they may depend on factors for which we don't have data
  - there may be noise in our training data – imprecise or inaccurate values

- Our goal is not to create a model that perfectly matches the training data.

- Instead, we want a model that performs well on previously unseen examples.
  - we say that we want the model to generalize

Another Example: Labor Negotiations

- Goal: to be able to predict whether a labor contract will be acceptable to the union and to management.
  - class = good (acceptable) or bad (unacceptable)

- Training data = 40 examples from actual labor negotiations
  - 17 attributes
  - lots of missing values

- Source of this case study: Witten and Frank
Another Example: Labor Negotiations (cont.)

- Here’s one possible decision tree based on the training data:
  - simple model
  - makes intuitive sense
  - misclassifies some of the training examples

![Decision Tree 1]

Another Example: Labor Negotiations (cont.)

- Here’s another possible decision tree from the same data:

![Decision Tree 2]

- It does a better job classifying the training examples.
- However, it may not do as well on previously unseen examples.
  - it may not generalize as well as the simpler model
Overfitting

- In general, working too hard to match the training examples can lead to an overly complicated model that doesn't generalize well.
- This problem is known as overfitting the training data.
- Extreme example of overfitting: memorize the training examples!
  - example: store all 10 training examples from the diagnosis problem in a table
  - when a new patient comes in, see if his/her symptoms match any of the training examples and use that diagnosis
  - why won't this work?

Overfitting (cont.)

- Overfitting can also happen in numeric estimation.

- Example:

  ![Graphs showing overfitting](image)

  - points = training examples
  - $h(x)$ = the actual function we are trying to estimate
  - $g(x)$ underfits the training data – it's too simple
  - $j(x)$ overfits the training data – it's too complicated
  - neither $g(x)$ nor $j(x)$ will do well on unseen examples
Evaluating Classification Learning

• To test how well a model generalizes, we typically withhold some of the examples as test examples.
  • these examples are not used to train the model

• Example: decision-tree model for medical diagnosis
  • trained using the 10 earlier examples
  • we can test it using the examples shown below

```
<table>
<thead>
<tr>
<th>Patient ID#</th>
<th>Sore Throat</th>
<th>Fever</th>
<th>Swollen Glands</th>
<th>Congestion</th>
<th>Headache</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Strep throat</td>
</tr>
<tr>
<td>13</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Cold</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Strep throat</td>
</tr>
<tr>
<td>15</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
<tr>
<td>16</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Cold</td>
</tr>
<tr>
<td>17</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Allergy</td>
</tr>
</tbody>
</table>
```

Model's Diagnosis

```
17 Yes No No No Yes Allergy
```

Evaluating Classification Learning (cont.)

• The error rate of a model is the percentage of test examples that it misclassifies.
  • in our example, the error rate = 2/6 = 33.3 %
  • the model's accuracy = 100 – error rate

• Problem: these metrics treat all misclassifications as being equal.
  • this isn't always the case
  • example: more problematic to misclassify strep throat than to misclassify a cold or allergy
Evaluating Classification Learning (cont.)

To provide a more detailed picture of the model's accuracy, we can use a confusion matrix:

<table>
<thead>
<tr>
<th>Predicted Class</th>
<th>Cold</th>
<th>Allergy</th>
<th>Strep Throat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Class:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cold</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Allergy</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Strep Throat</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

- The diagonal of the matrix shows cases that were correctly classified.

Interpreting a Confusion Matrix

Let's say that we had a larger number of test examples, and that we obtained the following confusion matrix:

<table>
<thead>
<tr>
<th>Actual Class:</th>
<th>Cold</th>
<th>Allergy</th>
<th>Strep Throat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cold</td>
<td>25</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Allergy</td>
<td>6</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>Strep Throat</td>
<td>5</td>
<td>4</td>
<td>33</td>
</tr>
</tbody>
</table>

- What is the accuracy of the model?

- What is its error rate?
Interpreting a Confusion Matrix (cont.)

<table>
<thead>
<tr>
<th>predicted class</th>
<th>cold</th>
<th>allergy</th>
<th>strep throat</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual class:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cold</td>
<td>25</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>allergy</td>
<td>6</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>strep throat</td>
<td>5</td>
<td>4</td>
<td>33</td>
</tr>
</tbody>
</table>

- how many test cases of strep throat are there?

- how many actual colds were misdiagnosed?

- what percentage of actual colds were correctly diagnosed?

Two-Class Confusion Matrices

- When there are only two classes, the classification problem is often framed as a yes / no judgement:
  - yes / no
  - fraudulent / not fraudulent
  - has cancer / doesn't have cancer

  The terms *positive* / *negative* are often used in place of yes / no.

- In such cases, there are four possible types of classifications:
  - *true positive (TP)*: the model correctly predicts "yes"
  - *false positive (FP)*: the model incorrectly predicts "yes"
  - *true negative (TN)*: the model correctly predicts "no"
  - *false negative (FN)*: the model incorrectly predicts "no"

<table>
<thead>
<tr>
<th>predicted</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>no</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>
Comparing Models Using Confusion Matrices

• Let's say we're trying to detect credit-card fraud.

• We use two different classification-learning techniques and get two different models.

• Performance on 400 test examples:

<table>
<thead>
<tr>
<th>predicted by model A</th>
<th>fraud</th>
<th>not fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual: fraud</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>not fraud</td>
<td>40</td>
<td>250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>predicted by model B</th>
<th>fraud</th>
<th>not fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual: fraud</td>
<td>80</td>
<td>30</td>
</tr>
<tr>
<td>not fraud</td>
<td>20</td>
<td>270</td>
</tr>
</tbody>
</table>

• which model is better?

Overall Accuracy Isn't Enough

• Someone tells you that they have a fraud-detection classifier with an overall accuracy of 99%. Should you use it?

• It depends on the test examples used to compute the accuracy!

• Example:
  • assume 1% of actual credit-card purchases are fraudulent
  • assume the test examples reflect this:
    • 10 examples of fraud, 990 examples of not fraud
  • on these examples, a model can be 99% accurate by always predicting "not fraud"!

<table>
<thead>
<tr>
<th>predicted</th>
<th>fraud</th>
<th>not fraud</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual: fraud</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>not fraud</td>
<td>0</td>
<td>990</td>
</tr>
</tbody>
</table>
Overall Accuracy Isn’t Enough (cont.)

- Test examples should include an adequate number of all possible classifications.
  - especially ones you're most concerned about getting right
  - in our example, need to include enough examples of fraud

- It's also important that your training examples include all possible classifications.

- If you're primarily concerned about getting one of the classifications right, it may make sense to artificially inflate the number of examples of that class in the training examples.