Data Mining
Practical Machine Learning Tools and Techniques

Slides for Chapter 3 of *Data Mining* by I. H. Witten, E. Frank and M. A. Hall
Output: Knowledge representation

- Tables
- Linear models
- Trees
- Rules
  - Classification rules
  - Association rules
  - Rules with exceptions
  - More expressive rules
- Instance-based representation
- Clusters
Many different ways of representing patterns
  - Decision trees, rules, instance-based, …
Also called “knowledge” representation
Representation determines inference method
Understanding the output is the key to understanding the underlying learning methods
Different types of output for different learning problems (e.g. classification, regression, …)

Output: representing structural patterns
Tables

• Simplest way of representing output:
  • Use the same format as input!

• Decision table for the weather problem:

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Humidity</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Normal</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Normal</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Rainy</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

• Main problem: selecting the right attributes
Linear models

- Another simple representation
- Regression model
  - Inputs (attribute values) and output are all numeric
- Output is the sum of weighted attribute values
  - The trick is to find good values for the weights
A linear regression function for the CPU performance data

\[ PRP = 37.06 + 2.47CACH \]
Linear models for classification

- Binary classification
- Line *separates* the two classes
  - Decision boundary - defines where the decision changes from one class value to the other
- Prediction is made by plugging in observed values of the attributes into the expression
  - Predict one class if output $\geq 0$, and the other class if output $< 0$
- Boundary becomes a high-dimensional plane (*hyperplane*) when there are multiple attributes
Separating setosas from versicolors

\[ 2.0 - 0.5\text{PETAL-LENGTH} - 0.8\text{PETAL-WIDTH} = 0 \]
Trees

- “Divide-and-conquer” approach produces tree
- Nodes involve testing a particular attribute
- Usually, attribute value is compared to constant
- Other possibilities:
  - Comparing values of two attributes
  - Using a function of one or more attributes
- Leaves assign classification, set of classifications, or probability distribution to instances
- Unknown instance is routed down the tree
Nominal and numeric attributes

• **Nominal:**
  number of children usually equal to number values
  ⇒ attribute won’t get tested more than once
  • Other possibility: division into two subsets

• **Numeric:**
  test whether value is greater or less than constant
  ⇒ attribute may get tested several times
  • Other possibility: three-way split (or multi-way split)
    • Integer: *less than, equal to, greater than*
    • Real: *below, within, above*
Missing values

- Does absence of value have some significance?
- Yes ⇒ “missing” is a separate value
- No ⇒ “missing” must be treated in a special way
  - Solution A: assign instance to most popular branch
  - Solution B: split instance into pieces
    - Pieces receive weight according to fraction of training instances that go down each branch
    - Classifications from leave nodes are combined using the weights that have percolated to them
Trees for numeric prediction

- **Regression**: the process of computing an expression that predicts a numeric quantity
- **Regression tree**: “decision tree” where each leaf predicts a numeric quantity
  - Predicted value is average value of training instances that reach the leaf
- **Model tree**: “regression tree” with linear regression models at the leaf nodes
  - Linear patches approximate continuous function
Linear regression for the CPU data

PRP =
- 56.1
+ 0.049 MYCT
+ 0.015 MMIN
+ 0.006 MMAX
+ 0.630 CACH
- 0.270 CHMIN
+ 1.46 CHMAX
Regression tree for the CPU data

```
ChMin
  <= 7.5
  > 7.5
    Cach
      <= 8.5
      (8.5, 28]
      > 28
        Mmax
          <= 28000
          > 28000
            CHMax
              <= 58
              > 58
                Mmin
                  783 (5/359%)
        Mmax
          <= 10000
          > 10000
            MYCT
              <= 0.5
              (0.5, 8.5]
              > 0.5
                Cach
                  <= 550
                  > 550
                    MYCT
                      37.3 (19/11.3%)
                      18.3 (7/3.83%)
            133 (16/28.8%)
            281 (11/56%)
            492 (7/53.9%)
        64.6 (24/19.2%)
        59.3 (24/16.9%)
```
Model tree for the CPU data

```
Model tree for the CPU data

CHMIN
  <= 7.5  > 7.5
  /     \
CACH   MMAX
  <= 8.5  > 8.5
  /     \
MMAX   LM4 (50/22.1%)
  <= 4250 > 4250
  /     \
LM1 (65/7.32%) CACH
  <= 0.5 (0.5, 8.5]
  /     \
LM2 (26/6.37%)  LM3 (24/14.5%)
  /     \
LM5 (21/45.5%) LM6 (23/63.5%)
  <= 28000 > 28000
```

Classification rules

- Popular alternative to decision trees
- Antecedent (pre-condition): a series of tests (just like the tests at the nodes of a decision tree)
- Tests are usually logically ANDed together (but may also be general logical expressions)
- Consequent (conclusion): classes, set of classes, or probability distribution assigned by rule
- Individual rules are often logically ORed together
  - Conflicts arise if different conclusions apply
From trees to rules

- **Easy: converting a tree into a set of rules**
  - One rule for each leaf:
    - Antecedent contains a condition for every node on the path from the root to the leaf
    - Consequent is class assigned by the leaf
- **Produces rules that are unambiguous**
  - Doesn’t matter in which order they are executed
- **But: resulting rules are unnecessarily complex**
  - Pruning to remove redundant tests/rules
From rules to trees

• More difficult: transforming a rule set into a tree
  • Tree cannot easily express disjunction between rules
• Example: rules which test different attributes
  
  If a and b then x
  If c and d then x

• Symmetry needs to be broken
• Corresponding tree contains identical subtrees
  (⇒ “replicated subtree problem”)


A tree for a simple disjunction
The exclusive-or problem

If $x = 1$ and $y = 0$ then class = a
If $x = 0$ and $y = 1$ then class = a
If $x = 0$ and $y = 0$ then class = b
If $x = 1$ and $y = 1$ then class = b
A tree with a replicated subtree

If $x = 1$ and $y = 1$
then class = a

If $z = 1$ and $w = 1$
then class = a

Otherwise class = b
“Nuggets” of knowledge

- Are rules independent pieces of knowledge? (It seems easy to add a rule to an existing rule base.)
- Problem: ignores how rules are executed
- Two ways of executing a rule set:
  - Ordered set of rules ("decision list")
    - Order is important for interpretation
  - Unordered set of rules
    - Rules may overlap and lead to different conclusions for the same instance
Interpreting rules

- What if two or more rules conflict?
  - Give no conclusion at all?
  - Go with rule that is most popular on training data?
  - …

- What if no rule applies to a test instance?
  - Give no conclusion at all?
  - Go with class that is most frequent in training data?
  - …
Special case: boolean class

- Assumption: if instance does not belong to class “yes”, it belongs to class “no”
- Trick: only learn rules for class “yes” and use default rule for “no”

\[
\begin{align*}
\text{If } x = 1 \text{ and } y = 1 \text{ then } \text{class} = a \\
\text{If } z = 1 \text{ and } w = 1 \text{ then } \text{class} = a \\
\text{Otherwise } \text{class} = b
\end{align*}
\]

- Order of rules is not important. No conflicts!
- Rule can be written in *disjunctive normal form*
Association rules

- Association rules...
  - ... can predict any attribute and combinations of attributes
  - ... are not intended to be used together as a set
- Problem: immense number of possible associations
  - Output needs to be restricted to show only the most predictive associations \( \Rightarrow \) only those with high support and high confidence
Support and confidence of a rule

- Support: number of instances predicted correctly
- Confidence: number of correct predictions, as proportion of all instances that rule applies to
- Example: 4 cool days with normal humidity

If temperature = cool then humidity = normal

⇒ Support = 4, confidence = 100%

- Normally: minimum support and confidence pre-specified (e.g. 58 rules with support ≥ 2 and confidence ≥ 95% for weather data)
Interpreting association rules

• Interpretation is not obvious:

If windy = false and play = no then outlook = sunny
and humidity = high

is *not* the same as

If windy = false and play = no then outlook = sunny
If windy = false and play = no then humidity = high

• It means that the following also holds:

If humidity = high and windy = false and play = no
then outlook = sunny
Rules with exceptions

• Idea: allow rules to have *exceptions*
• Example: rule for iris data

If petal-length $\geq 2.45$ and petal-length $< 4.45$ then Iris-versicolor

• New instance:

<table>
<thead>
<tr>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>2.6</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
</tbody>
</table>

• Modified rule:

If petal-length $\geq 2.45$ and petal-length $< 4.45$ then Iris-versicolor  
EXCEPT if petal-width $< 1.0$ then Iris-setosa
A more complex example

• Exceptions to exceptions to exceptions …

default: Iris-setosa

except if petal-length $\geq 2.45$ and petal-length $< 5.355$
   and petal-width $< 1.75$
   then Iris-versicolor

   except if petal-length $\geq 4.95$ and petal-width $< 1.55$
   then Iris-virginica

   else if sepal-length $< 4.95$ and sepal-width $\geq 2.45$
   then Iris-virginica

else if petal-length $\geq 3.35$
   then Iris-virginica

   except if petal-length $< 4.85$ and sepal-length $< 5.95$
   then Iris-versicolor
Advantages of using exceptions

- Rules can be updated incrementally
  - Easy to incorporate new data
  - Easy to incorporate domain knowledge
- People often think in terms of exceptions
- Each conclusion can be considered just in the context of rules and exceptions that lead to it
  - Locality property is important for understanding large rule sets
  - “Normal” rule sets don’t offer this advantage
More on exceptions

- Default...except if...then...
  is logically equivalent to
  if...then...else
  (where the else specifies what the default did)

- But: exceptions offer a psychological advantage
  - Assumption: defaults and tests early on apply more widely than exceptions further down
  - Exceptions reflect special cases
Rules involving relations

- So far: all rules involved comparing an attribute-value to a constant (e.g. temperature < 45)
- These rules are called “propositional” because they have the same expressive power as propositional logic
- What if problem involves relationships between examples (e.g. family tree problem from above)?
  - Can’t be expressed with propositional rules
  - More expressive representation required
The shapes problem

- Target concept: *standing up*
- Shaded: *standing*
  Unshaded: *lying*
A propositional solution

<table>
<thead>
<tr>
<th>Width</th>
<th>Height</th>
<th>Sides</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>Standing</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>Standing</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>Lying</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>3</td>
<td>Standing</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>3</td>
<td>Lying</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>4</td>
<td>Standing</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>4</td>
<td>Lying</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td>Lying</td>
</tr>
</tbody>
</table>

*If width ≥ 3.5 and height < 7.0 then lying*

*If height ≥ 3.5 then standing*
A relational solution

- Comparing attributes with each other
  - If width > height then lying
  - If height > width then standing

- Generalizes better to new data
- Standard relations: =, <, >
- But: learning relational rules is costly
- Simple solution: add extra attributes
  (e.g. a binary attribute is width < height?)
Rules with variables

- Using variables and multiple relations:
  
  If height_and_width_of(x,h,w) and h > w
  then standing(x)

- The top of a tower of blocks is standing:
  
  If height_and_width_of(x,h,w) and h > w
  and is_top_of(y,x)
  then standing(x)

- The whole tower is standing:
  
  If is_top_of(x,z) and
  height_and_width_of(z,h,w) and h > w
  and is_rest_of(x,y) and standing(y)
  then standing(x)
  If empty(x) then standing(x)

- Recursive definition!
Inductive logic programming

• Recursive definition can be seen as logic program
• Techniques for learning logic programs stem from the area of “inductive logic programming” (ILP)
• But: recursive definitions are hard to learn
  ♦ Also: few practical problems require recursion
  ♦ Thus: many ILP techniques are restricted to non-recursive definitions to make learning easier
Instance-based representation

- Simplest form of learning: *rote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning
- Similarity function defines what’s “learned”
- Instance-based learning is *lazy* learning
- Methods: *nearest-neighbor, k-nearest-neighbor*, ...
The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary
Learning prototypes

- Only those instances involved in a decision need to be stored
- Noisy instances should be filtered out
- Idea: only use *prototypical* examples
Rectangular generalizations

- Nearest-neighbor rule is used outside rectangles
- Rectangles are rules! (But they can be more conservative than “normal” rules.)
- Nested rectangles are rules with exceptions
Representing clusters I

**Simple 2-D representation**

**Venn diagram**

Overlapping clusters
Representing clusters II

**Probabilistic assignment**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.4</td>
<td>0.1</td>
<td>0.5</td>
</tr>
<tr>
<td>b</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>c</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>d</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>e</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>f</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
</tr>
<tr>
<td>g</td>
<td>0.7</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>h</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dendrogram**

NB: dendron is the Greek word for tree