Data Mining Output: Knowledge Representation

Chapter 3 of Data Mining

Output: Knowledge representation

- Tables
- Linear models
- Trees
- Rules
  - Classification rules
  - Association rules
  - Rules with exceptions
- Instance-based representation
- Clusters

Output: representing structural patterns

- Many different ways of representing patterns
  - Decision trees, rules, …
- Also called "knowledge" representation
- Representation determines inference method
  - Algorithm is targeted to a specific output
- Understanding the output is the key to understanding the underlying learning methods
- Different types of output for different learning problems (e.g. classification, numeric estimation, …)

Tables

- Simplest way of representing output:
  - Use the same format as input!
- Decision table for the weather problem:

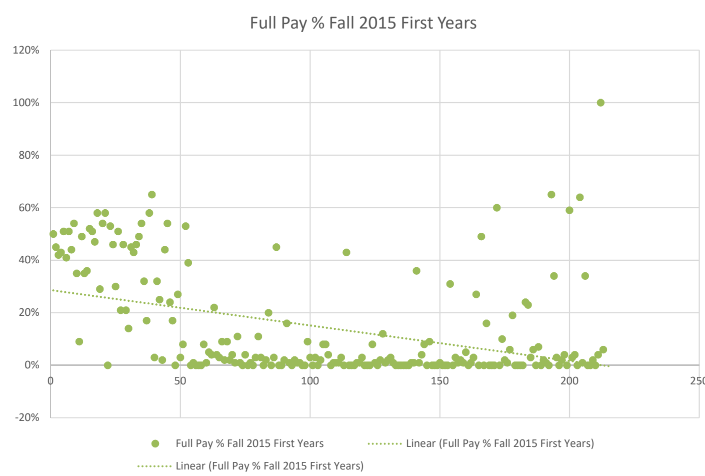
Outlook	Humidity	Play
Sunny	High	No
Sunny	Normal	Yes
Overcast	High	Yes
Overcast	Normal	Yes
Rainy	High	No
Rainy	Normal	No

- Main problem: selecting the right attributes
  - Have to experiment to decide which are not relevant to the concept to be learned

Linear models

- Another simple representation
- Also called a *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 that give a good fit to the training data
- Easiest to visualize in two dimensions
  - Straight line drawn through the data points represents the regression model / function

An interesting linear regression function

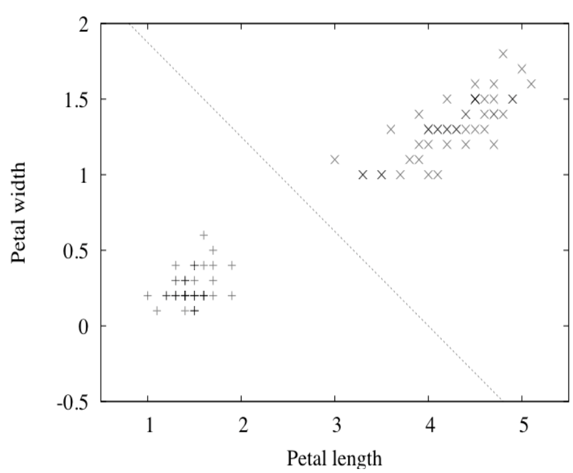


`FullPayPct = 29.73 – 0.13*Ranking`

Linear models for classification

- Can be applied to binary classification
  - Not only numerical estimation
- 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 ≥ 0, and the other class if output < 0
- Boundary becomes a high-dimensional plane (*hyperplane*) when there are multiple attributes

Separating setosas from versicolors

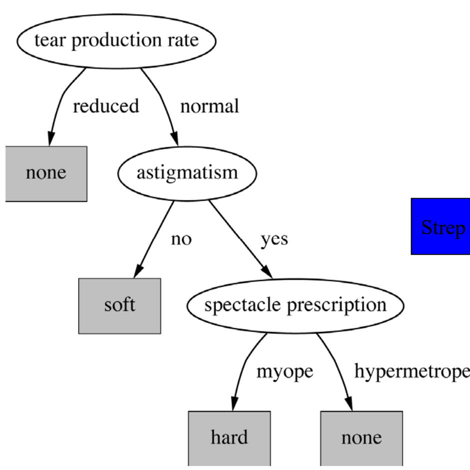


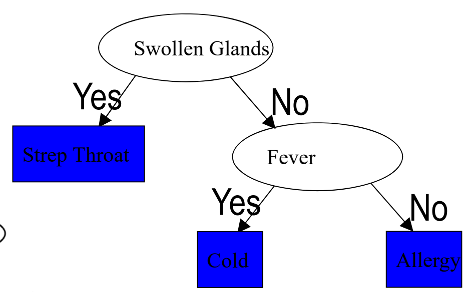
`2.0 – 0.5PETAL-LENGTH – 0.8PETAL-WIDTH = 0`

Trees

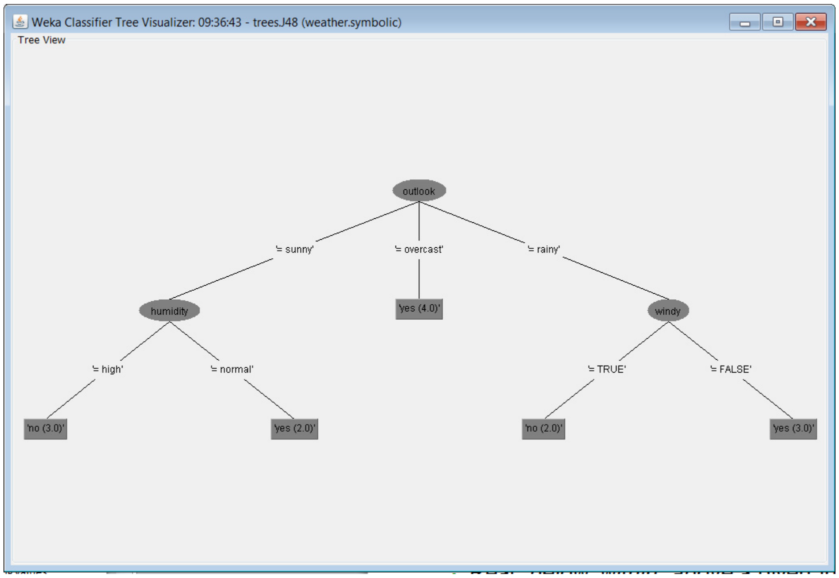
- "Divide-and-conquer" approach produces tree
  - Value of one attribute rules out certain classes
- Nodes involve testing a particular attribute
- Usually, attribute value is compared to constant
- Other possibilities:
  - Comparing values of two attributes
  - Using a function of one or more attributes
- Leaves assign classification, set of classifications, or probability distribution to instances
- New, unknown instance is routed down the tree for classification / prediction

Trees





Trees



Nominal and numeric attributes

- Nominal:
  number of children usually equal to number of values
  ⇒ attribute typically 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* a given interval
    - Sometimes missing values require their own split

Missing values

- Does absence of value have some significance?
- Yes ⇒ "missing" is ideally a separate value
  - Must be factored in when building the model (see Chapter 2 notes)
- 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 leaf 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 the class attribute) of training instances that reach the leaf
- *Model tree:* combine regression tree with linear regression equations at the leaf nodes
  - Linear patches approximate continuous function

Linear regression formula

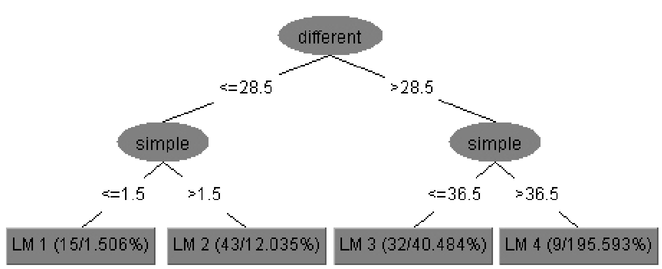
- Consider a dataset containing video game reviews
- Each instance represents the frequency of words used collectively in all reviews
-

```
amazing = 0.1224 * different + 0.4284 * realistic
          + 0.5013 * original + 0.496 * simple
          + 0.6497 * stupid + 0.6423 * repetitive
          - 3.3661 * lame - 5.5266
```

- How many times would the model predict the word *amazing* is used in reviews of this game?

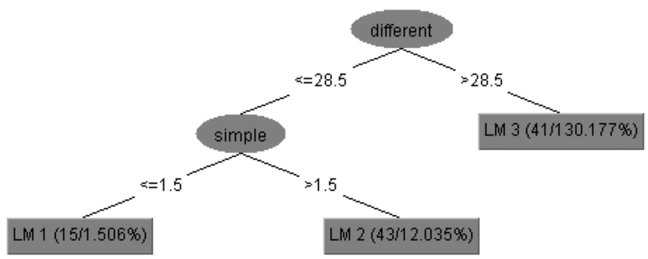
different	realistic	original	challenging	simple	stupid	repetitive	lame	amazing
53	14	16	18	12	5	3	1	???

Regression tree




```
Where:
  LM1 = 8.0525
  LM2 = 10.8965
  LM3 = 21.5446
  LM4 = 46.4725
```

Model tree




```
Where:
      LM1 = 0.1492 * different + 0.1691 * simple + 0.688
      LM2 = 0.1492 * different + 0.0874 * simple + 3.9694
      LM3 = 0.1944 * different + 14.9213
```


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

Classification rules

```
if Swollen Glands == Yes
then Diagnosis = Strep Throat

if Swollen Glands == No and Fever == Yes
then Diagnosis = Cold

if Swollen Glands == No and Fever == No
then Diagnosis = Allergy
```

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity = normal then play = yes
If none of the above then play = yes
```

“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
    - E.g. whether or not to play
  - Unordered set of rules
    - E.g. medical diagnosis, contact lense prescription
    - Rules may overlap and lead to different conclusions for the same instance; sometimes may give no answer

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?
  - …

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
  - Also called *coverage*
- Confidence: number of correct predictions, as proportion of all instances that rule applies to
  - Also called *accuracy*
- Example: 4 cool days with normal humidity

```
If temperature = cool then humidity = normal
```

$\Rightarrow$ Support = 4, confidence = 100%

- Normally: minimum support and confidence pre-specified (e.g. 58 rules with support $\geq$ 2 and confidence $\geq$ 95% for weather data)
  - Support/coverage can also be measured as a percentage of the training instances that the rule applies to

Rules with exceptions

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

```
If petal-length ≥ 2.45 and petal-length < 4.45 then Iris-versicolor
```

- New instance:

Sepal length	Sepal width	Petal length	Petal width	Type
5.1	3.5	2.6	0.2	Iris-setosa

- Modified rule:

```
If petal-length ≥ 2.45 and petal-length < 4.45 then Iris-versicolor
  EXCEPT if petal-width < 1.0 then Iris-setosa
```

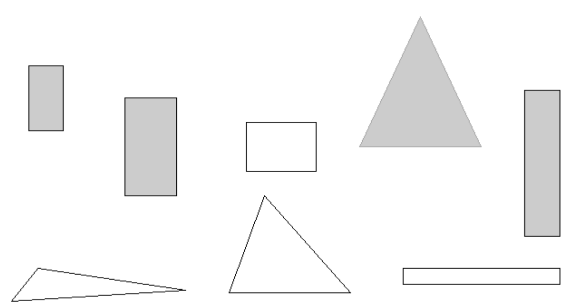
Lesson: Fixing up a rule set is not as simple as it sounds!

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 attributes (e.g. family tree problem, "sister-of")?
  - 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

Width	Height	Sides	Class
2	4	4	Standing
3	6	4	Standing
4	3	4	Lying
7	8	3	Standing
7	6	3	Lying
2	9	4	Standing
9	1	4	Lying
10	2	3	Lying

```
If width ≥ 3.5 and height < 7.0
then lying
If height ≥ 3.5 then standing
```

New instance: width=1, height=2
New instance: width=4, height=6

A relational solution

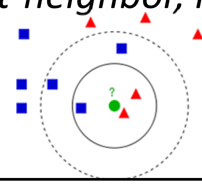
- Comparing attributes with each other

```
If width > height then lying
If height > width then standing
```

- Generalizes better to new data
- Standard relations: =, <, >
- But: learning relational rules is costly
  - Addition of large number of conditions to consider
- Simple solution: add extra attributes (e.g. a binary attribute *is width < height?*)
  - E.g., “sister-of”

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
  - Or eliminating some of them (see Lab 2)

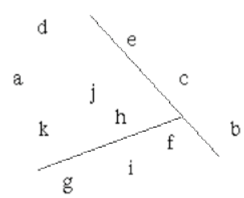
Structural description of patterns?

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

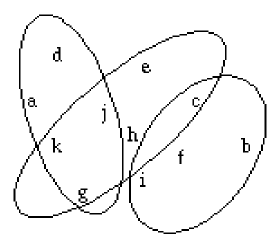
Representing clusters

- For cluster learning, output is a diagram:

Simple 2-D representation



Venn diagram

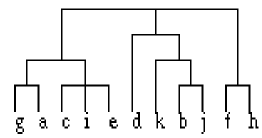


Representing clusters

Probabilistic assignment

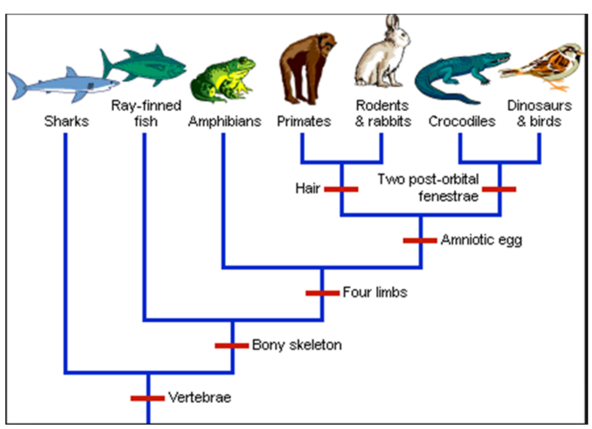
	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
…			

Dendrogram



- Frequent next step: derive a decision tree or rule set that allocates each new instance into a cluster based on the clusters learned

Representing clusters



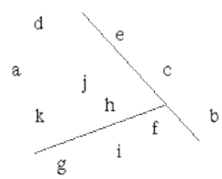
https://www.instituteofcaninebiology.org/how-to-read-a-dendrogram.html

Cluster 1

 # Instances: 3
Sex: Male => 3
Female => 0
Age: 43.3
Credit Card Insurance: Yes => 0
No => 3
Life Insurance Promotion: Yes => 0
No => 3

Cluster 2

 # Instances: 5
Sex: Male => 3
Female => 2
Age: 37.0
Credit Card Insurance: Yes => 1
No => 4
Life Insurance Promotion: Yes => 2
No => 3



Cluster 3

 # Instances: 7
Sex: Male => 2
Female => 5
Age: 39.9
Credit Card Insurance: Yes => 2
No => 5
Life Insurance Promotion: Yes => 7
No => 0

Example: An unsupervised clustering of the credit card ad database. What are the clusters?