F20DL Data Mining and Machine Learning

Diana Bental
(with material from David Corne and slides from http://www.cs.waikato.ac.nz/ml/weka/book.html)

Lecture 5 Output: Knowledge Representations

- We've talked about inputs
  - Concepts, instances, attributes
  - And some practical issues in preparing inputs
    - Normalisation
    - Discretizatiopn (binning)
    - Missing values
    - Data inspection
- Now we look at outputs
  - Knowledge representation

Output: Knowledge Representation

- We have met some knowledge representations, e.g.
- Lecture 2 (and Lab 1)
  - Structural – we can look at the model
  - Tables, rules, decision trees
- Lecture 4
  - Instance-based – the examples are the model
  - 1-nearest-neighbour

Output: Knowledge Representation

- Different learning methods produce different knowledge representations
- The knowledge representation determines what we can do with the model – how we can reason or compute with it

Tables

- Very simple, readable
- Same format as the input!
- Main issue: How to select the most relevant attributes?
- E.g. Decision Table for the Weather problem – ignore temperature because it (usually) makes no difference

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

Linear Models

- Simple
- Regression – attributes are all numeric (including the one we want to predict)
- Output is a sum of weighted attribute values
- Main issue: How to find good values for the weights?
- Linear models for prediction and classification

Linear Regression

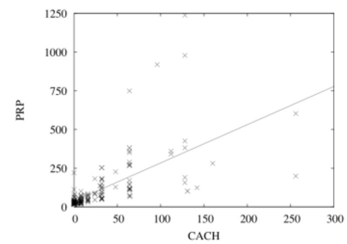
- Input: 209 different computer configurations

	Cycle time (ns)	Main memory (Kb)		Cache (Kb)	Channels		Performance
	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
…							
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

- Model: a linear regression function to predict performance

```
PRP = -55.9 + 0.0489 MYCT + 0.0153 MMIN + 0.0056 MMAX
      + 0.6410 CACH - 0.2700 CHMIN + 1.480 CHMAX
```


Visualise one variable (Cache) and its effect on performance (PRP)



Linear models for (binary) classification

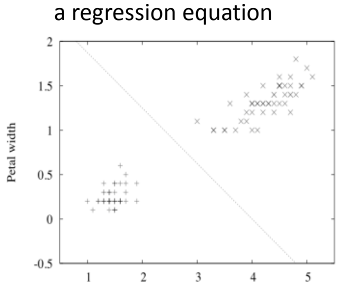
- Binary classification
- Line *separates* the two classes
  - Decision boundary - defines where the decision changes from one class value to the other
- Prediction is made by plugging in observed values of the attributes into the expression
  - Predict one class if output >= 0, and the other class if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

Classifying iris flowers – using rules

	Sepal length	Sepal width	Petal length	Petal width	Type
1	5.1	3.5	1.4	0.2	Iris setosa
2	4.9	3.0	1.4	0.2	Iris setosa
…					
51	7.0	3.2	4.7	1.4	Iris versicolor
52	6.4	3.2	4.5	1.5	Iris versicolor
…					
101	6.3	3.3	6.0	2.5	Iris virginica
102	5.8	2.7	5.1	1.9	Iris virginica
…					

```
If petal length < 2.45 then Iris setosa
If sepal width < 2.10 then Iris versicolor
...
```


Separating setosas from versicolors – using a regression equation

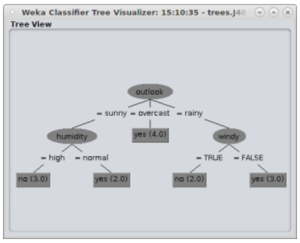



```
2.0 – 0.5PETAL-LENGTH – 0.8PETAL-WIDTH = 0
```


Decision Trees

- Top down "divide and conquer" approach produces a tree
- Each node tests a particular attribute
  - Usually compare the attribute value to a constant
    - One branch per (categorical) value
  - Or Compare two attributes
  - Or Use a function of the attribute(s)
- To classify a new instance:
  - Travel down the tree, testing each attribute
  - Classification at a leaf
  - Classification(s) or probability distribution

Decision tree example – weather data



outlook = sunny, temperature= cool, humidity = high, windy = true, play = ?

Trees for Nominal and Numeric Attributes

- Nominal (categorical) attributes
  - One child per value
  - So each attribute only gets tested once
- Numeric attributes
  - Greater / less than / equal to a constant (3 children)
  - Test for an interval or range (within/ above / below)
  - One attribute may be tested several times, to compare with different values or ranges

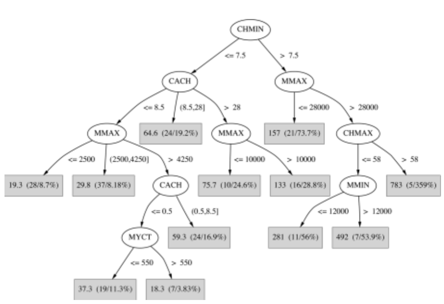
Trees – what about missing values?

- If the absence of a value has a known meaning then need an explicit special value and test for it
- If not then need a process
  - Follow the branch with most instances in the training set
  - Or assign a weight to all possible leaves, depending how many training instances go to each leaf

Using Trees for Numeric Prediction

- Motivation
  - Recall the linear regression equation for the CPU data.....
  - PRP = -56.1 +0.049 x MYCT + 0.015 x MMIN + 0.006 x MMAX + 0.630 x CACH -0.270 x CHMIN + 1.46 x CHMAX
  - But equation is not very accurate
- Regression tree: a “decision tree” where each *leaf* predicts a numeric quantity
  - The predicted value is the average value of the training instances that reach the leaf

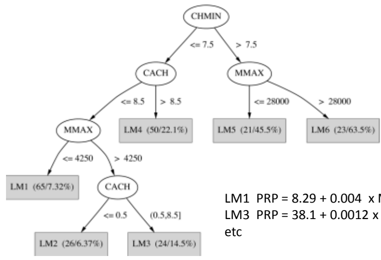
Regression tree for the CPU data



Using Trees for Numeric Prediction

- Model tree: a “decision tree” with **linear regression models** at the leaf nodes
  - Linear patches that approximate the continuous function

Model tree for the CPU data



LM1 PRP = 8.29 + 0.004 x MMAX +2.77 x CHMIN
LM3 PRP = 38.1 + 0.0012 x MMIN
etc

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
```

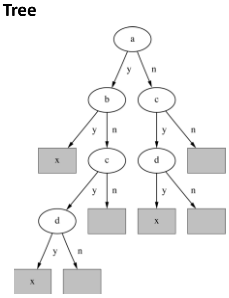
- Symmetry needs to be broken
- Corresponding tree contains identical subtrees ("replicated subtree problem")

A complex tree for a simple disjunction

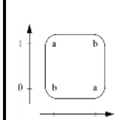
Rules

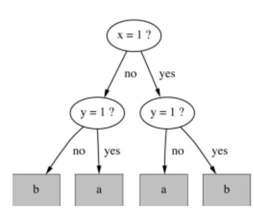
```
If a and b then x
If c and d then x
```

Tree



The *exclusive-or* problem

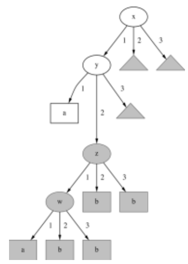





```
If x = 1 and y = 0
   then class = a
If x = 0 and y = 1
   then class = a
If x = 0 and y = 0
   then class = b
If x = 1 and y = 1
   then class = b
```

Another 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 for Rules: Boolean Classification

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

```
If x = 1 and y = 1 then class = a
If z = 1 and w = 1 then class = a
Otherwise class = b
```

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

Association Rules

- Association rules…
  - can predict any attribute
  - can predict combinations of attributes
  - are not intended to be used together as a set
- Problem: immense number of possible associations
  - Which ones?
  - Show only the most predictive associations
  - high support and high confidence

Association rules

Outlook	Temperature	Humidity	Windy	Play
Sunny	Hot	high	false	no
Sunny	Hot	high	true	no
Overcast	Hot	high	false	yes
Rainy	Mild	high	false	yes
Rainy	**Cool**	**normal**	false	yes
Rainy	**Cool**	**normal**	true	no
Overcast	**Cool**	**normal**	true	yes
Sunny	Mild	high	false	no
Sunny	**Cool**	**normal**	false	yes
Rainy	Mild	normal	false	yes
Sunny	Mild	normal	true	yes
Overcast	Mild	high	true	yes
Overcast	Hot	normal	false	yes
Rainy	Mild	high	true	no

Example – weather data

4 **Cool** days with **normal** humidity; no **cool** days with any other humidity

Association rule:

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

Support and Confidence of Rules

- Support: number of instances predicted correctly
- Confidence: number of correct predictions, as a proportion of all instances that rule applies to
- Example:

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

- Support = 4
  - 4 Cool days
- Confidence = 100%
  - All Cool days have normal humidity

Support and Confidence of Rules

- Normally: pre-specify minimum support and confidence
  - E.g. Support >= 2 and confidence >= 95%
- Is this an acceptable rule?

```
If temperature = hot then humidity = high
```

  - Support =
  - Confidence =
- 58 rules with support >= 2 and confidence >= 95% for weather data!

Meaning of Association Rules

- Be careful – the logical meaning of association rules is complex

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

- Does not mean:

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

- Does mean:

```
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 ≥ 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
```


Rules with exceptions

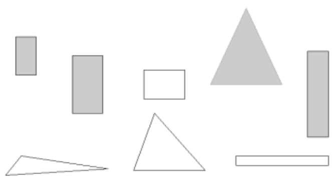
- Can create rules with nested exceptions (exceptions to exceptions)
- Logically equivalent to ordinary rules
  - Default … Except If … Then …
  - If .. Then .. Else …
- Defaults are more common, exceptions are the special cases
- Modular and incremental, useful for understanding large rule sets
- Psychologically plausible

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 **instances**?
  - E.g. family tree problem from earlier lecture
    - Could be denormalised
  - E.g. temperature today < temperature yesterday
    - Can't be denormalized (in any obvious way)
- What if problems involve relationships between **attributes**?
- Can't be expressed with propositional rules
- More expressive representation required

Complex reasoning: 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
```


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?)

More Complications

- 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"

- Relational and recursive rules
- Open research topic
- We're not going there!

So that was rules…

- So far
  - Linear models
  - Rules
  - Trees
- What else can we learn?
  - Instance-based representations
  - Clusters

Instance-based representations

- Rote learning – remember a similar instance
- Look in the training set for instances that are most like the new example
- Knowledge = instances + similarity function
- Simple example: nearest neighbour
- "Lazy" because we only do any work when a new instance comes along to be classified

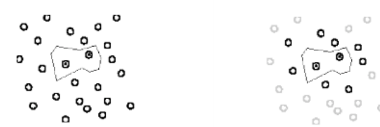
Instance-based representations

- 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

Instance-based representations

- Nearest neighbour kNN – Weka IBk
- Use one nearest neighbour (k=1) or several nearest neighbours
- Classification or regression
  - Classification – same class as nearest neighbour, or largest vote if several neighbours
  - Regression – same value as nearest neighbour or mean if several neighbours
- Nearest neighbour is sensitive to noisy data; more neighbours means less sensitivity to noise but more overlap / fuzziness
- Unbalanced classes: majority voting means likely to get the most common class value; weight the values by distance

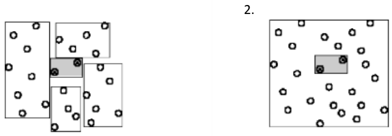
Learning prototypes



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

Rectangular generalizations

1. 2.



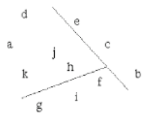
1. Nearest-neighbor rule is used outside the rectangles
- Rectangles are rules! (But they can be more conservative than "normal" rules.)
2. Nested rectangles are rules with exceptions

Clusters

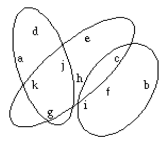
- Identify clusters of similar data
- Unsupervised method
- Simple: Label each instance with the cluster it belongs to

Representing clusters I

Simple 2-D representation



Venn diagram

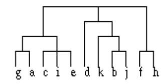


Representing clusters II

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



Knowledge Representation

- Chapter 3 "Data Mining" Witten Frank & Hall
- Next lecture:
  - Statistics
  - Coursework 1