Data Mining with Weka

Class 3 – Lesson 1

Simplicity first!

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.1 Simplicity first!

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 3.1 Simplicity first!

Lesson 3.2 Overfitting

Lesson 3.3 Using probabilities

Lesson 3.4 Decision trees

Lesson 3.5 Pruning decision trees

Lesson 3.6 Nearest neighbor
Lesson 3.1 Simplicity first!

Simple algorithms often work very well!

- There are many kinds of simple structure, eg:
  - One attribute does all the work  
  Lessons 3.1, 3.2
  - Attributes contribute equally and independently  
  Lesson 3.3
  - A decision tree that tests a few attributes  
  Lessons 3.4, 3.5
  - Calculate distance from training instances  
  Lesson 3.6
  - Result depends on a linear combination of attributes  
  Class 4

- Success of method depends on the domain
  - Data mining is an experimental science
Lesson 3.1 Simplicity first!

OneR: One attribute does all the work

❖ Learn a 1-level “decision tree”
  – *i.e.*, rules that all test one particular attribute

❖ Basic version
  – *One branch for each value*
  – *Each branch assigns most frequent class*
  – *Error rate: proportion of instances that don’t belong to the majority class of their corresponding branch*
  – *Choose attribute with smallest error rate*
Lesson 3.1 Simplicity first!

For each attribute,
   For each value of the attribute,
   make a rule as follows:
       count how often each class appears
       find the most frequent class
       make the rule assign that class
           to this attribute-value
   Calculate the error rate of this attribute's rules
Choose the attribute with the smallest error rate
### Lesson 3.1 Simplicity first!

#### Attribute Rules Errors Total errors

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Rules</th>
<th>Errors</th>
<th>Total errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>Sunny → No</td>
<td>2/5</td>
<td>4/14</td>
</tr>
<tr>
<td></td>
<td>Overcast → Yes</td>
<td>0/4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rainy → Yes</td>
<td>2/5</td>
<td></td>
</tr>
<tr>
<td>Temp</td>
<td>Hot → No*</td>
<td>2/4</td>
<td>5/14</td>
</tr>
<tr>
<td></td>
<td>Mild → Yes</td>
<td>2/6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cool → Yes</td>
<td>1/4</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>High → No</td>
<td>3/7</td>
<td>4/14</td>
</tr>
<tr>
<td></td>
<td>Normal → Yes</td>
<td>1/7</td>
<td></td>
</tr>
<tr>
<td>Wind</td>
<td>False → Yes</td>
<td>2/8</td>
<td>5/14</td>
</tr>
<tr>
<td></td>
<td>True → No*</td>
<td>3/6</td>
<td></td>
</tr>
</tbody>
</table>

* indicates a tie
Lesson 3.1 Simplicity first!

Use OneR

- Open file *weather.nominal.arff*
- Choose OneR rule learner (*rules* > *OneR*)
- Look at the rule (*note: Weka runs OneR 11 times*)
Lesson 3.1 Simplicity first!

OneR: One attribute does all the work

- Incredibly simple method, described in 1993
  
  “Very Simple Classification Rules Perform Well on Most Commonly Used Datasets”
  - Experimental evaluation on 16 datasets
  - Used cross-validation
  - Simple rules often outperformed far more complex methods

- How can it work so well?
  - some datasets really are simple
  - some are so small/noisy/complex that nothing can be learned from them!

Course text

- Section 4.1 *Inferring rudimentary rules*
Data Mining with Weka

Class 3 – Lesson 2

Overfitting

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.2 Overfitting

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 3.1 Simplicity first!

Lesson 3.2 Overfitting

Lesson 3.3 Using probabilities

Lesson 3.4 Decision trees

Lesson 3.5 Pruning decision trees

Lesson 3.6 Nearest neighbor
Lesson 3.2 Overfitting

- Any machine learning method may “overfit” the training data ...
  ... by producing a classifier that fits the training data too tightly
- Works well on training data but not on independent test data
- Remember the “User classifier”? Imagine tediously putting a tiny circle around every single training data point
- Overfitting is a general problem
- ... we illustrate it with OneR
Lesson 3.2 Overfitting

Numeric attributes

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>86</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>75</td>
<td>80</td>
<td>False</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- OneR has a parameter that limits the complexity of such rules
- How exactly does it work? Not so important ...

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Rules</th>
<th>Errors</th>
<th>Total errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td>85 → No</td>
<td>0/1</td>
<td>0/14</td>
</tr>
<tr>
<td>80 → Yes</td>
<td>0/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83 → Yes</td>
<td>0/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75 → No</td>
<td>0/1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lesson 3.2 Overfitting

Experiment with OneR

- Open file `weather.numeric.arff`
- Choose OneR rule learner (`rules>OneR`)
- Resulting rule is based on `outlook` attribute, so remove `outlook`
- Rule is based on `humidity` attribute
  
  ```
  humidity:  < 82.5  -> yes
             >= 82.5  -> no
  ```

  (10/14 instances correct)
Lesson 3.2 Overfitting

Experiment with diabetes dataset

- Open file diabetes.arff
- Choose ZeroR rule learner (rules>ZeroR)
- Use cross-validation: 65.1%
- Choose OneR rule learner (rules>OneR)
- Use cross-validation: 72.1%
- Look at the rule (plas = plasma glucose concentration)
- Change minBucketSize parameter to 1: 54.9%
- Evaluate on training set: 86.6%
- Look at rule again
Lesson 3.2 Overfitting

- Overfitting is a general phenomenon that plagues all ML methods
- One reason why you must never evaluate on the training set
- Overfitting can occur more generally
- E.g. try many ML methods, choose the best for your data
  - you cannot expect to get the same performance on new test data
- Divide data into training, test, validation sets?

Course text
- Section 4.1 *Inferring rudimentary rules*
Data Mining with Weka

Class 3 – Lesson 3

Using probabilities

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.3 Using probabilities

- Class 1
  Getting started with Weka

- Class 2
  Evaluation

- Class 3
  Simple classifiers

- Class 4
  More classifiers

- Class 5
  Putting it all together

- Lesson 3.1 Simplicity first!
- Lesson 3.2 Overfitting
- Lesson 3.3 Using probabilities
- Lesson 3.4 Decision trees
- Lesson 3.5 Pruning decision trees
- Lesson 3.6 Nearest neighbor
Lesson 3.3 Using probabilities

(OneR: One attribute does all the work)

Opposite strategy: use *all* the attributes

“Naïve Bayes” method

- Two assumptions: Attributes are
  - *equally important a priori*
  - *statistically independent* *(given the class value)*
    i.e., knowing the value of one attribute says nothing about the value of another *(if the class is known)*

- Independence assumption is never correct!
- But ... often works well in practice
Lesson 3.3 Using probabilities

Probability of event $H$ given evidence $E$

- $\Pr[H]$ is a priori probability of $H$
  - Probability of event before evidence is seen
- $\Pr[H \mid E]$ is a posteriori probability of $H$
  - Probability of event after evidence is seen
- “Naïve” assumption:
  - Evidence splits into parts that are independent

$$
\Pr[H \mid E] = \frac{\Pr[E \mid H \Pr[H]}{\Pr[E]}
$$

Thomas Bayes, British mathematician, 1702 – 1761
### Lesson 3.3 Using probabilities

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>6</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>3</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>3/5</td>
</tr>
</tbody>
</table>

\[
Pr[ H \mid E ] = \frac{Pr[ E_1 \mid H ] Pr[ E_2 \mid H ] \ldots Pr[ E_n \mid H ] Pr[ H ]}{Pr[ E ]}
\]
### Lesson 3.3 Using probabilities

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes  No</td>
<td>Yes  No</td>
<td>Yes  No</td>
<td>Yes  No</td>
</tr>
<tr>
<td>Sunny</td>
<td>2  3</td>
<td>Hot 2  2</td>
<td>High 3  4</td>
<td>False 6  2</td>
</tr>
<tr>
<td>Overcast</td>
<td>4  0</td>
<td>Mild 4  2</td>
<td>Normal 6  1</td>
<td>True 3  3</td>
</tr>
<tr>
<td>Rainy</td>
<td>3  2</td>
<td>Cool 3  1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sunny</td>
<td>2/9  3/5</td>
<td>Hot 2/9 2/5</td>
<td>High 3/9 4/5</td>
<td>False 6/9 2/5</td>
</tr>
<tr>
<td>Overcast</td>
<td>4/9 0/5</td>
<td>Mild 4/9 2/5</td>
<td>Normal 6/9 1/5</td>
<td>True 3/9 3/5</td>
</tr>
<tr>
<td>Rainy</td>
<td>3/9 2/5</td>
<td>Cool 3/9 1/5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Likelihood of the two classes**

For “yes” = \( \frac{2/9 \times 3/9 \times 3/9 \times 9/14}{0.0053} = 0.0053 \)

For “no” = \( \frac{3/5 \times 1/1 \times 4/5 \times 3/5 \times 5/14}{0.0206} = 0.0206 \)

**Conversion into a probability by normalization:**

\[
P(\text{"yes"}) = \frac{0.0053}{0.0053 + 0.0206} = 0.205
\]

\[
P(\text{"no"}) = \frac{0.0206}{0.0053 + 0.0206} = 0.795
\]

A new day:

\[
\Pr[H \mid E] = \frac{\Pr[E_1 \mid H] \Pr[E_2 \mid H] \ldots \Pr[E_n \mid H] \Pr[H]}{\Pr[E]}
\]
Lesson 3.3 Using probabilities

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>High</td>
<td>True</td>
<td>?</td>
</tr>
</tbody>
</table>

Evidence $E$

Pr[$yes | E] = Pr[Outlook = Sunny | yes]  
$\times$ Pr[Temperature = Cool | yes]  
$\times$ Pr[Humidity = High | yes]  
$\times$ Pr[Windy = True | yes]  
$\times$ Pr[$yes]$  
$\times$ Pr[$E$]  
$\times$ $\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}$  
$\frac{Pr[E]}{Pr[E]}$  

Probability of class “yes”
Lesson 3.3 Using probabilities

Use Naïve Bayes

- Open file `weather.nominal.arff`
- Choose Naïve Bayes method (`bayes>NaiveBayes`)
- Look at the classifier
- Avoid zero frequencies: start all counts at 1
Lesson 3.3 Using probabilities

❖ “Naïve Bayes”: all attributes contribute equally and independently
❖ Works surprisingly well
  – even if independence assumption is clearly violated
❖ Why?
  – classification doesn’t need accurate probability estimates
    so long as the greatest probability is assigned to the correct class
❖ Adding redundant attributes causes problems
  (e.g. identical attributes) → attribute selection

Course text
❖ Section 4.2 Statistical modeling
Data Mining with Weka

Class 3 – Lesson 4

Decision trees

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.4 Decision trees

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 3.1 Simplicity first!

Lesson 3.2 Overfitting

Lesson 3.3 Using probabilities

Lesson 3.4 Decision trees

Lesson 3.5 Pruning decision trees

Lesson 3.6 Nearest neighbor
Lesson 3.4 Decision trees

Top-down: recursive divide-and-conquer

- **Select** attribute for root node
  - Create branch for each possible attribute value
- **Split** instances into subsets
  - One for each branch extending from the node
- **Repeat** recursively for each branch
  - using only instances that reach the branch
- **Stop**
  - if all instances have the same class
Lesson 3.4 Decision trees

Which attribute to select?
Lesson 3.4 Decision trees

Which is the best attribute?

- **Aim:** to get the smallest tree
- **Heuristic**
  - *choose the attribute that produces the “purest” nodes*
  - *i.e. the greatest information gain*
- **Information theory:** measure information in bits

\[
\text{entropy}(p_1, p_2, ..., p_n) = -p_1 \log p_1 - p_2 \log p_2 ... - p_n \log p_n
\]

**Information gain**

- Amount of information gained by knowing the value of the attribute
- *(Entropy of distribution before the split) – (entropy of distribution after it)*

Claude Shannon, American mathematician and scientist 1916–2001
Lesson 3.4 Decision trees

Which attribute to select?

- **outlook**
  - sunny: yes, yes, yes, yes, no, no
  - overcast: yes, yes, yes, yes, yes, no
  - rainy: yes, yes, yes, yes, no, no

- **windy**
  - false:
    - yes, yes, yes, yes, no, no
  - true:
    - yes, yes, yes, yes, yes, no

- **humidity**
  - high:
    - yes, yes, yes, yes, yes, no
  - normal:
    - yes, yes, yes, yes, no, no

- **temperature**
  - hot:
    - yes, yes, yes, yes, yes, no
  - mild:
    - yes, yes, yes, yes, yes, no
  - cool:
    - yes, yes, yes, yes, yes, no
Lesson 3.4 Decision trees

Continue to split ...

\[
\begin{align*}
\text{gain}(temperature) & = 0.571 \text{ bits} \\
\text{gain}(windy) & = 0.020 \text{ bits} \\
\text{gain}(humidity) & = 0.971 \text{ bits}
\end{align*}
\]
Lesson 3.4 Decision trees

Use J48 on the weather data

- Open file weather.nominal.arff
- Choose J48 decision tree learner (trees>J48)
- Look at the tree
- Use right-click menu to visualize the tree
Lesson 3.4 Decision trees

- J48: “top-down induction of decision trees”
- Soundly based in information theory
- Produces a tree that people can understand
- Many different criteria for attribute selection
  - rarely make a large difference
- Needs further modification to be useful in practice
  (next lesson)

Course text
- Section 4.3 Divide-and-conquer: Constructing decision trees
Data Mining with Weka

Class 3 – Lesson 5

Pruning decision trees

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.5 Pruning decision trees

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 3.1 Simplicity first!

Lesson 3.2 Overfitting

Lesson 3.3 Using probabilities

Lesson 3.4 Decision trees

Lesson 3.5 Pruning decision trees

Lesson 3.6 Nearest neighbor
Lesson 3.5 Pruning decision trees
## Lesson 3.5 Pruning decision trees

### Highly branching attributes — Extreme case: ID code

<table>
<thead>
<tr>
<th>ID code</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>b</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>c</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>d</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>e</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>f</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>g</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>h</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>i</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>j</td>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>k</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>l</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>m</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>n</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

Information gain is maximal (0.940 bits)
Lesson 3.5 Pruning decision trees

How to prune?

- Don’t continue splitting if the nodes get very small (J48 minNumObj parameter, default value 2)
- Build full tree and then work back from the leaves, applying a statistical test at each stage (confidenceFactor parameter, default value 0.25)
- Sometimes it’s good to prune an interior node, raising the subtree beneath it up one level (subtreeRaising, default true)
- Messy … complicated … not particularly illuminating
Lesson 3.5 Pruning decision trees

Over-fitting (again!)

Sometimes simplifying a decision tree gives better results

- Open file diabetes.arff
- Choose J48 decision tree learner (trees>J48)
- Prunes by default: 73.8% accuracy, tree has 20 leaves, 39 nodes
- Turn off pruning: 72.7% 22 leaves, 43 nodes
- Extreme example: breast-cancer.arff
- Default (pruned): 75.5% accuracy, tree has 4 leaves, 6 nodes
- Unpruned: 69.6% 152 leaves, 179 nodes
Lesson 3.5 Pruning decision trees

- C4.5/J48 is a popular early machine learning method
- Many different pruning methods
  - mainly change the size of the pruned tree
- Pruning is a general technique that can apply to structures other than trees (e.g. decision rules)
- Univariate vs. multivariate decision trees
  - Single vs. compound tests at the nodes
- From C4.5 to J48 (recall Lesson 1.4)

Course text
- Section 6.1 Decision trees

Ross Quinlan, Australian computer scientist
Data Mining with Weka

Class 3 – Lesson 6

Nearest neighbor

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz
Lesson 3.6 Nearest neighbor
Lesson 3.6 Nearest neighbor

“Rote learning”: simplest form of learning

❖ To classify a new instance, search training set for one that’s “most like” it
  – *the instances themselves represent the “knowledge”*
  – *lazy learning: do nothing until you have to make predictions*
❖ “Instance-based” learning = “nearest-neighbor” learning
Lesson 3.6 Nearest neighbor
Lesson 3.6 Nearest neighbor

*Search training set for one that’s “most like” it*

- Need a similarity function
  - Regular (“Euclidean”) distance? (sum of squares of differences)
  - Manhattan (“city-block”) distance? (sum of absolute differences)
  - Nominal attributes? Distance = 1 if different, 0 if same
  - Normalize the attributes to lie between 0 and 1?
Lesson 3.6 Nearest neighbor

What about noisy instances?

- Nearest-neighbor
- $k$-nearest-neighbors
  - choose majority class among several neighbors ($k$ of them)
- In Weka,
  
  lazy>IBk (instance-based learning)
Lesson 3.6 Nearest neighbor

Investigate effect of changing $k$

- Glass dataset
- lazy > IBk, $k = 1, 5, 20$
- 10-fold cross-validation

<table>
<thead>
<tr>
<th>$k$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70.6%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>67.8%</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>65.4%</td>
<td></td>
</tr>
</tbody>
</table>
Lesson 3.6 Nearest neighbor

- Often very accurate ... but slow:
  - scan entire training data to make each prediction?
  - sophisticated data structures can make this faster
- Assumes all attributes equally important
  - Remedy: attribute selection or weights
- Remedies against noisy instances:
  - Majority vote over the $k$ nearest neighbors
  - Weight instances according to prediction accuracy
  - Identify reliable “prototypes” for each class
- Statisticians have used $k$-NN since 1950s
  - If training set size $n \to \infty$ and $k \to \infty$ and $k/n \to 0$, error approaches minimum

Course text
- Section 4.7 Instance-based learning
Data Mining with Weka

Department of Computer Science
University of Waikato
New Zealand

Creative Commons Attribution 3.0 Unported License
creativecommons.org/licenses/by/3.0/

weka.waikato.ac.nz