Input:
Concepts, Attributes, Instances
Module Outline

- Terminology
- What’s a concept?
  - Classification, association, clustering, numeric prediction
- What’s in an example?
  - Relations, flat files, recursion
- What’s in an attribute?
  - Nominal, ordinal, interval, ratio
- Preparing the input
  - ARFF, attributes, missing values, getting to know data
Terminology

Components of the input:

- Concepts: kinds of things that can be learned
  - Aim: intelligible and operational concept description
- Instances: the individual, independent examples of a concept
  - Note: more complicated forms of input are possible
- Attributes: measuring aspects of an instance
  - We will focus on nominal and numeric ones
What’s a concept?

- **Data Mining Tasks (Styles of learning):**
  - Classification learning: predicting a discrete class
  - Association learning: detecting associations between features
  - Clustering: grouping similar instances into clusters
  - Numeric prediction: predicting a numeric quantity

- **Concept:** thing to be learned
- **Concept description:** output of learning scheme
Classification learning

- Example problems: attrition prediction, using DNA data for diagnosis, weather data to predict play/not play

- Classification learning is supervised
  - Scheme is being provided with actual outcome

- Outcome is called the *class* of the example

- Success can be measured on fresh data for which class labels are known (test data)

- In practice success is often measured subjectively
Association learning

- Examples: supermarket basket analysis - what items are bought together (e.g. milk+cereal, chips+salsa)
- Can be applied if no class is specified and any kind of structure is considered “interesting”
- Difference with classification learning:
  - Can predict any attribute’s value, not just the class, and more than one attribute’s value at a time
  - Hence: far more association rules than classification rules
  - Thus: constraints are necessary
    - Minimum coverage and minimum accuracy
Clustering

- Examples: customer grouping
- Finding groups of items that are similar
- Clustering is *unsupervised*
  - The class of an example is not known

**Success often measured subjectively**

<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>1.4</td>
<td>0.2</td>
<td>Iris setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris setosa</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>3.2</td>
<td>4.7</td>
<td>1.4</td>
<td>Iris versicolor</td>
</tr>
<tr>
<td>6.4</td>
<td>3.2</td>
<td>4.5</td>
<td>1.5</td>
<td>Iris versicolor</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>3.3</td>
<td>6.0</td>
<td>2.5</td>
<td>Iris virginica</td>
</tr>
<tr>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9</td>
<td>Iris virginica</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Numeric prediction

- Classification learning, but “class” is numeric
- Learning is supervised
  - Scheme is being provided with target value
- Measure success on test data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>5</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>True</td>
<td>0</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>55</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>40</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
What’s in an example?

- **Instance**: specific type of example
  - Thing to be classified, associated, or clustered
  - Individual, independent example of target concept
  - Characterized by a predetermined set of attributes

- **Input to learning scheme**: set of instances/dataset
  - Represented as a single relation/flat file

- **Rather restricted form of input**
  - No relationships between objects

- **Most common form in practical data mining**
A family tree

Peter M = Peggy F

Steven M  Graham M  Pam F

Grace F = Ray M

Ian M  Pippa F  Brian M

Anna F  Nikki F
## Family tree represented as a table

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Parent1</th>
<th>Parent2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>Male</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Peggy</td>
<td>Female</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Steven</td>
<td>Male</td>
<td>Peter</td>
<td>Peggy</td>
</tr>
<tr>
<td>Graham</td>
<td>Male</td>
<td>Peter</td>
<td>Peggy</td>
</tr>
<tr>
<td>Pam</td>
<td>Female</td>
<td>Peter</td>
<td>Peggy</td>
</tr>
<tr>
<td>Ian</td>
<td>Male</td>
<td>Grace</td>
<td>Ray</td>
</tr>
<tr>
<td>Pippa</td>
<td>Female</td>
<td>Grace</td>
<td>Ray</td>
</tr>
<tr>
<td>Brian</td>
<td>Male</td>
<td>Grace</td>
<td>Ray</td>
</tr>
<tr>
<td>Anna</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
</tr>
<tr>
<td>Nikki</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
</tr>
</tbody>
</table>
The “sister-of” relation

<table>
<thead>
<tr>
<th>First person</th>
<th>Second person</th>
<th>Sister of?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>Peggy</td>
<td>No</td>
</tr>
<tr>
<td>Peter</td>
<td>Steven</td>
<td>No</td>
</tr>
<tr>
<td>Steven</td>
<td>Peter</td>
<td>No</td>
</tr>
<tr>
<td>Steven</td>
<td>Graham</td>
<td>No</td>
</tr>
<tr>
<td>Steven</td>
<td>Pam</td>
<td>Yes</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Ian</td>
<td>Pippa</td>
<td>Yes</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Anna</td>
<td>Nikki</td>
<td>Yes</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Nikki</td>
<td>Anna</td>
<td>Yes</td>
</tr>
</tbody>
</table>

First person | Second person | Sister of? |
Steven | Pam | Yes |
Graham | Pam | Yes |
Ian | Pippa | Yes |
Brian | Pippa | Yes |
Anna | Nikki | Yes |
Nikki | Anna | Yes |
| All the rest | No |

Closed-world assumption
A full representation in one table

<table>
<thead>
<tr>
<th>First person</th>
<th></th>
<th></th>
<th></th>
<th>Second person</th>
<th></th>
<th></th>
<th>Sister of?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Gender</td>
<td>Parent1</td>
<td>Parent2</td>
<td>Name</td>
<td>Gender</td>
<td>Parent1</td>
<td>Parent2</td>
</tr>
<tr>
<td>Steven</td>
<td>Male</td>
<td>Peter</td>
<td>Peggy</td>
<td>Pam</td>
<td>Female</td>
<td>Peter</td>
<td>Peggy</td>
</tr>
<tr>
<td>Graham</td>
<td>Male</td>
<td>Peter</td>
<td>Peggy</td>
<td>Pam</td>
<td>Female</td>
<td>Peter</td>
<td>Peggy</td>
</tr>
<tr>
<td>Ian</td>
<td>Male</td>
<td>Grace</td>
<td>Ray</td>
<td>Pippa</td>
<td>Female</td>
<td>Grace</td>
<td>Ray</td>
</tr>
<tr>
<td>Brian</td>
<td>Male</td>
<td>Grace</td>
<td>Ray</td>
<td>Pippa</td>
<td>Female</td>
<td>Grace</td>
<td>Ray</td>
</tr>
<tr>
<td>Anna</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
<td>Nikki</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
</tr>
<tr>
<td>Nikki</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
<td>Anna</td>
<td>Female</td>
<td>Pam</td>
<td>Ian</td>
</tr>
</tbody>
</table>

All the rest

If second person’s gender = female  
and first person’s parent = second person’s parent 
then sister-of = yes
Generating a flat file

- Process of flattening a file is called “denormalization”
  - Several relations are joined together to make one
- Possible with any finite set of finite relations
- Problematic: relationships without pre-specified number of objects
  - Example: concept of nuclear-family
- Denormalization may produce spurious regularities that reflect structure of database
  - Example: “supplier” predicts “supplier address”
What’s in an attribute?

- Each instance is described by a fixed predefined set of features, its “attributes”
- But: number of attributes may vary in practice
  - Possible solution: “irrelevant value” flag
- Related problem: existence of an attribute may depend of value of another one
- Possible attribute types (“levels of measurement”):
  - Nominal, ordinal, interval and ratio
Nominal quantities

- Values are distinct symbols
  - Values themselves serve only as labels or names
  - *Nominal* comes from the Latin word for name
- Example: attribute “outlook” from weather data
  - Values: “sunny”, “overcast”, and “rainy”
- No relation is implied among nominal values (no ordering or distance measure)
- Only equality tests can be performed
Ordinal quantities

- Impose order on values
- But: no distance between values defined
- Example:
  attribute “temperature” in weather data
  - Values: “hot” > “mild” > “cool”
- Note: addition and subtraction don’t make sense
- Example rule:
  temperature < hot ⇒ play = yes
- Distinction between nominal and ordinal not always clear (e.g. attribute “outlook”)

witten&eibe
Interval quantities (Numeric)

- Interval quantities are not only ordered but measured in fixed and equal units.
- Example 1: attribute “temperature” expressed in degrees Fahrenheit.
- Example 2: attribute “year”.
- Difference of two values makes sense.
- Sum or product doesn’t make sense.
  - Zero point is not defined!
Ratio quantities

- Ratio quantities are ones for which the measurement scheme defines a zero point
- Example: attribute “distance”
  - Distance between an object and itself is zero
- Ratio quantities are treated as real numbers
  - All mathematical operations are allowed
- But: is there an “inherently” defined zero point?
  - Answer depends on scientific knowledge (e.g. Fahrenheit knew no lower limit to temperature)
Attribute types used in practice

- Most schemes accommodate just two levels of measurement: nominal and ordinal
- Nominal attributes are also called “categorical”, “enumerated”, or “discrete”
  - But: “enumerated” and “discrete” imply order
- Special case: dichotomy (“boolean” attribute)
- Ordinal attributes are called “numeric”, or “continuous”
  - But: “continuous” implies mathematical continuity
Attribute types: Summary

- Nominal, e.g. eye color=brown, blue, ...
  - only equality tests
  - important special case: boolean (True/False)
- Ordinal, e.g. grade=k,1,2,...,12
- Continuous (numeric), e.g. year
  - interval quantities – integer
  - ratio quantities -- real
Why specify attribute types?

- **Q: Why Machine Learning algorithms need to know about attribute type?**
- **A: To be able to make right comparisons and learn correct concepts, e.g.:**
  - **Outlook > “sunny”** does not make sense, while
  - **Temperature > “cool”** or
  - **Humidity > 70** does
- **Additional uses of attribute type:** check for valid values, deal with missing, etc.
Transforming ordinal to boolean

- Simple transformation allows ordinal attribute with $n$ values to be coded using $n-1$ boolean attributes
- Example: attribute “temperature”

<table>
<thead>
<tr>
<th>Original data</th>
<th>Transformed data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>Temperature &gt; cold</td>
</tr>
<tr>
<td>Cold</td>
<td>False</td>
</tr>
<tr>
<td>Medium</td>
<td>True</td>
</tr>
<tr>
<td>Hot</td>
<td>True</td>
</tr>
</tbody>
</table>

- Better than coding it as a nominal attribute
Metadata

- Information about the data that encodes background knowledge
- Can be used to restrict search space
- Examples:
  - Dimensional considerations (i.e. expressions must be dimensionally correct)
  - Circular orderings (e.g. degrees in compass)
  - Partial orderings (e.g. generalization/specialization relations)
Preparing the input

- Problem: different data sources (e.g. sales department, customer billing department, ...)
  - Differences: styles of record keeping, conventions, time periods, data aggregation, primary keys, errors
  - Data must be assembled, integrated, cleaned up
  - “Data warehouse”: consistent point of access
- Denormalization is not the only issue
- External data may be required (“overlay data”)
- Critical: type and level of data aggregation
The ARFF format

```arff
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {true, false}
@attribute play? {yes, no}

@data
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
...
Attribute types in Weka

- ARFF supports numeric and nominal attributes
- Interpretation depends on learning scheme
  - Numeric attributes are interpreted as
    - ordinal scales if less-than and greater-than are used
    - ratio scales if distance calculations are performed (normalization/standardization may be required)
  - Instance-based schemes define distance between nominal values (0 if values are equal, 1 otherwise)
- Integers: nominal, ordinal, or ratio scale?
Nominal vs. ordinal

- Attribute “age” nominal

  If age = young and astigmatic = no
  and tear production rate = normal
  then recommendation = soft

  If age = pre-presbyopic and astigmatic = no
  and tear production rate = normal
  then recommendation = soft

- Attribute “age” ordinal

  (e.g. “young” < “pre-presbyopic” < “presbyopic”)

  If age ≤ pre-presbyopic and astigmatic = no
  and tear production rate = normal
  then recommendation = soft
Missing values

- Frequently indicated by out-of-range entries
  - Types: unknown, unrecorded, irrelevant
  - Reasons:
    - malfunctioning equipment
    - changes in experimental design
    - collation of different datasets
    - measurement not possible

- Missing value may have significance in itself (e.g. missing test in a medical examination)
  - Most schemes assume that is not the case ➞ “missing” may need to be coded as additional value
Missing values - example

- Value may be missing because it is unrecorded or because it is inapplicable.

- In medical data, value for **Pregnant?** attribute for **Jane** is missing, while for **Joe** or **Anna** should be considered **Not applicable**.

- Some programs can infer missing values.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Sex</th>
<th>Pregnant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>25</td>
<td>F</td>
<td>N</td>
</tr>
<tr>
<td>Jane</td>
<td>27</td>
<td>F</td>
<td>-</td>
</tr>
<tr>
<td>Joe</td>
<td>30</td>
<td>M</td>
<td>-</td>
</tr>
<tr>
<td>Anna</td>
<td>2</td>
<td>F</td>
<td>-</td>
</tr>
</tbody>
</table>
Inaccurate values

- Reason: data has not been collected for mining it
- Result: errors and omissions that don’t affect original purpose of data (e.g. age of customer)
- Typographical errors in nominal attributes $\Rightarrow$ values need to be checked for consistency
- Typographical and measurement errors in numeric attributes $\Rightarrow$ outliers need to be identified
- Errors may be deliberate (e.g. wrong zip codes)
- Other problems: duplicates, stale data
Precision “Illusion”

- Example: gene expression may be reported as \( X83 = 193.3742 \), but measurement error may be +/- 20.
- Actual value is in \([173, 213]\) range, so it is appropriate to round the data to 190.
- Don’t assume that every reported digit is significant!
Getting to know the data

- Simple visualization tools are very useful
  - Nominal attributes: histograms (Distribution consistent with background knowledge?)
  - Numeric attributes: graphs (Any obvious outliers?)
- 2-D and 3-D plots show dependencies
- Need to consult domain experts
- Too much data to inspect? Take a sample!
Summary

- Concept: thing to be learned
- Instance: individual examples of a concept
- Attributes: Measuring aspects of an instance

Note: Don’t confuse learning “Class” and “Instance” with Java “Class” and “instance”
Assignment

- Use Weka to classify
  - weather data
  - zoo data
- Why accuracy is higher for models evaluated on training set only than for models evaluated with cross-validation?
Exploring data with WEKA

- Use Weka to explore
  - Weather data
  - Iris data (+ visualization)
  - Labor negotiation
- Use Emacs to examine ARFF file
- Filters:
  - Copy
  - Make_indicator
  - Nominal to binary
  - Merge-two-values