Lecture 3
Inputs: Concepts, Instances and Attributes
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)

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

Today's lecture

- Input = Concept + Instances + Attributes
- Concepts
  - What kind of learning? What kind of thing can we learn?
  - *Classification*, **association**, *clustering*, **numeric prediction**
  - We want to learn a *concept description* that is *operational* and *intelligible*
- Instances
  - Learn from a Flat file of instances
  - Each instance is an individual, independent example of a concept
  - Creating a flat file from a relationship structure
  - 2 hard problems – Recursion and Multi-Instance lines
- Attributes
  - Covered in Lecture 2
- Preparing data and getting to know the data
  - WEKA, arff format
  - Missing values

Concepts

- Concept = Thing to be learned
- We want to learn a *concept description* that is *operational* and *intelligible*
- Styles of learning
  1. *Classification:* predict a *discrete* class
  2. Association: detect associations between *features*
  3. *Clustering*: group similar instances into *clusters*
  4. Numeric prediction: predicting a *numeric* quantity

1. Classification learning

- Example problems:
  - weather data, contact lenses, irises, labour negotiations
- Classification learning is *supervised*
- Scheme is provided with the actual outcome
- Outcome is called the *class* of the example
- Measure success on fresh data for which class labels are known (test data)
- In practice success is often measured subjectively

2. Association learning

- Can be used if no class attribute is specified and any kind of structure is considered "interesting"
- Differences from classification learning:
  - Can predict *any* attribute's value, not just the class
  - Can predict *more than one* attribute's value at a time
- So : far more association rules than classification rules
  - Constraints are necessary
  - Minimum *coverage* and minimum *accuracy*
- Usually applied to categorical / nominal data not numeric

3. Clustering

- Finding groups of items that are similar
- Clustering is *unsupervised*
- The class of an example is not known
- Success often measured subjectively

	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~~
…					

4. Numeric prediction

- Variant of classification learning where the "class" is numeric (also called "regression")
- Learning is *supervised*
  - Scheme is provided with a target value
  - Measure success on test data
- We often want the prediction and the structure
  - Identify important attributes, how big is the effect of changing them
  - These are the attributes we want to control

Outlook	Temperature	Humidity	Windy	Play-time
Sunny	Hot	High	False	5
Sunny	Hot	High	True	0
Overcast	Hot	High	False	55
Rainy	Mild	Normal	False	40
…	…	…	…	…

So that was Concepts…

- Now: Instances / examples

So that was Concepts…

- Now: Instances and examples
  - Lines of data

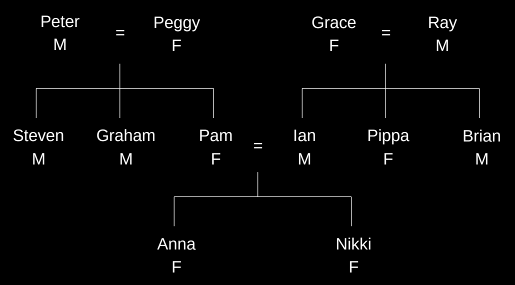
What's in an instance?

- Now: Instances / examples
  - Simple case - lines of data
  - Should be simple!
  - More complex case: examples

What's in an instance?

- Instance: a specific type of example
  - The "Thing" to be classified, associated, or clustered
  - An individual, independent example of the concept we want to learn
  - Has a fixed, predefined set of attributes
- Input to a learning scheme: set of instances (dataset)
  - Represented as a single relation, or a flat file
- Rather restricted form of input
  - No relationships between objects
- Most common form in practical data mining
  - But more complex examples are possible
  - E.g. Relationships

Relationships: A family tree



Family tree represented as a table

Name	Gender	Parent1	parent2
Peter	Male	?	?
Peggy	Female	?	?
Steven	Male	Peter	Peggy
Graham	Male	Peter	Peggy
Pam	Female	Peter	Peggy
Ian	Male	Grace	Ray
Pippa	Female	Grace	Ray
Brian	Male	Grace	Ray
Anna	Female	Pam	Ian
Nikki	Female	Pam	Ian

The "sister-of" relation

First person	Second person	Sister of?
Peter	Peggy	No
Peter	Steven	No
…	…	…
Steven	Peter	No
Steven	Graham	No
Steven	Pam	Yes
…	…	…
Ian	Pippa	Yes
…	…	…
Anna	Nikki	Yes
…	…	…
Nikki	Anna	yes

The "sister-of" relation

First person	Second person	Sister of?
Peter	Peggy	No
Peter	Steven	No
…	…	…
Steven	Peter	No
Steven	Graham	No
Steven	Pam	Yes
…	…	…
Ian	Pippa	Yes
…	…	…
Anna	Nikki	Yes
…	…	…
Nikki	Anna	yes

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

First person				Second person				Sister of?
Name	Gender	Parent1	Parent2	Name	Gender	Parent1	Parent2	
Steven	Male	Peter	Peggy	Pam	Female	Peter	Peggy	Yes
Graham	Male	Peter	Peggy	Pam	Female	Peter	Peggy	Yes
Ian	Male	Grace	Ray	Pippa	Female	Grace	Ray	Yes
Brian	Male	Grace	Ray	Pippa	Female	Grace	Ray	Yes
Anna	Female	Pam	Ian	Nikki	Female	Pam	Ian	Yes
Nikki	Female	Pam	Ian	Anna	Female	Pam	Ian	Yes
All the rest								No

```
If second person's gender = female
  and first person's parent = second person's parent
  then sister-of = yes
```


Generating the flat file

- This process is called denormalization
  - Several relations are joined together to make one
  - Inverse of database normalization
  - Possible with any (finite) set of (finite) relations
- Problems with generating the flat file
  - What if there isn't a fixed number of attributes?
    - E.g. nuclear family – father, mother, how many siblings?
  - Denormalizing can introduce fake regularities
    - E.g. **supplier** predicts **supplier address** is not interesting

Generating the flat file

- Hard problems with generating the flat file
  1. What if the relation is recursive?
     - E.g. Ancestor-of
  2. Multi-instance

Hard problem 1: The "ancestor-of" relation

First person				Second person				Ancestor of?
Name	Gender	Parent1	Parent2	Name	Gender	Parent1	Parent2	
Peter	Male	?	?	Steven	Male	Peter	Peggy	Yes
Peter	Male	?	?	Pam	Female	Peter	Peggy	Yes
Peter	Male	?	?	Anna	Female	Pam	Ian	Yes
Peter	Male	?	?	Nikki	Female	Pam	Ian	Yes
Pam	Female	Peter	Peggy	Nikki	Female	Pam	Ian	Yes
Grace	Female	?	?	Ian	Male	Grace	Ray	Yes
Grace	Female	?	?	Nikki	Female	Pam	Ian	Yes
Other positive examples here								Yes
All the rest								No

Hard problem 1: Recursion

- Infinite relations require recursion

```
If person1 is a parent of person2
   then person1 is an ancestor of person2

If person1 is a parent of person2
   and person2 is an ancestor of person3
   then person1 is an ancestor of person3
```

- Appropriate techniques are known as "inductive logic programming"
  - (e.g. Quinlan's FOIL)
  - Problems: (a) noise and (b) computational complexity

Hard problem 2: Multi-instance Examples

- Each individual example comprises a *set* of instances
  - All instances are described by the same attributes
  - One or more instances within an example may be responsible for its classification
- Goal of learning is still to produce a concept description
- Has important real world applications
  - e.g. drug activity prediction

So that was

- Concepts
- Instances
  - Independent lines of data
  - Learning about relations between data
  - Multi-instance examples
- Now on to
  - Attributes

What's in an attribute?

- Each instance is described by a fixed predefined set of attributes
- But:
  - Sometimes the number of attributes may vary
  - E.g. "wheels" for a vehicle dataset – cars yes, ships no
  - Possible solution: "irrelevant value" flag
- Related problem:
  - The existence of an attribute may depend of value of another one

That was…

- Concepts
- Instances
- Attributes
- And now…
  - Basic data format for WEKA
    - Header
    - Attribute types
    - Relational attributes
    - Sparse data

The ARFF format

```
%
% ARFF file for weather data with some numeric features
%
@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

- Interpretation of attribute types in ARFF depends on the learning scheme
- Integers in a data file:
  - Could be nominal, ordinal, or ratio scale?
- Numeric attributes are interpreted as
  - *ordinal* scales if less-than and greater-than are used
  - *ratio* scales if distance calculations are performed, or using numeric prediction methods like regression
- Instance-based learning schemes (like Nearest Neighbour) can define a distance between nominal values
  - 0 if values are equal, 1 otherwise

Attribute types: Nominal vs ordinal

- If attribute *Age* is nominal – need two rules

```
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
```

- Or if *Age* is ordinal – one rule

```
young < pre-presbyopic < presbyopic
```

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


Additional attribute types

- ARFF supports *string* attributes:

```
@attribute description string
```

- Similar to nominal attributes but list of values is not pre-specified
- It also supports *date* attributes:

```
@attribute today date
```

- Uses the ISO-8601 combined date and time format *yyyy-MM-dd-THH:mm:ss*

Relational attributes

- Allow multi-instance problems to be represented in ARFF format
  - The value of a relational attribute is a *separate* set of instances

```
@attribute bag relational
@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@end bag
```

  - Nested attribute block gives the structure of the referenced instances

Multi-instance ARFF

```
%
% Multiple instance ARFF file for the weather data
%
@relation weather

@attribute bag_ID { 1, 2, 3, 4, 5, 6, 7 }
@attribute bag relational
@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
@attribute humidity numeric
@attribute windy {true, false}
@end bag
@attribute play? {yes, no}

@data
1, "sunny, 85, 85, false\nsunny, 80, 90, true", no
2, "overcast, 83, 86, false\nrainy, 70, 96, false", yes
...
```


Sparse data in ARFF

- In some applications most attribute values in a dataset are zero
  - E.g.: word counts in a text categorization problem
- ARFF supports sparse data

```
0, 26, 0,  0, 0 ,0, 63, 0, 0, 0, "class A"
0,  0, 0, 42, 0, 0,  0, 0, 0, 0, "class B"
```

```
{1 26, 6 63, 10 "class A"}
{3 42, 10 "class B"}
```

- This also works for nominal attributes (where the first value corresponds to "zero")

So that was …

- Concepts
- Instances
- Attributes
- Data input to Weka
- Friday
  - Preparing data for input to DM/ML in general