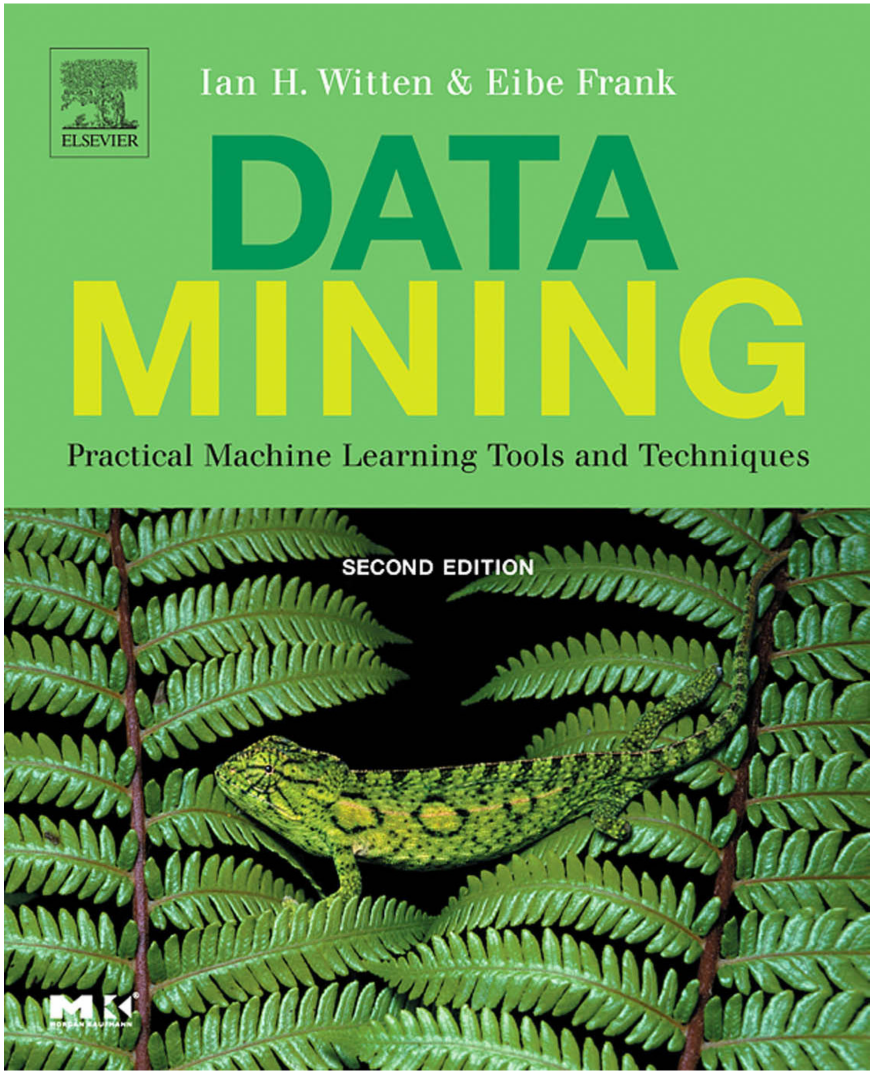


Chapter 2: Input: Concepts, Instances, and Attributes

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
  - **Attributes:** measuring aspects of an instance
    - We will focus on nominal (categorical) and numeric ones

2.1 What's a concept?

What's a concept?

- 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: weather data, contact lenses, irises, labor negotiations
  - Scheme is provided with 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

Association learning

- Can be applied if no class is specified and any kind of structure is considered "interesting"
- Difference to 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 (80% of data set), and
    - Minimum accuracy (95% accurate)

Clustering

- Finding groups of items that are similar
  - The class of an example is not known
- Success often measured subjectively
- Example: a version of the iris data in which the type of iris is omitted

Iris data as a clustering problem

	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
...				
51	7.0	3.2	4.7	1.4
52	6.4	3.2	4.5	1.5
53	6.9	3.1	4.9	1.5
54	5.5	2.3	4.0	1.3
55	6.5	2.8	4.6	1.5
...				
101	6.3	3.3	6.0	2.5
102	5.8	2.7	5.1	1.9
103	7.1	3.0	5.9	2.1
104	6.3	2.9	5.6	1.8
105	6.5	3.0	5.8	2.2
...				

Numeric prediction

- Variant of classification learning where “class” is numeric (also called “regression”)
- Scheme is being provided with target value
- Measure success on test data
- To find the important attributes and how they relate to the numeric outcome
- Examples:
  - The CPU performance problem
  - a version of the weather data in which what is to be predicted is the time (in minutes) to play

Weather data with a numeric class

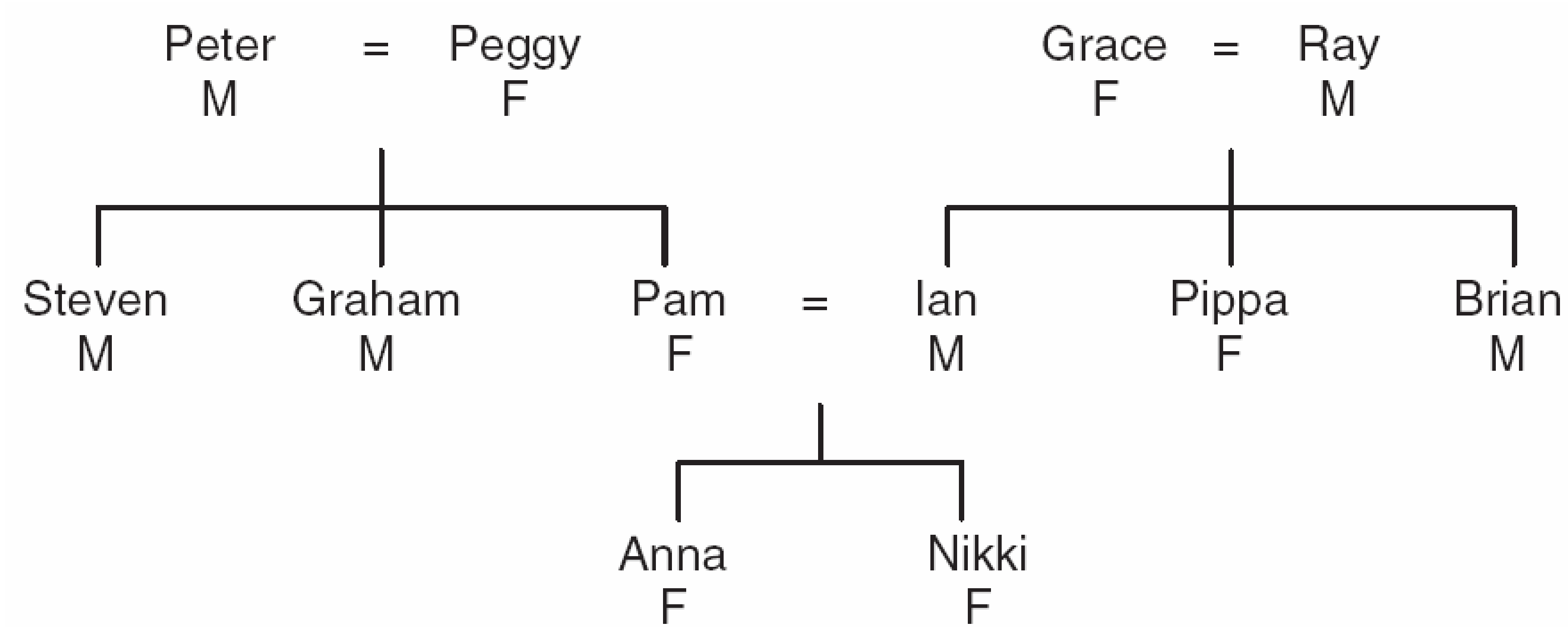
Outlook	Temperature	Humidity	Windy	Play time (min.)
sunny	85	85	false	5
sunny	80	90	true	0
overcast	83	86	false	55
rainy	70	96	false	40
rainy	68	80	false	65
rainy	65	70	true	45
overcast	64	65	true	60
sunny	72	95	false	0
sunny	69	70	false	70
rainy	75	80	false	45
sunny	75	70	true	50
overcast	72	90	true	55
overcast	81	75	false	75
rainy	71	91	true	10

2.2 What's in an example?

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
- Each dataset is represented as a matrix of instances versus attributes
  - Represented as a single relation/flat file
- Rather restricted form of input
  - No relationships between objects

A family tree



Two ways of expressing 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
Steven	Grace	no
...	......	
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

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
. . .			

The sister-of relation represented in a table

First person				Second person				
Name	Gender	Parent1	Parent2	Name	Gender	Parent1	Parent2	Sister of?
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

A simple rule for the sister-of relation

```
If second person's gender = female
   and first person's parent1 = second person's parent1
   then sister-of = yes
```

Generating a flat file

- Process of flattening called “denormalization”
  - Several relations are joined together to make one
- Possible with any finite set of finite relations
- Problematic: relationships without prespecified number of objects
- Denormalization may produce spurious regularities that reflect structure of database
  - Example: “supplier” predicts “supplier address”

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 examples here				yes
				all the rest				no

2.3 What's in an attribute?

What's in an attribute?

- Each instance is described by a fixed predefined set of features or **attributes**
- But: number of attributes may vary in practice
  - Possible solution: "irrelevant value" flag
  - If the instances were transportation vehicles
- Related problem: existence of an attribute may depend of value of another one
  - Spouse's name depends on the value of married or single attribute
- 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

```
outlook: sunny    → no
         overcast → yes
         rainy    → yes
```

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”)

Interval quantities

- Interval quantities are not only ordered but measured in fixed and equal units
- Example 1: attribute “temperature” expressed in degrees Fahrenheit
- Example 2: attribute “date” (year)
- Difference of two values makes sense
- Sum or product doesn’t make sense
  - E.g. sum of the years 1939 and 1945 (3884)
  - Or, three times the year 1939 (5817)
- 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 data mining 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)

- Information about the data is called ***metadata***

2.4 Preparing the input

Preparing the input

- Denormalization is not the only issue
- ***Data cleaning:*** a process of checking data in quality and careful
- 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”: The idea of company wide database integration
- External data may be required
- Critical: type and level of data aggregation

The ARFF format

- The attribute-relation file format (ARFF)
- a standard way of representing datasets that
  - consist of independent, unordered instances
  - do not involve relationships among instances
- ARFF is used in the Java package Called the Waikato Environment for Knowledge Analysis, or Weka

ARFF file for the weather data

```
% ARFF file for the 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
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
    ....
```

Additional attribute types

- ARFF supports *string* attributes:

```
@attribute description string
```

  – Similar to nominal attributes but list of values is not prespecified

- It also supports *date* attributes:

```
@attribute today date
```

  – Uses the ISO8601

  – combined date and time format *yyyyMMddTHH:mm:ss*

Sparse data

- In some applications most attribute values in a dataset are zero
  - E.g.1: supermarket basket data
  - E.g.2: 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

Attribute types

- Interpretation of attribute types in ARFF 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
  - Instance-based schemes define distance between nominal values (0 if values are equal, 1 otherwise)
- Integers in some given data file
  - Part number, student number

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
- 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

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 Þ values need to be checked for consistency
- Typographical and measurement errors in numeric attributes => outliers need to be identified
- Errors may be deliberate (e.g. wrong zip codes)
- Other problems: duplicates, stale data

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?)
- 2D and 3D plots show dependencies
- Need to consult domain experts
- Too much data to inspect? Take a sample!

The end of

Chapter 2: Input: Concepts, Instances, and Attributes