Introduction to Machine Learning

Prof. Fabrizio Riguzzi

ML Definitions

- Definition 1:
  - Learning is constructing or modifying representations of what is being experienced [Michalski 1986], pag. 10
- Definition 2:
  - Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time [Simon 1984], pag. 28

ML Applications

- A) Knowledge extraction
  - To be employed in knowledge-based systems (e.g. in classification systems)
  - To be presented to humans (e.g. for scientific purposes, i.e., discovery of new scientific theories)
- B) Improvement of the performances of a machine
  - E.g. improvement of the motion and sensing capabilities of a robot

Machine Learning Application Examples

- Spam filter
- Search suggestions (Google)
- Information retrieval
- Recommender systems (Amazon)
- Computer vision
- Natural language processing
- Bioinformatics
- Chemioinformatics
- …....
-

ML Techniques

- Symbolic techniques (applications A and B)
  - Representation languages
    - Propositional
    - First order
- Statistical techniques (applications A and B)
  - Representation languages
    - Propositional
    - First order
- Neural networks (application B)

Inductive Learning

- The system starts from observations provided by a teacher or from the environment
- It generalizes them, i.e. it obtains knowledge hopefully valid also in cases not yet observed (induction).
- Two types of inductive learning:
  - Learning from examples: the observations are grouped into a set of positive examples, instances of the concept to be learned, and into a set of negative examples, non-instances of the concept
  - Descriptive learning: the aim is to find regularities in the data

Inductive Learning from Examples

- Universe U: set U of all the **objects** (also called **instances**) of the domain
- **Concept** C: subset of U, $C \subseteq U$
- Also called **class**
- Object description language L_o
- Concept description language L_c
- A procedure for verifying whether a description D_C of a concept C is satisfied by a description D_x of an object x, meaning that $x \in C$

Inductive Learning from Examples

- Informally:
  - Learning a concept C means finding a description D_C of C such that, for all objects $x \in U$, $x \in C$ iff D_x satisfies D_C.

Inductive Learning from Examples

- Fact: description of an object
- Example for a concept C: labeled fact,
  - + label if the object belongs to the concept (**positive example**)
  - - label if the object does not belong to the concept (**negative example**)
- Training set E: set of labeled facts, subsets:
  - E^+: set of positive examples
  - E^-: set of negative examples
- Two classes: + and -, in general we may have more than two classes

Inductive Learning from Examples

- Hypothesis: description of the concept to be learned
- If a fact satisfies a hypothesis we say that the hypothesis **covers** the fact
- Coverage test function

  covers(H,e)

  – returns true if e is covered by H and false otherwise
- Extension to sets of examples:

  covers(H,E)={e $\in$ E| covers(H,e)= true}

Inductive Learning from Examples

- We want to find an hypothesis H such that

$$\forall e \in U: \text{covers}(H,e) \Leftrightarrow e \in C$$

- In practice, we know only instances from E
- So we require that

$$\forall e \in E: \text{covers}(H,e) \Leftrightarrow e \in C$$

- Thus
  - $\text{covers}(H,e) \Rightarrow e \in C$
  - $\neg\, \text{covers}(H,e) \Rightarrow \neg\, (e \in C)$

Inductive Learning from Examples

- Given a set E of positive and negative examples of a concept C, expressed in object description language L_o
- Find an hypothesis H, expressed in a concept description language L_c, such that
  - Every positive example $e^+ \in E^+$ is covered by H
  - No negative example $e^- \in E^-$ is covered by H

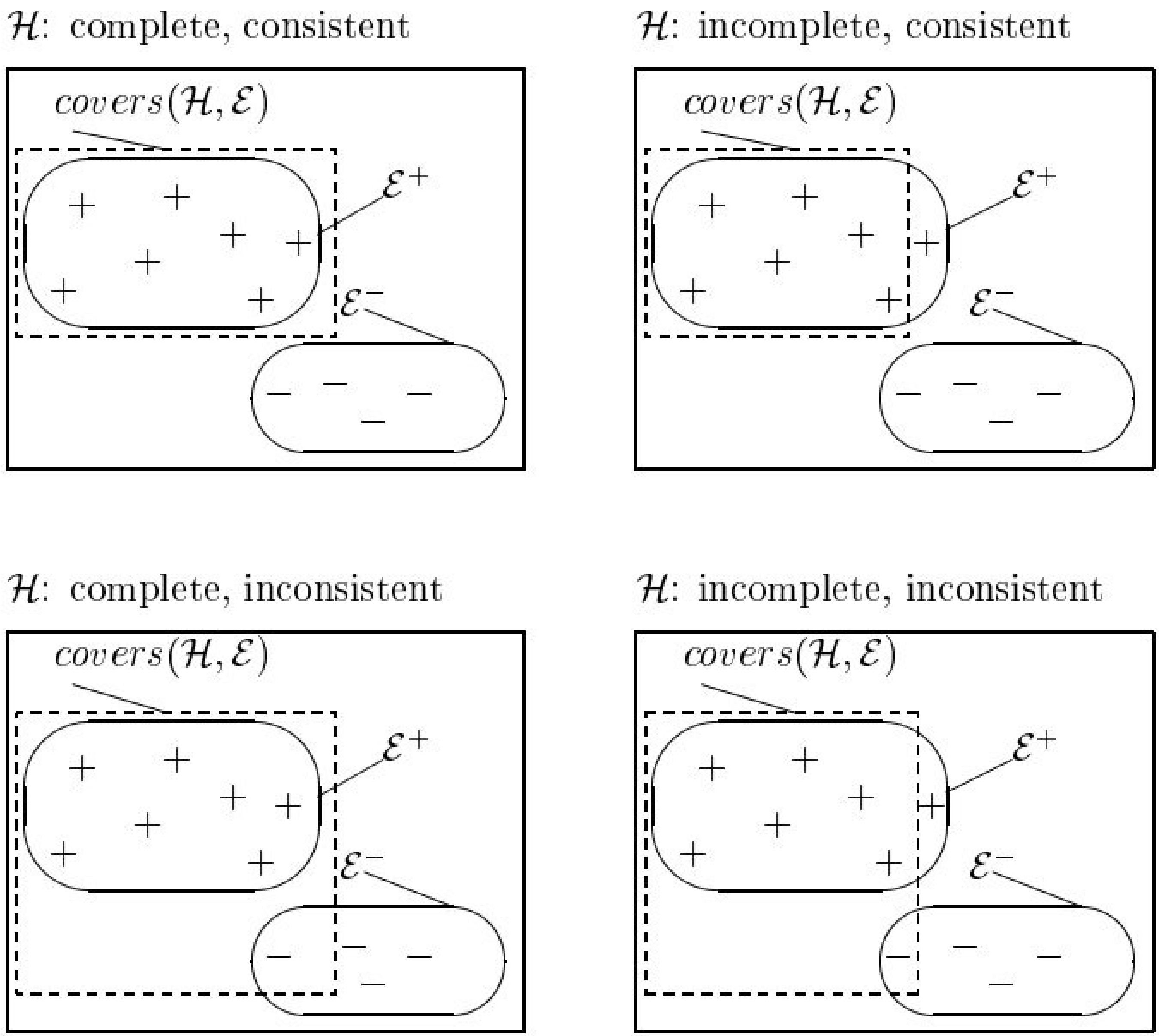
Completeness and Consistency

- An hypothesis H is called complete if it covers all the positive examples, i.e.

$$covers(H,E^+)=E^+$$

- An hypothesis H is called consistent if it does not cover any negative examples, i.e.

$$covers(H,E^-)= \emptyset$$



$\mathcal{H}$: complete, consistent
$covers(\mathcal{H}, \mathcal{E})$
$\mathcal{E}^+$
$\mathcal{E}^-$
$\mathcal{H}$: incomplete, consistent
$covers(\mathcal{H}, \mathcal{E})$
$\mathcal{E}^+$
$\mathcal{E}^-$
$\mathcal{H}$: complete, inconsistent
$covers(\mathcal{H}, \mathcal{E})$
$\mathcal{E}^+$
$\mathcal{E}^-$
$\mathcal{H}$: incomplete, inconsistent
$covers(\mathcal{H}, \mathcal{E})$
$\mathcal{E}^+$
$\mathcal{E}^-$

Representation Languages

- Propositional or attribute-value languages
- First order/logical/relational languages

Example

- Universe: set of athletes
- Instances described by
  - height=continuous (real)
  - weight=continuous (integer),
  - preferred_music=discrete, values={rock,pop,rap}
- Example of instance

  height=1.85m, weight=90kg, preferred_music=rock
- Class=sport played, values={football, baseball, basketball, boxing}

Attribute-value languages

- Every instance is described by the values taken by a set of attributes (fixed for all the instances)
- The training set can be represented as a table
- Attributes can be
  - Boolean or binary (2 values)
  - Discrete or nominal (more than 2 values)
  - Ordinal (ordered discrete values)
  - Continuous or numeric (interval or ratio scale, integer or real)
- If there are k attributes, each instances can be described by a point in a k-dimensional space

Attribute-Value Concept Description Languages

- Single production rule, conjunction in the body, class in the head
- Example: football player
  - If weight<100 and preferred_music=rock → football
- They can also be represented as a tuple with a constraint for every attribute (class is left implicit):
  - attribute=?, all the values satisfy the constraint
  - attribute=value
  - attribute<value
  - attribute=∅, no value is acceptable (the hypothesis covers 0 examples)
- Example: football player
  - <height=?,weight<100, preferred_music=rock>

Attribute-Value Concept Description Languages

- Set of production rules, any Boolean formula in the body, class in the head
- Often only conjunctions in the body
- Often decision lists:
  - The rules are tried in order
  - The first that fires gives the class
- Example:

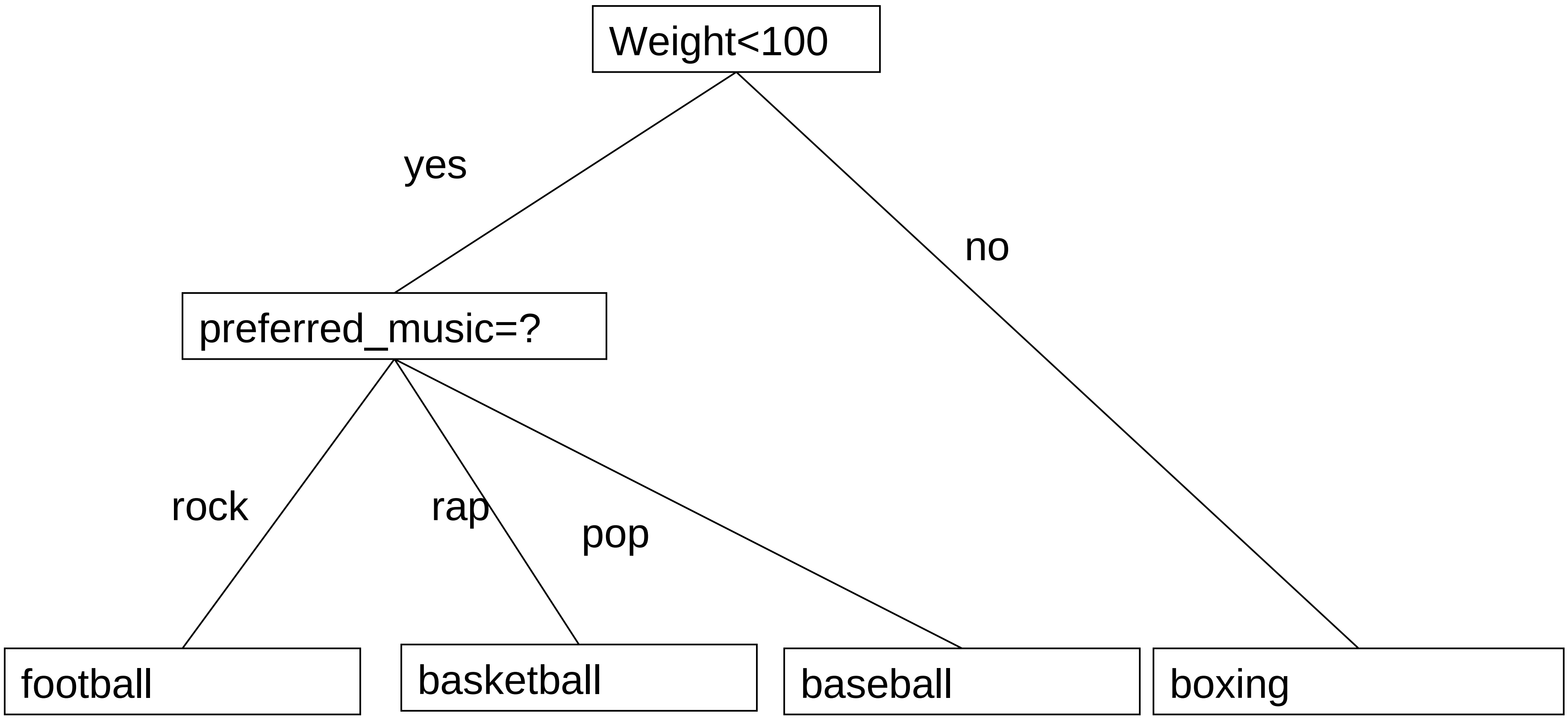
  weight<100 and preferred_music=pop $\rightarrow$ baseball

  weight<100 and preferred_music=rap $\rightarrow$ basketball

Attribute-Value Concept Description Languages

- Decision trees:
  - Every node corresponds to a test on an attribute
  - Every child corresponds to a result of the test
  - Every leaf is associated to a class

Example



Weight<100
yes
no
preferred_music=?
rock
rap
pop
football
basketball
baseball
boxing

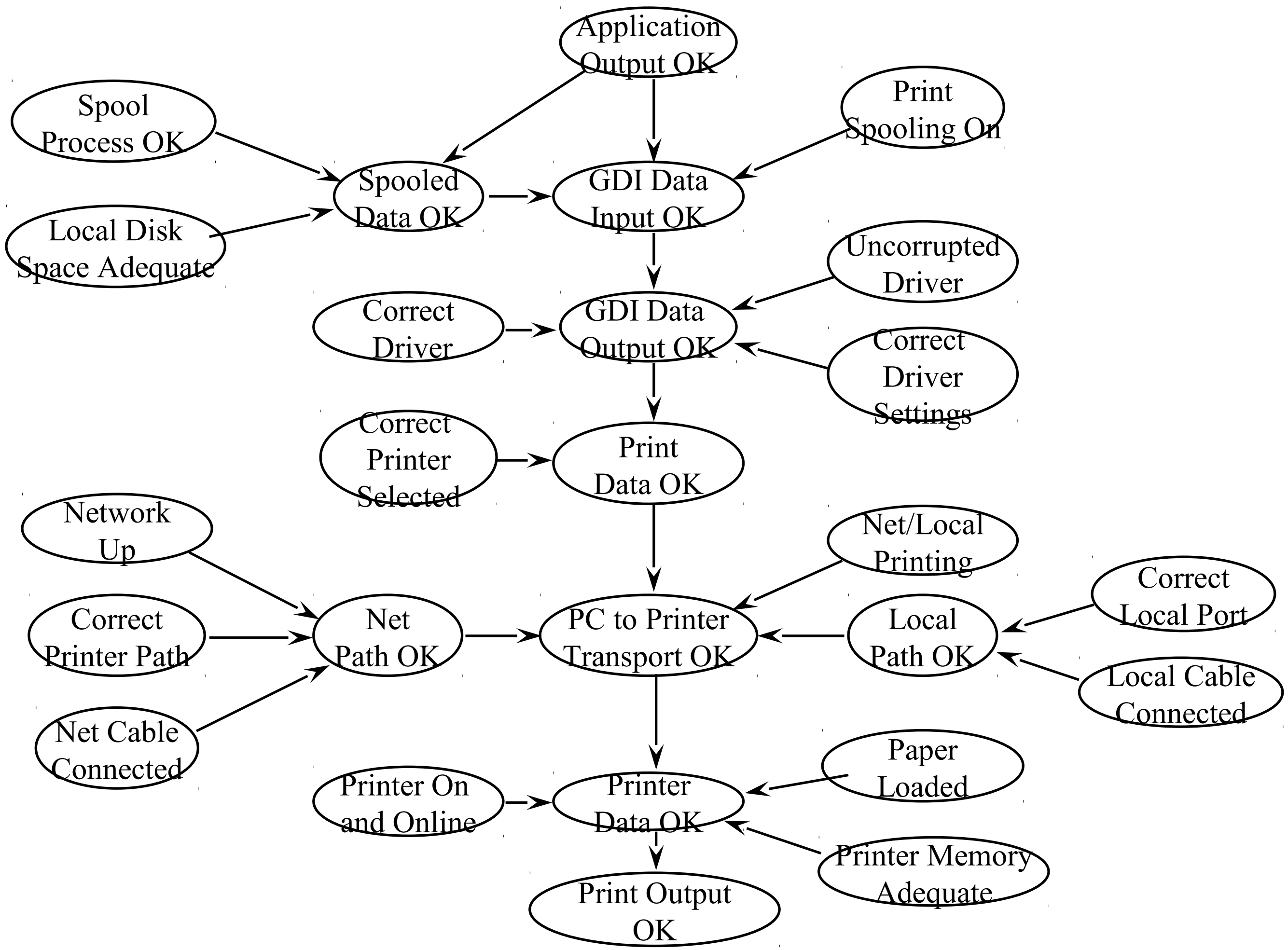
Attribute-Value Languages

- Equivalent to propositional logic:
  - Each equality or inequality can be seen as a proposition (zero arity predicate) that is either true or false for an instance
  - There are no variables, quantifiers and predicates with arity > 0

Attribute-Value Concept Description Languages

- Bayesian networks
  - Represent probabilistic dependencies among attributes
  - Qualitative component: the parents of a node are the attributes that directly influence the node
  - Quantitative component: strength of the dependency

Example: Printer Troubleshooting (Windows 95)



Problems of Propositional Languages

- Instances with parts, subparts, attributes of subparts and relation among subparts
- Example: jones family, components

name: dave, son: mike, father: ron, age: 70, hair: white

name: mike, son: junior, father: dave, age: 35

name: junior, father: mike, age: 3

- In general, families may have a variable number of components and each component may have a variable number of attributes

First-Order Languages

- Instances described by logic theories
- They allow to easily represents parts of objects, attributes of parts and relations among parts
- Example:
  – Parts: “object”(object_id, part_id)
  – Attributes of parts: “attribute”(part_id,value)
  – Relation among parts: below(part_id1,part_id2)

Example

- Instance: jones family, first-order representation:

family(jones,dave).

family(jones,mike).

family(jones,junior).

age(dave,70).

hair(dave,white).

age(mike,35).

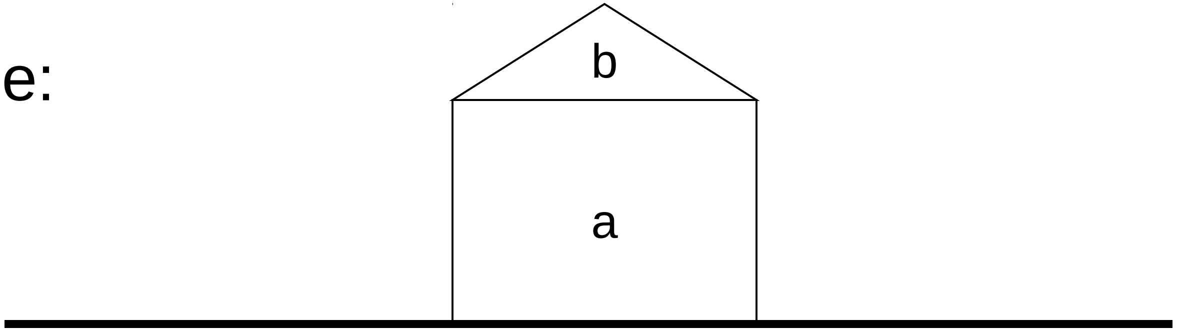
age(junior,3).

father(dave,mike).

father(mike,junior).

Example: Blocks World

- Instance:



- Instance e: first-order representation

object(e,a). object(e,b),

square(a). triangle(b).

large(a). small(b).

on-table(a). on(b,a).

First-Order Languages Concept Description Languages

- They allow to use variables and quantifiers
  - Example: family with a grandfather

  gf(X):-family(X,Y),father(Y,Z),father(Z,W).

  gf(X):-family(X,Y),father(Y,Z),mother(Z,W).
- They allow to represent recursive concepts
  - Example: ancestor

  ancestor(X,Y):-father(X,Y).

  ancestor(X,Y):-father(X,Z),ancestor(Z,Y).

First-Order Languages

- Advantages:
  - Uniform representation of instances and hypothesis
  - The semantics of these languages is well defined and amply studied
  - Availability of well-engineered interpreters

Example: Targeted Mailing

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	yes
ann	29	f	wa	no
steve	31	m	va	no

If Age<30 and Address=ca then Resp=yes

Example: Targeted Mailing

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	no
ann	29	f	wa	yes
steve	31	m	va	no

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250

transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

The customer will respond if she/he has bought an item of category clothing

Propositionalization

customer ▷◁transaction ▷◁ article

Name	Age	Sex	Address	Article	Quantity	Category	Size	Price	Resp
john	35	m	ca	bike_1	2	sport	l	1000	no
john	35	m	ca	tent_2	1	outdoor	m	250	no
mary	25	f	ca						no
ann	29	f	wa	jacket_2	1	clothing	l	150	yes
ann	29	f	wa	bike_1	3	sport	l	1000	yes
steve	31	m	va	bike_1	2	sport	l	1000	no

Propositionalization

Replicate attributes

Name	Age	Sex	Address	Article1	Quantity 1	Category 1	Size1	Price1
john	35	m	ca	bike_1	2	sport	l	1000
mary	25	f	ca					
ann	29	f	wa	jacket_2	1	clothing	l	150
steve	31	m	va	bike_1	2	sport	l	1000

Article2	Quantity2	Cateogory 2	Size2	Price2	Resp
tent_2	1	outdoor	m	250	no
					no
bike_1	3	sport	l	1000	yes
					no

Logic

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	no
ann	29	f	wa	yes
steve	31	m	va	no

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250

transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

```
respond(Customer):-
    transaction(Customer,Article,_Quantity),
    article(Article,clothing,_Size,_Price).
```

Machine Learning Techniques

- Attribute-value languages:
  - Version spaces
  - Decision trees
  - Production rules
  - Instance based methods
  - Bayesian networks
- First-order languages:
  - Inductive Logic Programming
  - Statistical Relational Learning

Machine Learning

- Usually Machine Learning algorithms perform a search in the space of the concept description language
- The aim is to find the hypothesis that best matches the training set
- Often the search space is a subset of all the hypotheses in the language (language bias)

Evaluation Measures

- Confusion matrix: predictions of a hypothesis on a set of examples

pos	neg	<-Predicted class
TP	FN	pos
FP	TN	neg

- TP=true positive, FN=false negative, FP=false positive, TN=true negative, P=TP+FN=positive, N=FP+TN=negative
- **Accuracy**=(TP+TN)/(TP+TN+FN+FP)
- **Error rate**=(FP+FN)/(TP+TN+FN+FP)=1-Accuracy
- **TP Rate**=TP/(TP+FN)=TP/P
- **FP Rate**=FP/(FP+TN)=FP/N
- **Precision**=TP/(TP+FP)
- **Recall**=TP/(TP+FN)=TP Rate
- **F-meaure**=2*Precision*Recall/(Precision+Recall)

References

[Ber96] F. Bergadano e D. Gunetti, *Inductive Logic Programming - From Machine Learning to Software Engineering*, MIT Press, Cambridge, Massachusetts, 1996

[Mit97] T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

[Michalski 1986] Michalski, R. S. “Understanding the nature of learning: Issues and research directions” in Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning - An Artificial Intelligence Approach, Volume II, Morgan Kaufmann Publishers, Los Altos, California, pages 3—26, 1986.

References

[Simon 1984] Simon, H. A. “Why should machines learn” In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, Machine Learning - An Artificial Intelligence Approach, Springer-Verlag, Berlin, pages 25—37, 1984.