Machine Learning

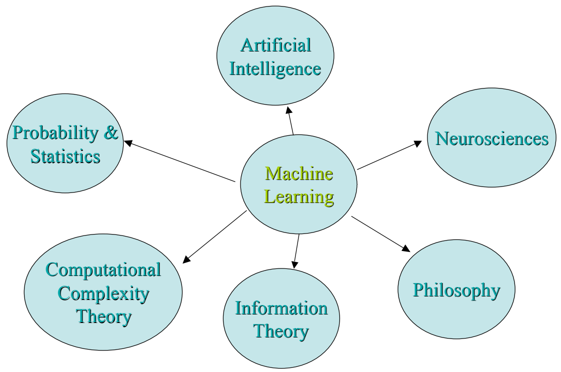
Outline

- Introduction and Basic Notions
- Fundamental tasks
  - Classification, Regression, Clustering, Reinforcement
- Representation
  - Symbolic vs. Subsymbolic
  - Propositional vs. Relational
- Ensemble Learning (Bagging, Boosting)
- Statistical Relational Learning
- Kernel Machines (NN, SVM)
- Clustering (Partitioning, hyerarchical)
- PAC Learning Model
- Evaluation (Overfitting, Error estimation, ROC analysis)
- Historical development
- Open problems and Current trends

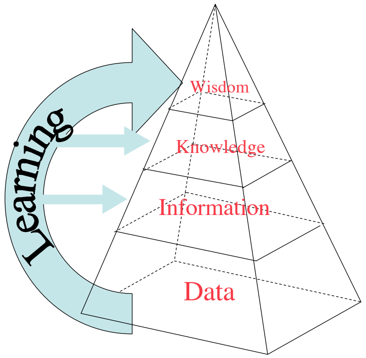
What is Learning ?

- LEARNING IN LIVING ORGANISMS DENOTES A SET OF POSSIBLY COMPLEX, DIFFERENT ACTIVITIES, INVOLVING A VARIETY OF COGNITIVE SKILLS
- LEARNING IS FUNDAMENTAL FOR SURVIVAL
- LEARNING IS THE BASIS OF INTELLIGENCE

Machine Learning: A Multidisciplinary Field



The DIKW Pyramid



" ...
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information? "
✓✓✓
(T. S. Eliot, *The Rock, 1934*)

Learning Facts, Names, Numbers

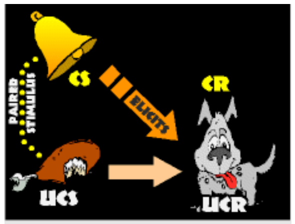
apple

telephone number

Blaise Pascal is a philosopher

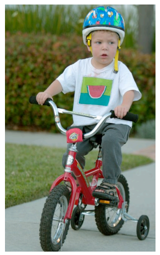
Earth is a sphere

Stimulus/Response



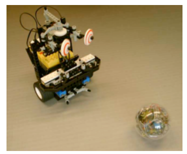
Motor Skills

Walking



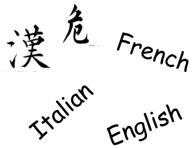
Riding a bicycle

Writing on a keybord



Talking, New Language

Learning to speak



Learning a new language

Recognition : Faces

Classification: Text Documents

TOTAL

all about the company

→ Company home page

vs

Personal home page

vs

University home page

vs

…

Grouping

Higher Cognitive Functions

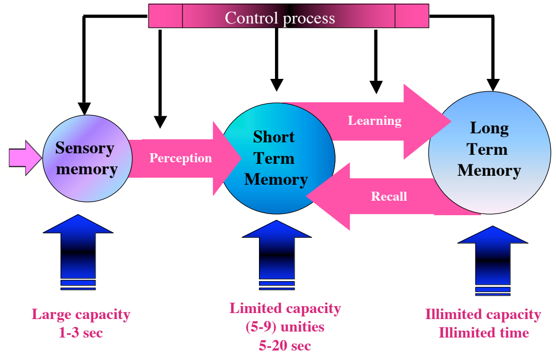
- Category formation
- Learning from reasoning
- Learning from experience
- Learning by analogy
- Active learning

« Natural » Learning

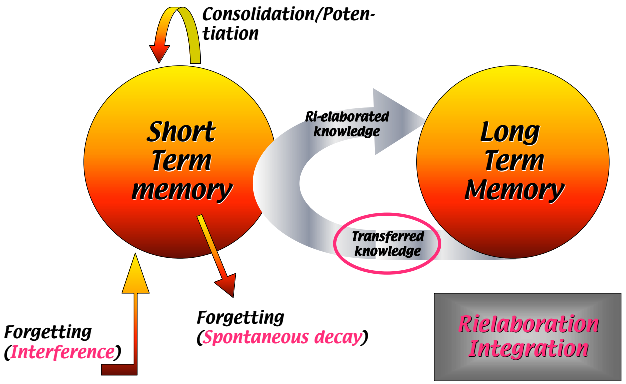
"Information Processing" Model

- Models of Memory and Learning that exploits the **Computer** as a metaphore
- **Learning** = information processing and memorization

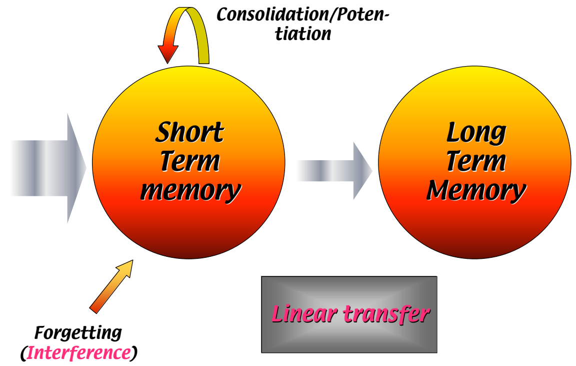
Model of Human Learning



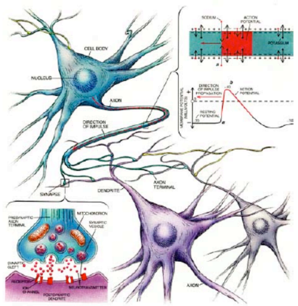
STM -> LTM (Human)



STM -> LTM (Machine)



Human Learning : Neural Networks



- Neural Networks are made up of many identical processors
- Processing is powerful because connections are rich in information

Origins

Machine Learning Precursor

Arthur Samuel developed a program that learned to play checkers well enough to beat skilled humans in the 1950s. This is the first notable success of machine learning.

Machine Learning : Origins

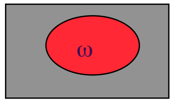
- Conceptual Learning
  - Principle of « Human Comprehensibility »
  - Michalski, Carbonell, Mitchell
    - Pittsburgh Workshop (1980, 1983, 1985, 1987)
    - Ann Harbor Workshop (1988)
- Neural networks
  - Rosenblatt (Perceptron, ~1940)
  - MacCullocch & Pitts (1943)
  - Minsky & Papert (Linear Discriminator, 1950)
  - NO in *Computers and Thought* (Feigembaum & Feldman, 1963)
  - McClelland & Rumelhart (Multi-layer perceptron, 1986)

Graph of historycal development --> at the end

Basic Notions

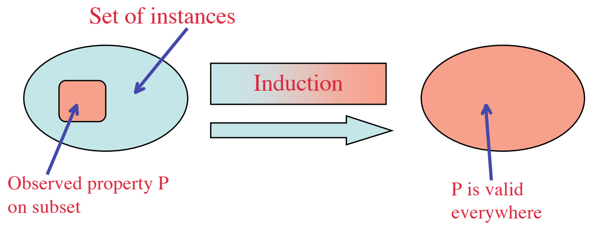
Concept

- Concept
  - Extensional notion
  - Intensional notion
    - Classic (matching necessary and sufficient conditions)
      - Aristotle
    - Heuristic (matching sufficient conditions)
      - Wittgestein
    - Prototypical (distance)
      - Rosch
    - Procedural notion (simulation)
      - Barsalou



Induction (Why can we learn?)

- Continuity hypothesis
  - The future is similar to the past



Induction is ***falsity preserving*** (Hume)

Generalization (How do we learn?)

Later on --->

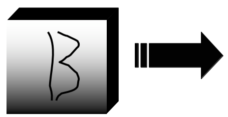
Choice (Occam's Razor)

- Inductive hypotheses explaining observations may be infinite in number
- Selection criteria

« ***Entia non sunt multiplicanda praeter necessitate*** » (Wilhelm von Occam)

Among alternative hypotheses, choose the simplest one, ***ceteris paribus***

Context

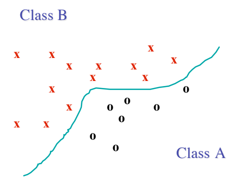


Letter ? ***B***

Number ? ***13***

Basic Tasks

Classification



Classification

Ex → Algorithm → (Sufficient) Description for each class

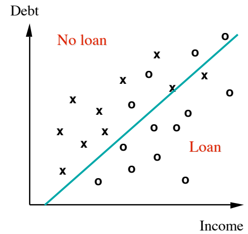
We want to label *portions of the instance space* with the name of one (or more) of the given classes

Different criteria:

Memory	**k-NN, CBR**
Generalization	**Decision trees, Classification rules, NN, SVM, …**
Probability	**Discriminant analysis, Bayes rule, …**
“Causality”	**Bayesiane nets, …**

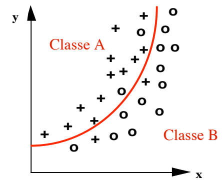
Discriminant Analysis

Funzione discriminante



Linear

Loan : **$y - a\,x - b < 0$**



Non Linear

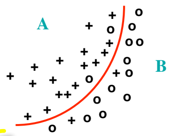
Class A : **$y - a\,x^2 - b > 0$**

Bayesian Classifier

Probabilistic discrimination function

Bayesian approach

Maximum Likelihood



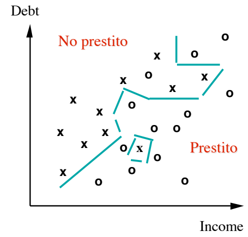
$$\Pr\{A \mid \vec{x}\} = \frac{\Pr\{\vec{x} \mid A\}\ P(A)}{\Pr\{\vec{x} \mid A\}\ P(A) + \Pr\{\vec{x} \mid B\}\ P(B)}$$

$$\Pr\{B \mid \vec{x}\} = \frac{\Pr\{\vec{x} \mid B\}\ P(B)}{\Pr\{\vec{x} \mid A\}\ P(A) + \Pr\{\vec{x} \mid B\}\ P(B)}$$

$$\begin{cases} \Pr\{A \mid \vec{x}\} = \Pr\{B \mid \vec{x}\} \\ \Pr\{A \mid \vec{x}\}\,P(A) = \Pr\{B \mid \vec{x}\}\,P(B) \end{cases}$$

Case-Based Approach

k-Nearest Neighbours

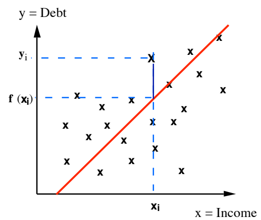


Regression

To find a functional relation between variables occurring ina database

$$y = f(x_1, \dots, x_{k-1})$$

Usually f has a (parametrized) known form and learning means to estimate the parameters $\theta = (\theta_1, \dots, \theta_K)$

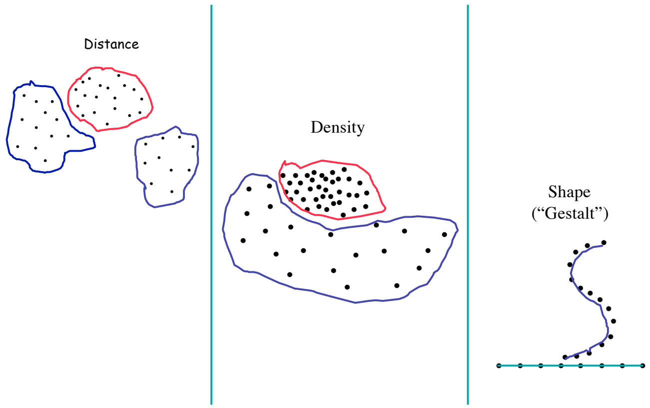


Minimize $H(\theta)$ w.r.t. θ

Linear regression:

$y = a\,x + b$

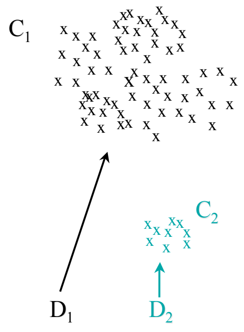
Clustering

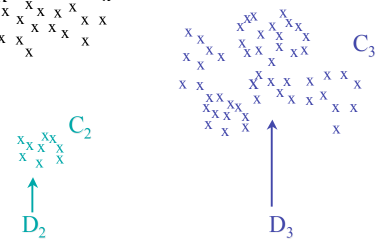


Summary / Characterization

Whe haved the point been grouped together?

What do they have in common?





Association Rule Discovery

Associations among facts, properties, or variables values

72% of those that buy salad also buy seasoning

Typical problem

Market Basket Analysis

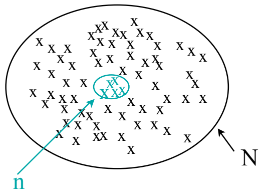
1/7/99	2/7/99
Bread	**Rice**
Peaches	**Bread**
Eggs	**Meat**
Pasta	**Peaches**
...	...
basket	basket

----> {Bread, Peaches}

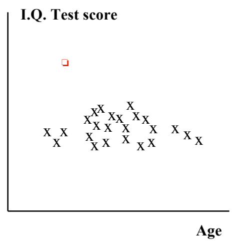
Exception detection

Detection of deviant values from "normal"

(Exceptions, Errors, ...)

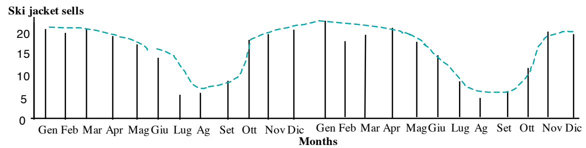


n << N



Temporal Series / Sequences

- Pattern (episode) discovery
- Trends and seasonality analysis



Representation Languages

Importance of Representation

- Data language
- Hypothesis language
- The languages delimit what can be expressed and what can be learned (representation power -> language *bias*)
- The languages determines the difficulty of learning (computational complexity)

Type of Languages

- Symbolic
  - Propositional logic
  - FOL logic
- Subsymbolic
  - Neural networks
  - Support Vector Machines

Propositional Languages

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

⇩

Data language

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"):
  - *Boolean, nominal, ordinal, continuous, hierarchical*

Nominal attributes

- Values are distinct symbols
  - Values themselves serve only as labels or names
- 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

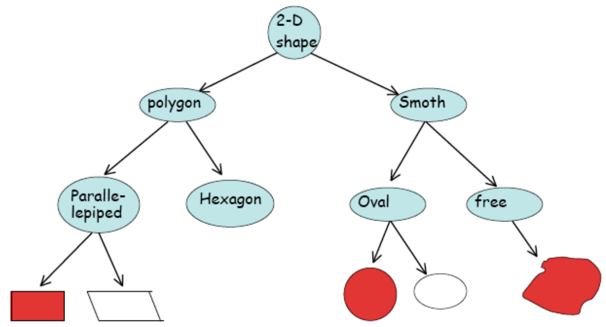
Ordinal attributes

- Impose order on values
- Example:
  attribute "temperature" in weather data
  - Values: "hot" > "mild" > "cool"
-
- Example rule:
  temperature < hot ⇨ play = yes
- Distinction between nominal and ordinal not always clear (e.g. attribute "color")

Continuous attributes

- Real valued attributes
- Example 1: attribute "temperature" expressed in degrees Fahrenheit
- Example 2: attribute "year"
- Example 3: attribute "length"
- Any algebraic operation makes sense

Hierarchical attributes



Why specify attribute types?

- ***Why Machine Learning algorithms need to know about attribute type?***
- In order 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.

Hypothesis Language

Selector

Complex

```
(InHurry = yes) and
(Temperature < 5) and
(Weather = wet or stormy)
```

Internal disjunction

Terminology from AQ (Michalski)

Cover - set of rules

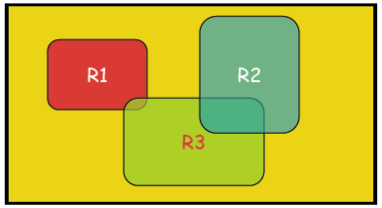
Star - set of alternative complexes

Types of Languages

- Classification rules (disjunction of conjunctions)
- Decision trees
- Decision lists
- Neural nets
- Kernels
- Associations
- Bayes discriminat function
- Bayes networks
- ………………….

Rule's Coverage

- Rule "covers" a subset of examples
- Rules may overlap
- Set of rules may not cover whole example space



<Attribute, Value> Vector

- X --> $\langle(A_1, v_1), (A_2, v_2), \ldots, (A_n, v_n)\rangle$

X --> <(Shape, *square*), (Length, *5 cm*), …, (Color, *red*)>

Example

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Covering Test

- A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition of the rule

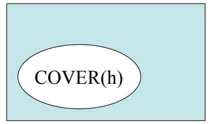
R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird
The rule R3 covers the grizzly bear => Mammal

Generalization Partial Order

h --> COVER(h)

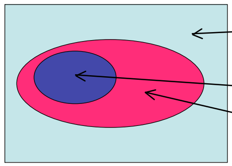


Covering Relation

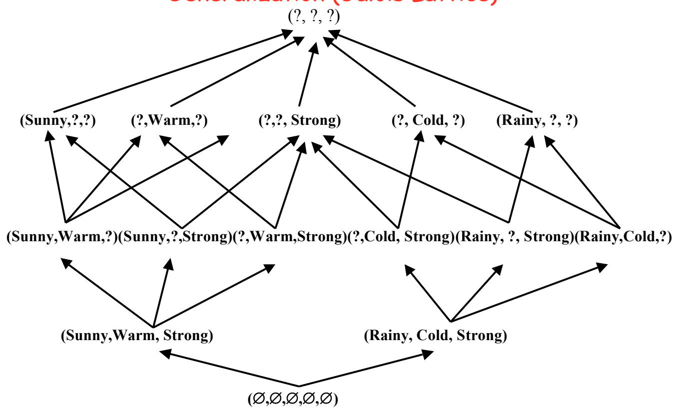
- Generalization operations induce a partial ordering on the concept space, called the ***covering relation*** -> extensional property
- Partial order = reflexive, antisymmetric and transitive
- Galois's Lattice topology

Impossible to test

$h_2 >| h_1$ iff $COVER(h_2) \supseteq COVER(h_1)$



Generalization (Galois Lattice)



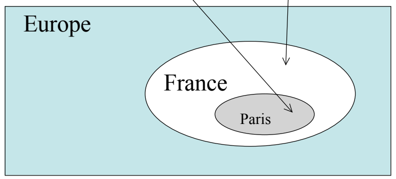
Intension vs Extension

Albert lives in Paris — Albert lives in France

Specific — **Generic**

Paris' inhabitants — France's inhabitants

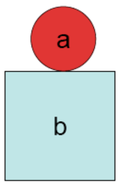
Particular — **General**



FOL Representation of Data and Hypotheses

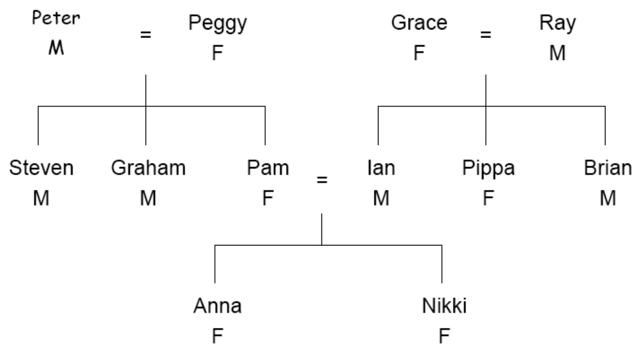
Relational Representation: Data

An instance is composed by parts, which satisfy some relations among each other



Red(a) ∧ Sphere(a) ∧ Square(b) ∧ Blue(b) ∧ **On(a,b)**

A family tree



Family tree represented as a table

Name	Gender	$Parent_1$	$Parent_2$
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

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
```

Relational Representation: Hypotheses

- Hypothesis Language Bias
- Subsets of First Order Logic
  - Horn clauses
  - Description logic

FOL Syntax

- A term denotes an object in the world.
  - **Constant:** `BobSmith, 2, Madison, Green, …`
  - **Variable:** `x, y, a, b, c, …`
  - **Function($Term_1$, …, $Term_n$):**
    `Sqrt(9), Distance(Madison,Milwaukee)`
    - maps one or more objects to another *single object*
    - can refer to an unnamed object: e.g. `LeftLegOf(John)`
    - represents a user defined *functional* relation
    - cannot be used with logical connectives
- A ground term is a term with no variables.

FOL Syntax

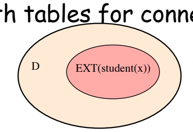
- An atom/literal is smallest expression to which a truth value can be assigned.
  - **Predicate($Term_1$, …, $Term_n$):**
    `Teacher(John, Deb), <=(Sqrt(2),Sqrt(7))`
    - maps one or more objects to a *truth value*
    - represents a user defined *truth* relation

FOL Syntax

- A sentence represents a fact in the world that is assigned a truth value.
  - atom
  - complex sentence using connectives
  - complex sentence using quantified variables

FOL Semantics

- A TRUTH value is assigne to each atom
- Boolean values => {1, 0} or {T, F}
- Truth values are propagated in complex formulas using the truth tables for connectives



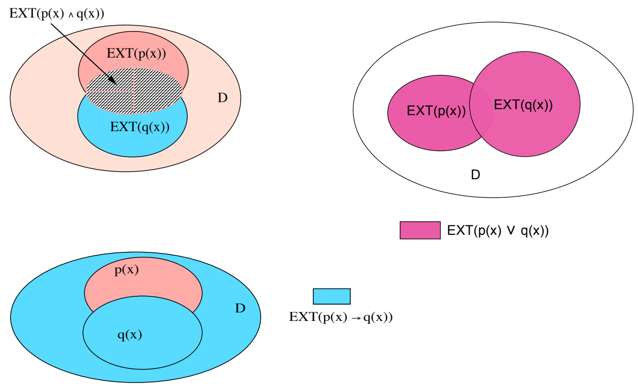
D = Set of Lyon's Inhabitants

FOL Connectives

- Conjunction AND $(\alpha \wedge \beta)$
- Disjunction OR $(\alpha \vee \beta)$
- Exclusive disjunction XOR $(\alpha \veebar \beta)$
- Negation NOT $(\neg\, \alpha)$
- Implication $(\alpha \rightarrow \beta)$

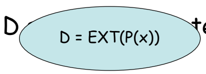
x	y	AND	OR	XOR	NOT	$\rightarrow$
0	0	0	0	0	1	1
0	1	0	1	1	1	1
1	0	0	1	1	0	0
1	1	1	1	0	0	1

Extension of Connectives

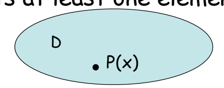


FOL Quantifiers

- Universal $\forall x\ P(x)$
  All elements id the domain D ... te P



- Existential $\exists x\ P(x)$
  In the domain P there exists at least one element that satisfies predicate P



Horn Clauses

- Horn Clause = Clause with one head and a body

$C = \{\neg p_1(x, y, \ldots), \ldots, \neg p_n(x, y, \ldots), h(x,y,\ldots)\} =$

$= \neg p_1(x, y, \ldots) \vee \ldots \vee \neg p_n(x, y, \ldots) \vee h(x,y,\ldots) =$

$= h(x,y,\ldots) \leftarrow p_1(x, y, \ldots) \wedge \ldots \wedge p_n(x, y, \ldots)$

Concepts and Predicates

- P -> set of basic predicates of a logical language L
- Ω -> set of individuals of the universe
- According to classical logic, formulas in L can be partitioned into two subsets
  - *open* formulas, with some occurrence of free variables ==> concepts [Frege]
  - *closed* ones (sentences), with no free variables.

A concept does not have a truth value associated with it; rather, it partitions Ω into the concept *extension* and its complement. The concept extension consists of the set of individuals (or tuples of individuals) which satisfy the concept definition.

Let $\varphi(x_1, x_2, \ldots, x_n)$ be a concept over the free variables $x_1, x_2, \ldots, x_n$; the extension of f is defined as follows:

$$EXT(\varphi) = \{< a_1, a_2, \ldots, a_n > \mid \varphi(a_1, a_2, \ldots, a_n) \text{ is true}\} \subseteq \Omega^n$$

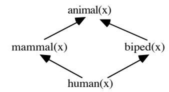
Predicates that are true of a given n-tuple $<a_1, a_2, \ldots, a_n> \in \Omega^n$ are said to belong to the "intension" of that n-tuple:

$$INT(< a_1, a_2, \ldots, a_n>) = \{p \in P \mid p(a_1, a_2, \ldots, a_n) \text{ is true}\}$$

Intension vs Extension

Intension and extension are dual properties of concept hierarchies. A certain confusion between these two aspects has influenced some of the current definitions of the *more-specific-than*.

Visiting the network bottom-up, a set inclusion relation holds between concepts extensions; visiting the network top-down, a set inclusion relation holds for concepts intensions. Any node in the hierarchy inherits *instances* from its descendants and *properties* from its ancestors.



More-specific-than relation should acknowledge that specificity (or generality) is an essentially *extensional* property and, hence, it *only pertains to concepts*, i.e., open formulas that have an associated extension. Closed formulas (i.e., sentences) are *statements about the generality of the associated concepts*. A concept and a sentence are not comparable with respect to generality.

Concepts and Sentences (1/2)

- Concepts become sentences by binding their variables. In this paper we consider three binding operators:

  (a) Grounding

  By binding a variable x_i ($1 \leq i \leq n$) to a constant a_i, the concept $\varphi(x_1, x_2, \ldots, x_n)$ becomes a new concept, ψ, with one free variable less
  $\Longrightarrow \psi(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \varphi(x_1,\ldots,x_{i-1},a_i,x_{i+1},\ldots,x_n)$

  (b) Existential quantification

  By existentially quantifying a variable x_i ($1 \leq i \leq n$), the concept $\varphi(x_1,x_2,\ldots,x_n)$ becomes a new concept, ψ, with one free variable less
  $\Longrightarrow \psi(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \exists x_i\, \varphi(x_1,x_2,\ldots,x_n)$

  (c) Universal quantification

  By universally quantifying a variable x_i ($1 \leq i \leq n$), the concept $\varphi(x_1,x_2,\ldots,x_n)$ becomes a new concept, ψ, with one free variable less
  $\Longrightarrow \psi(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \forall x_i\, \varphi(x_1,x_2,\ldots,x_n)$

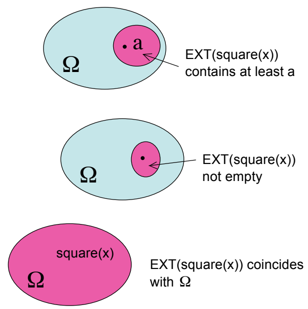
Concepts and Sentences (2/2)

- Difference between concepts (open formulas) and sentences (closed formulas)
- Concept square(x)

σ_1 = square(a), with $a \in \Omega$

$\sigma_2 = \exists x$ [square(x)]

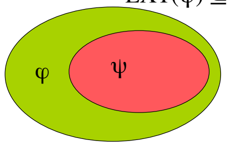
$\sigma_3 = \forall x$ [square(x)]



More-General-Than relation : Extensional Definition

Given two concepts $\varphi(x_1,x_2,\ldots,x_n)$ and $\psi(x_1,x_2,\ldots,x_n)$, and a universe of discourse Ω, the concept $\varphi(x_1, x_2, \ldots, x_n)$ will be said *more general than* the concept $\psi(x_1, x_2, \ldots, x_n)$ (denoted by $\psi |< \varphi$) iff:

$EXT(\psi) \subseteq EXT(\varphi)$



Generality may not be tested extensionally in practice

More-General-Than relation : Intensional Definition

- θ-subsumption [Plotkin, 1970]

  $\gamma_1\theta \subseteq \gamma_2$ ==> γ_1 is more general than γ_2 ==> $\gamma_2 |< \gamma_1$

  $\theta = \{x_1/a_1, \ldots, x_n/a_n\}$

- Implication

  $\gamma_2 \rightarrow \gamma_1$ ==> γ_1 is more general than γ_2 ==> $\gamma_2 |< \gamma_1$

Generalization Rules [Michalski]

1. Turning a constant to a variable

   square(a) |< square(x)

2. Dropping conditions

   square(x) ∧ small(x) |< small(x)

3. Introducing disjunction

   square(x) |< square(x) ∨ rectangle(x)

4. Extending/Closing intervals

   length(x, [0, 0.5]) |< length(x, [0, 1])

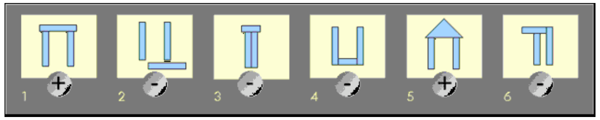
   length (x, [0, 0.5] ∨ [0.7, 1]) |< length(x, [0, 1])

5. Climbing a generalization hierarchy

   square(x) ∧ small(x) |< polygon (x) ∧ small(x)

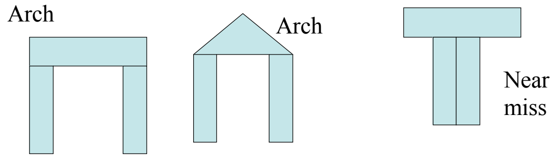
Hystory of FOL Machine Learning

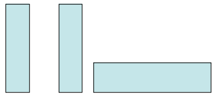
Winston's ARCH



Winston made a program that could learn the concept of an arch

Arch examples



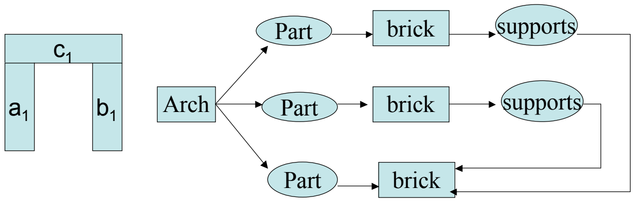


Near miss

How does machine recognize an arch? What concept should it learn in order to recognize an arch?

How do you teach machine to recognize an arch?

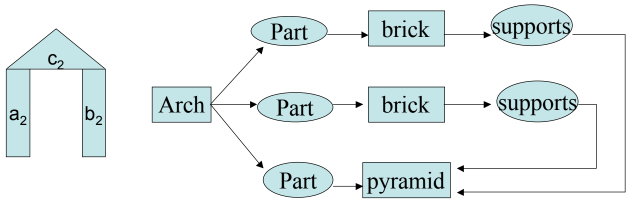
Concept of Arch



Representation Language - Input data: graph

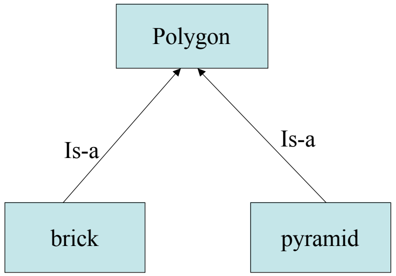
brick(a_1) ∧ brick(b_1) ∧ brick(c_1) ∧ supports(a_1, c_1) ∧ supports(b_1, c_1)

Pyramid arch

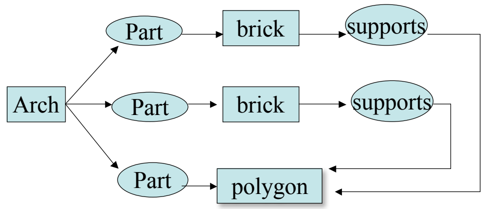


brick(a_2) ∧ brick(b_2) ∧ pyramid(c_2) ∧ supports(a_2, c_2) ∧ supports(b_2, c_2)

Background Knowledge



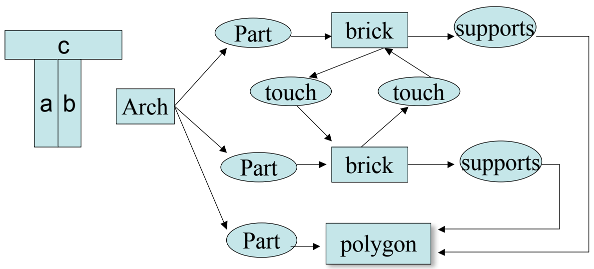
Generalization



Generalizing a network by replacing node or links names with a more general concept

brick(a) ∧ brick(b) ∧ polygon(c) ∧ supports(a, c) ∧ supports(b, c)

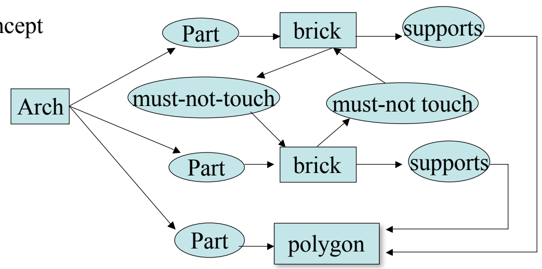
Description of a Near-Miss



brick(a) ∧ brick(b) ∧ polygon(c) ∧ supports(a, c) ∧ supports(b, c) ∧
∧ touch(a,b) ∧ touch(b,a)

Specialization: Excluding the near miss

Concept

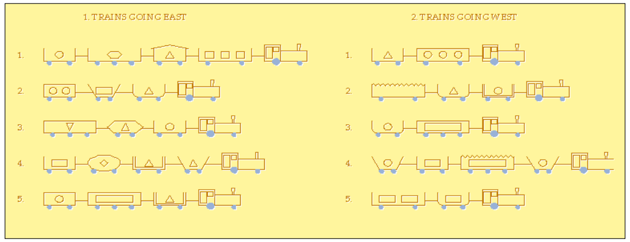


Operations: specializing a network by **adding links**

brick(a) ∧ brick(b) ∧ polygon(c) ∧ supports(a, c) ∧ supports(b, c) ∧
∧ must-not-touch(a,b) ∧ must-not-touch(b,a)

Michalski's INDUCE

- Language VL_2 -> Extension of VL_1
- More-general-than --> Generalization rules
- Transformation of examples into propositional description -> Use AQ -> Back to FOL



Multiple relations

This is structured data

has_car

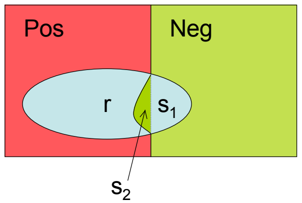
Train	Car
t1	c11
t1	c12
t1	c13
t1	c14
t2	c21
...	...

car_properties

Car	Length	Shape	Axes	Roof	...
c11	short	rectangle	2	none	...
c12	long	rectangle	3	none	...
c13	short	rectangle	2	peaked	...
c14	long	rectangle	2	none	...
c21	short	rectangle	2	flat	...
...	...	...	...	...	...

Vere' Toth

- More-general-than --> θ-subsumption
- Counterfactuals -> Nested exceptions



$r \wedge \neg (s_1 \wedge \neg s_2) \rightarrow \omega$

exception

exception of exception

Other Approaches

- SMART+ [Giordana et al.]
  - Hybrid strategy SBL + EBL
- G-Net [Giordana et al.]
  - Genetic Algorithms : Theory of Niches and Species to learn multimodal concepts
  - Cromosome = FOL formulas with internal disjunction
  - Distributed evolution and covering test

FOIL

Quinlan, 1990

FOIL

- FOIL learns rules that predict when the target is true; sequential covering learns both rules that are true and false.
- FOIL performs a hill-climbing search; sequential covering performs a beam search.
- FOIL rules are more expressive than Horn Clauses, because the precondition can have negated literals.

FOIL

FOIL's method is very similar to sequential covering.

FOIL(target-predicate, predicates, examples)
1. Pos ← Those examples where target-predicate is true
2. Neg ← Those examples where target-predicate is false
3. Learned-Rules ← {}
4. ***While*** Pos ***do***
   Learn a new rule NewRule
5. Learned-Rules ← Learned-Rules + NewRule
6. Pos ← Pos – {members of Pos covered by NewRule}
7. ***Return*** Learned-Rules

FOIL

- Foil does a general to specific search on each rule by starting with a NULL precondition and adding more literals (hill-climbing).

Learning New Rules

NewRule

1. NewRule ← If {} then target-predicate
2. CoveredNeg ← Neg
3. ***While*** CoveredNeg ***do***
   a. candidate-literals ← new literals for NewRule
   b. BestLiteral ← argmax Foil_Gain L in candidate-literals (L,NewRule)
   c. Add BestLiteral to preconditions of NewRule
   d. CoveredNeg ← subset of CoveredNeg satisfied by NewRule

   End While

Generating Specializations

Assume our current rule is as follows:

$P(x_1, x_2, \ldots, x_k) \leftarrow L_1, \ldots, L_n$

Where each Li is a literal and $P(x_1, x_2, \ldots, x_k)$ is the head or postcondition. FOIL considers new literals L_{n+1} to add to the rule such as:

- *Predicates*: $Q(v_1,\ldots,v_r)$ where Q is a predicate and v_i is an existing or new variable (at least one v_i must be already present).
- *Functions*: $Equal(x_j,x_k)$ where x_j and x_k are present in the rule.
- *Negated literals*.

Example

We wish to learn the target predicate GrandDaughter(x,y)

Our predicates are Father(x,y) and Female(x)
Our constants are Victor, Sharon, Bob, and Tom.

We start with the most general rule:

GrandDaughter(x,y) ←

Example

Possible literals we could add:

Equal(x,y), Female(x), Female(y), Father(x,y) … and their negations

Assume we find the best choice is

GrandDaughter(x,y) ← Father(y,z)

Example

We add the best candidate literal and continue adding literals until we generate a rule like the following:

GrandDaughter(x,y) ← Father(y,z) ∧ Father(z,x) ∧ Female(x)

At this point we remove all positive examples covered by the rule and begin the search for a new rule.

Choosing the Best Literal

Consider the target predicate:

GrandDaughter(x,y) ←

Consider all bindings. Example {x/Bob, y/Sharon}

Choosing the Best Literal

Now compare rule R before adding a literal and after adding a literal.

Foil-Gain(L,R) = t [$\log_2 (p_1 / p_1 + n_1)$ - $\log_2 (p_0 / p_0 + n_0)$]

t: positive bindings of rule R still covered after adding literal L
p_0: positive bindings of rule R n_0: negative bindings of rule R
p_1: positive bindings of rule R' n_1: negative bindings of rule R'

Inductive Logic Programming (ILP)

Muggleton,

De Raedt

Morik

Sebag

Saitta et al.

Esposito et al.

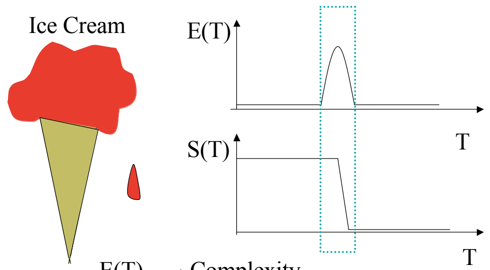
…….

Complexity of Learning and Phase Transitions

Complexity of Learning

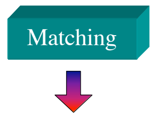
- Classical Complexity Theory -> Worst-case
- PAC Framework -> Polynomial learnability
- Complexity distribution -> "Typical" complexity
  -> Phase Transitions

Phase Transitions



E(T) → Complexity
S(T) → Probability of Solution
T → Some Order Parameter

Matching

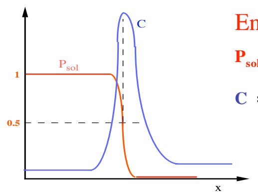


Matching problem Match(φ, U)

Given a FOL formula φ and a universe U,
does there exists a model of φ in U?

Main source of complexity in relational learning

Phase Transition



Ensemble of problems

P_{sol} = Probability that a randomly generated problem is solvable
C = Complexity to find one solution or to prove that none exists

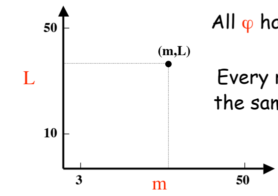
Stochastic problem generation

When the size of the problem increases:
- The transition in P_{sol} becomes sharper
- The complexity peak becomes higher

Shift from Worst-case complexity to Typical complexity
Much more detailed view of complexity issues than classical complexity theory

Exploration of the (m,L) plane

Ensemble of artificial matching problems Match(φ,U) generated according to a specified stochastic model



All φ have the same number n of variables
(n = 4, 6, 10, 12, 14)

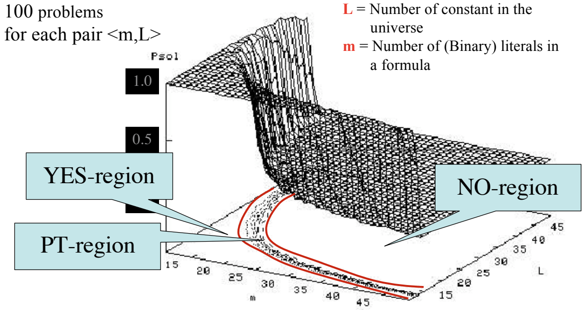
Every relation in any universe U contains the same number N of tuples
(N = 30, 50, 100, 130)

m = Number of predicates in φ
L = Number of constants in U

Phase Transition in Plane (m, L)

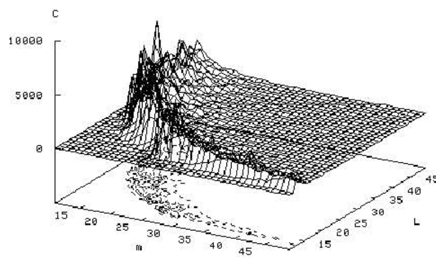
100 problems
for each pair <m,L>

L = Number of constant in the universe
m = Number of (Binary) literals in a formula



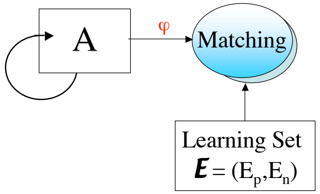
Relation Size: N = 100 Number of variables: n = 10

Matching Complexity



Relation Size: N = 100
Number of variables: n = 10

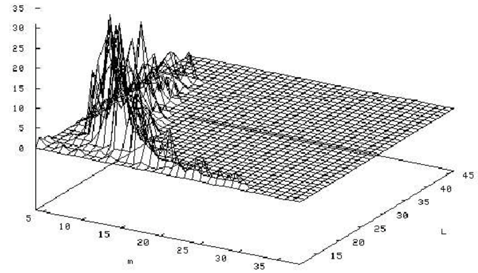
Relational Learning: Hypothesis Generation



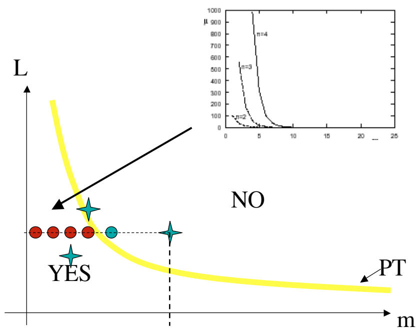
The matching problem ensemble is generated by A itself according to its exploration strategy

R = Number of generated clauses φ
S = Cardinality of the learning set E
R* S = Number of matching problems

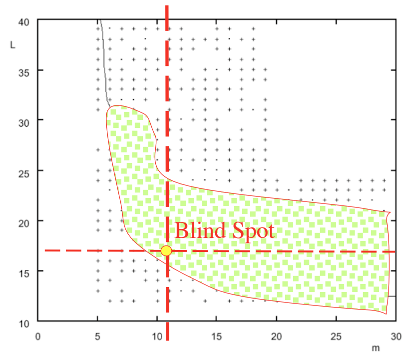
All solutions are around the PT



Why does this happens?



451 Relational Problems



Learners
FOIL,
Smart+,
G-Net
Still
Progol