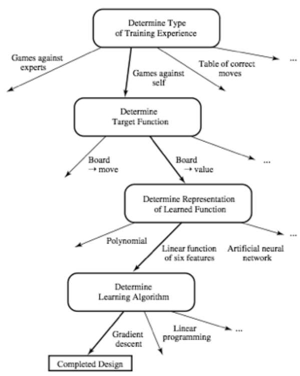
Concept learning

Maria Simi, 2010/2011

Plan

- Introduction to machine learning
  - Task definition
  - When appropriate and when not appropriate
  - Learning methodology: design, experiment, evaluation
  - Learning issues: representing hypothesis
  - Taxonomy of learning models
- Inductive learning: basic schemas (supervised)
  - Concept learning
    - Based on a logical representation of the hypothesis
    - Formalization of nductive bias
  - Decision trees learning
    - The problem of overfitting

Design of a learning system



Definition of concept learning

- *Task*: learning a category description (*concept*) from a set of positive and negative training examples.
- Inferring a boolean function $c: X \rightarrow \{0, 1\}$
- A search problem for best fitting hypothesis in a hypotheses space
- Search may be made more efficient by structuring the hypotheses space (general to specific ordering)

Sport example

- Concept: *Days in which Aldo can enjoy water sport*

  Attributes: *Sky: Sunny, Cloudy, Rainy*
  AirTemp: Warm, Cold
  Humidity: Normal, High
  Wind: Strong, Weak
  Water: Warm, Cool
  Forecast: Same, Change
- Instances in the training set (out of the 96 possible):

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Hypotheses representation

- h is a set of constraints on attributes:
  - a specific value: e.g. *Water = Warm*
  - any value allowed: e.g. *Water = ?*
  - no value allowed: e.g. *Water = Ø*
- Example hypothesis:

Sky	*AirTemp*	*Humid*	*Wind*	*Water*	*Forecast*
⟨*Sunny,*	?,	?,	*Strong,*	?,	*Same*⟩

- H, hypotheses space, all possible h

Hypothesis satisfaction

- An instance x *satisfies* an hypothesis h iff all the constraints expressed by h are satisfied by the attribute values in x.
- Example 1:
  x_1: ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩
  h_1: ⟨*Sunny*, ?, ?, *Strong*, ?, *Same*⟩ Satisfies? Yes
- Example 2:
  x_2: ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩
  h_2: ⟨*Sunny*, ?, ?, ∅, ?, *Same*⟩ Satisfies? No

Formal task description

- Given:
  - X all possible days, as described by the attributes
  - A set of hypothesis H, a conjunction of constraints on the attributes (including '?' and '∅'), representing a function $h: X \rightarrow \{0, 1\}$
    $[h(x) = 1$ if x satisfies h; $h(x) = 0$ if x does not satisfy $h]$
  - A target concept: $c: X \rightarrow \{0, 1\}$ where
    $c(x) = 1$ iff *EnjoySport = Yes*;
    $c(x) = 0$ iff *EnjoySport = No*;
  - A training set of possible instances D: $\{\langle x, c(x)\rangle\}$
- Goal:
  - find a hypothesis h in H such that
    $h(x) = c(x)$ for all x in D (hopefully in X)

The inductive learning assumption

- We can at best guarantee that the output hypothesis fits the target concept over the training data
- *Assumption*: an hypothesis that approximates well the training data will also approximate the target function over unobserved examples
- i.e. given a significant training set, the output hypothesis is able to make predictions

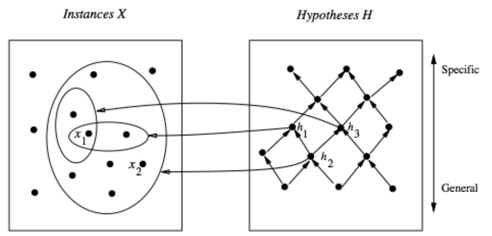
Concept learning as search

- Concept learning is a task of searching an hypotheses space
- The representation chosen for hypotheses determines the search space
- In the example we have:
  - $3 \times 2^5 = 96$ possible instances (6 attributes)
  - $1 + 4 \times 3^5 = 973$ possible hypothesis
    (considering that all the hypothesis with some ∅ are semantically equivalent)

General to specific ordering

- $h_1 = \langle Sunny, ?, ?, Strong, ?, ?\rangle$
- $h_2 = \langle Sunny, ?, ?, ?, ?, ?\rangle$
- Any instance classified positive by h_1 will also be classified positive by h_2
- h_2 is more general than h_1
- Definition: $h_j \geq_g h_k$ *iff* $(\forall x \in X)\ [(h_j = 1) \rightarrow (h_2 = 1)]$
  $\geq_g$ *more general than or equal to*
  $>_g$ *strictly more general than*
- Most general hypothesis: ⟨?, ?, ?, ?, ?, ?⟩
- Most specific hypothesis: ⟨∅, ∅, ∅, ∅, ∅, ∅⟩

General to specific ordering: induced structure



x_1= <*Sunny, Warm, High, Strong, Cool, Same*>
x_2= <*Sunny, Warm, High, Light, Warm, Same*>
h_1= <*Sunny*, ?, ?, *Strong*, ?, ?>
h_2= <*Sunny*, ?, ?, ?, ?, ?>
h_3= <*Sunny*, ?, ?, ?, *Cool*, ?>

Find-S: finding the most specific hypothesis

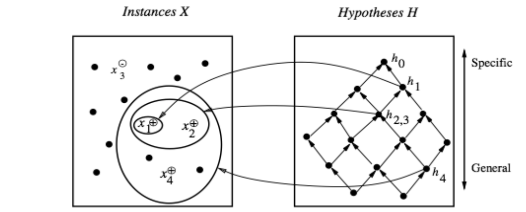
1. Initialize h to the most specific hypothesis in H
2. For each positive training instance:

   For each attribute constraint a_i in h:

   If the constraint a_i is satisfied by x *then do nothing*

   else replace a_i in h by the next more general constraint that is satified by x
3. Output hypothesis h

Find-S in action



x_1 = *<Sunny Warm Normal Strong Warm Same>*, +
x_2 = *<Sunny Warm High Strong Warm Same>*, +
x_3 = *<Rainy Cold High Strong Warm Change>*, -
x_4 = *<Sunny Warm High Strong Cool Change>*, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
h_1 = *<Sunny Warm Normal Strong Warm San*
h_2 = *<Sunny Warm ? Strong Warm Same>*
h_3 = *<Sunny Warm ? Strong Warm Same>*
h_4 = *<Sunny Warm ? Strong ? ? >*

Properties of *Find-S*

- *Find-S* is guaranteed to output the *most specific hypothesis* within *H* that is consistent with the *positive* training examples
- The final hypothesis will also be consistent with the negative examples
- Problems:
  - We cannot say if the learner converged to the correct target.
  - Why choose the most specific?
  - If the training examples are inconsistent, the algorithm can be mislead. No tolerance to rumor.
  - There can be more than one maximally specific hypotheses
  - Negative example are not considered

Candidate elimination algorithm: the idea

- *The idea*: output a description of the set of *all hypotheses consistent* with the training examples (correctly classify training examples).
- *Version space*: a representation of the set of hypotheses which are *consistent* with *D*
  1. an explicit list of hypotheses (List-Than-Eliminate)
  2. a compact representation of hypotheses which exploits the *more_general_than* partial ordering (Candidate-Elimination)

Version space

- The version space $VS_{H,D}$ is the subset of the hypothesis from H *consistent* with the training example in D

  $VS_{H,D} \equiv \{h \in H \mid Consistent(h, D)\}$
- An hypothesis h is consistent with a set of training examples D iff $h(x) = c(x)$ for each example in D

  $Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$

  Note: "*x satisfies h*" ($h(x)=1$) different from "*h consistent with x*"

  In particular when an hypothesis h is consistent with a negative example $d = \langle x, c(x)=No\rangle$, then x must not satisfy h

The *List-Then-Eliminate* algorithm

Version space as list of hypotheses

1. $VersionSpace \leftarrow$ a list containing every hypothesis in H
2. For each training example, $\langle x, c(x)\rangle$

   Remove from $VersionSpace$ any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$

- Problems
  - The hypothesis space must be finite
  - Enumeration of all the hypothesis, rather inefficient

A compact representation for *Version Space*

S: { <Sunny, Warm, ?, Strong, ?, ?> }

G: { <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

Note: The output of *Find-S* is just ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

- Version space represented by its most general members G and its most specific members S *(boundaries)*

General and specific boundaries

- The *Specific boundary*, *S*, of version space $VS_{H,D}$ is the set of its minimally general (most specific) members
  S ≡ $\{s \in H \mid Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$
  Note: any member of *S* is satisfied by all positive examples, but more specific hypotheses fail to capture some
- The *General boundary*, *G*, of version space $VS_{H,D}$ is the set of its maximally general members
  G ≡ $\{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$
  Note: any member of G is satisfied by no negative example but more general hypothesis cover some negative example

Version Space representation theorem

- G and S completely define the Version Space
- *Theorem*: Every member of the version space is in S or G or lies between these boundaries
  $VS_{H,D}=\{h \in H \mid (\exists s \in S)\ (\exists g \in G)\ (g \geq_g h \geq_g s)\}$
  where $x \geq_g y$ means x is more general or equal to y
  Sketch of proof:

⇐ If $g \geq_g h \geq_g s$, since *s* is in S and $h \geq_g s$, *h* is satisfied by all positive examples in *D*; *g* is in G and $g \geq_g h$, then *h* is satisfied by no negative examples in *D*; therefore *h* belongs to $VS_{H,D}$

⇒ It can be proved by assuming a consistent *h* that does not satisfy the right-hand side and by showing that this leads to a contradiction; dovrebbe cadere fuori dal reticolo …

Candidate elimination algorithm-1

S ← minimally general hypotheses in H,
G ← maximally general hypotheses in H
Initially any hypothesis is still possible
$S_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ $\quad G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$
For each training example d, do:
If d is a *positive* example:

1. Remove from G any h inconsistent with d
2. *Generalize*(S, d)

If d is a *negative* example:

1. Remove from S any h inconsistent with d
2. *Specialize*(G, d)

Note: when $d = \langle x, \text{No} \rangle$ is a negative example, an hypothesis h is inconsistent with *d* iff h *satisfies* x

Candidate elimination algorithm-2

Generalize(S, d): d is *positive*

For each hypothesis s in S not consistent with d:

1. Remove s from S
2. Add to S all minimal generalizations of s consistent with d and having a generalization in G
3. Remove from S any hypothesis with a more specific h in S

Specialize(G, d): d is *negative*

For each hypothesis g in G not consistent with d:

1. Remove g from G
2. Add to G all minimal specializations of g consistent with d and having a specialization in S
3. Remove from G any hypothesis having a more general hypothesis in G

Example: initially

S_0: $\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset. \emptyset \rangle$

G_0 $\langle ?, ?, ?, ?, ?, ? \rangle$

Example:

after seing ⟨*Sunny,Warm, Normal, Strong, Warm, Same* ⟩ +

S_0: ⟨∅, ∅, ∅, ∅, ∅. ∅⟩

S_1: ⟨*Sunny,Warm, Normal, Strong, Warm, Same*⟩

G_0, G_1 ⟨?, ?, ?, ?, ?, ?⟩

Example:

after seing ⟨*Sunny,Warm, High, Strong, Warm, Same*⟩ +

S_1: ⟨*Sunny,Warm, Normal, Strong, Warm, Same*⟩

S_2: ⟨*Sunny,Warm, ?, Strong, Warm, Same*⟩

G_1, G_2 ⟨?, ?, ?, ?, ?, ?⟩

Example:

after seing ⟨*Rainy, Cold, High, Strong, Warm, Change* ⟩-

S_2, S_3: ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

G_3: ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩ ⟨*?, ?, ?, ?, ?, Same*⟩

G_2: ⟨?, ?, ?, ?, ?, ?⟩

Example:

after seing ⟨*Sunny, Warm, High, Strong, Cool Change* ⟩ +

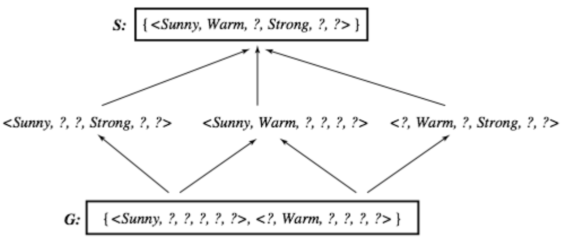
S_3 ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

S_4 ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

G_4: ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩

G_3: ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩ ⟨*?, ?, ?, ?, ?, Same*⟩

Learned Version Space



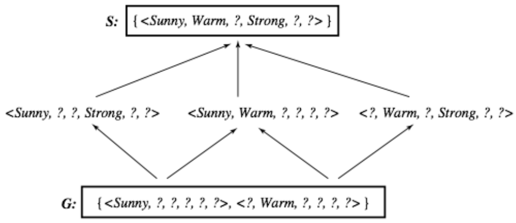
Observations

- The learned Version Space correctly describes the target concept, provided:
  1. There are no errors in the training examples
  2. There is some hypothesis that correctly describes the target concept
- If *S* and *G* converge to a single hypothesis the concept is exactly learned
- In case of errors in the training, useful hypothesis are discarded, no recovery possible
- An empty version space means no hypothesis in H is consistent with training examples

Ordering on training examples

- The learned version space does not change with different orderings of training examples
- Efficiency does
- Optimal strategy (if you are allowed to choose)
  - Generate instances that satisfy half the hypotheses in the current version space. For example:
    $\langle$*Sunny, Warm, Normal, Light, Warm, Same*$\rangle$ satisfies 3/6 hyp.
  - Ideallt the VS can be reduced by half at each experiment
  - Correct target found in $\lceil log_2|VS|\rceil$ experiments

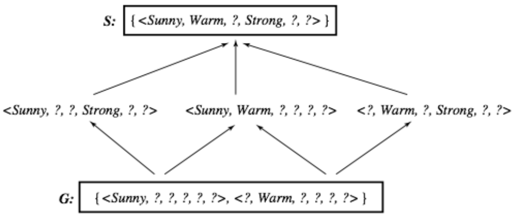
Use of partially learned concepts



$\langle$*Sunny Warm Normal Strong Cool Change*$\rangle$

Classified as *positive* by all hypothesis, since satisfies any hypothesis in S

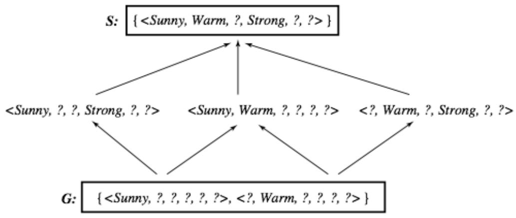
Classifying new examples



$\langle$*Rainy Cool Normal Light Warm Same*$\rangle$

Classified as *negative* by all hypothesis, since does not satisfy any hypothesis in G

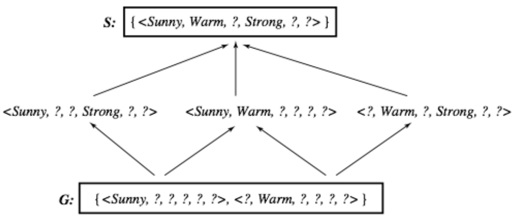
Classifying new examples



$\langle$*Sunny Warm Normal Light Warm Same*$\rangle$

Uncertain classification: half hypothesis are consistent, half are not consistent

Classifying new examples



$\langle$*Sunny, Cold, Normal, Strong, Warm, Same*$\rangle$

4 hypothesis not satisfied; 2 satisfied
Probably a negative instance. Majority vote?

Hypothesis space and bias

- What if H does not contain the target concept?
- Can we improve the situation by extending the hypothesis space?
- Will this influence the ability to generalize?
- These are general questions for inductive inference, addressed in the context of Candidate-Elimination
- Suppose we include in H every possible hypothesis … including the ability to represent disjunctive concepts

Extending the hypothesis space

	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoyS
1	Sunny	Warm	Normal	Strong	Cool	Change	YES
2	Cloudy	Warm	Normal	Strong	Cool	Change	YES
3	Rainy	Warm	Normal	Strong	Cool	Change	NO

- No hypothesis consistent with the three examples with the assumption that the target is a conjunction of constraints
  ⟨?, *Warm, Normal, Strong,Cool, Change*⟩ is too general
- Target concept exists in a different space H', including disjunction and in particular the hypothesis
  Sky=Sunny or *Sky=Cloudy*

An unbiased learner

- Every possible subset of X is a possible target
  $|H'| = 2^{|X|}$, or 2^{96} (vs $|H| = 973$, a strong bias)
- This amounts to allowing conjunction, disjunction and negation
  ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ∨ <*Cloudy, ?, ?, ?, ?, ?*⟩
- We are guaranteed that the target concept exists
- No generalization is however possible!!!
  Let's see why …

No generalization without bias!

- *VS* after presenting three positive instances x_1, x_2, x_3, and two negative instances x_4, x_5
  $S = \{(x_1 \vee x_2 \vee x_3)\}$
  $G = \{\neg(x_4 \vee x_5)\}$
  … all subsets including $x_1 x_2 x_3$ and not including $x_4 x_5$
- We can only classify precisely examples already seen!
  - Unseen instances, e.g. x, are classified positive (and negative) by half of the hypothesis
  - For any hypothesis h that classifies x as positive, there is a complementary hypothesis $\neg h$ that classifies x as negative

No inductive inference without a bias

- *A learner that makes no a priori assumptions regarding the identity of the target concept, has no rational basis for classifying unseen instances*
- The *inductive bias* of a learner are the assumptions that justify its inductive conclusions or the policy adopted for generalization
- Different learners can be characterized by their bias
- See next for a more formal definition of *inductive bias* …

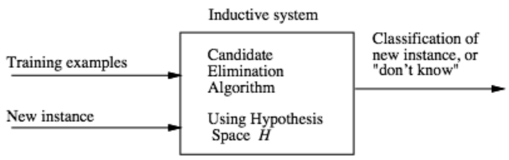
Inductive bias: definition

- Given:
  - a concept learning algorithm *L* for a set of instances *X*
  - a concept c defined over *X*
  - a set of training examples for c: $D_c = \{\langle x, c(x)\rangle\}$
  - $L(x_i, D_c)$ outcome of classification after learning
- Inductive inference ($\succ$):
  $D_c \wedge x_i \succ L(x_i, D_c)$
- The *inductive bias* is defined as a minimal set of assertions *B*, such that ($\vdash$ for deduction)
  $\forall (x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$

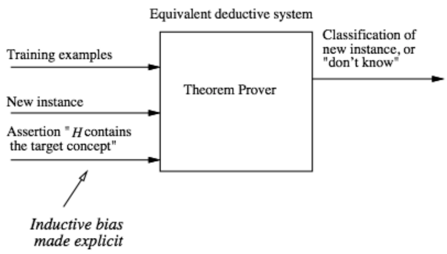
Inductive bias of Candidate-Elimination

- Assume *L* is defined as follows:
  - compute $VS_{H,D}$
  - classify new instance by complete agreement of all the hypotheses in *VS*
- Then the inductive bias of Candidate-Elimination is simply $B \equiv (c \in H)$
- In fact by assuming $c \in H$:
  1. $c \in VS_{H,D}$, in fact $VS_{H,D}$ includes all hypotheses in *H* consistent with D
  2. $L(x_i, D_c)$ outputs a classification "by complete agreement", hence any hypothesis, including c, outputs $L(x_i, D_c)$

Inductive system



Equivalent deductive system



Each learner has an inductive bias

- Three learner with three different inductive bias:
  1. Rote learner: no inductive bias, just stores examples and is able to classify only previously observed examples
  2. *CandidateElimination*: the concept is a conjunction of constraints.
  3. *Find-S*: the concept is in *H* (a conjunction of constraints) plus "all instances are negative unless seen as positive examples"

Bibliography

- Machine Learning, Tom Mitchell, Mc Graw-Hill International Editions, 1997 (Cap 2).