Concept Learning

- Learning from examples
- General-to specific ordering of hypotheses
- Version spaces and candidate elimination algorithm
- Inductive bias

What's Concept Learning?

- infer the general definition of some concept, given examples labeled as members or nonmembers of the concept.
- example: learn the category of "car" or "bird"
- concept is often formulated as boolean-valued function
- can be formulated as a problem of searching a hypothesis space

Training Examples for Concept Enjoy Sport

Concept: "days on which my friend Tom enjoys his favourite water sports"

Task: predict the value of "Enjoy Sport" for an arbitrary day based on the values of the other attributes

attributes

example

Sky	Temp	Humid	Wind	Water	Fore-cast	Enjoy Sport
Sunny	Warm	Norr[illegible]	[illegible]	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Representing Hypothesis

- Hypothesis h is described as a conjunction of constraints on attributes
- Each constraint can be:
  - A specific value : e.g. *Water=Warm*
  - A don't care value : e.g. *Water=?*
  - No value allowed (null hypothesis): e.g. *Water=Ø*
- Example: hypothesis h

	Sky	Temp	Humid	Wind	Water	Forecast	
<	Sunny	?	?	Strong	?	Same	>

Prototypical Concept Learning Task

Given:

- Instance Space X : Possible days decribed by the attributes *Sky, Temp, Humidity, Wind, Water, Forecast*
- Target function c: EnjoySport X $\rightarrow$ {0,1}
- Hypothese Space H: conjunction of literals e.g.

  < Sunny ? ? Strong ? Same >
- Training examples D : positive and negative examples of the target function: $<x_1,c(x_1)>, \ldots , <x_n,c(x_n)>$

Determine:

- A hypothesis h in H such that h(x)=c(x) for all x in D.

Inductive Learning Hypothesis

- Any hypothesis found to approximate the target function well over the training examples, will also approximate the target function well over the unobserved examples.

$\Rightarrow$ find the hypothesis that best fits the training data

Number of Instances, Concepts, Hypotheses

- Sky: Sunny, Cloudy, Rainy
- AirTemp: Warm, Cold
- Humidity: Normal, High
- Wind: Strong, Weak
- Water: Warm, Cold
- Forecast: Same, Change

#distinct instances : 3*2*2*2*2*2 = 96

#distinct concepts : 2^{96}

#syntactically distinct hypotheses : 5*4*4*4*4*4=5120

#semantically distinct hypotheses : 1+4*3*3*3*3*3=973

organize the search to take advantage of the structure of the hypothesis space to improve running time

General to Specific Ordering

- Consider two hypotheses:
  - h_1=< Sunny,?,?,Strong,?,?>
  - h_2=< Sunny,?,?,?,?,?>
- Set of instances covered by h_1 and h_2:

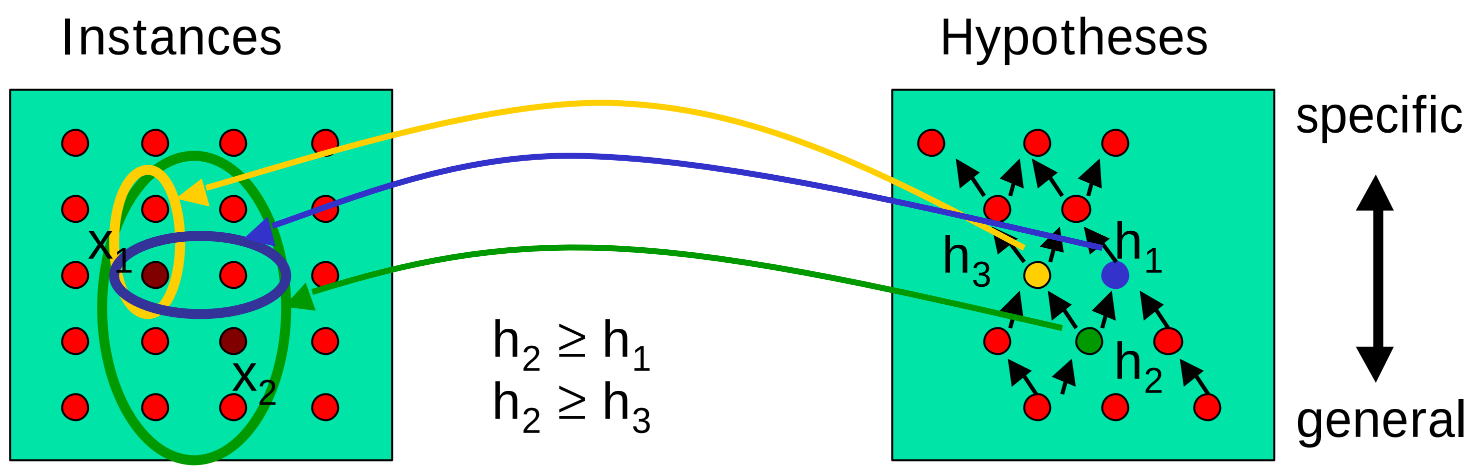
h_2 imposes fewer constraints than h_1 and therefore classifies more instances x as positive h(x)=1. h_2 is a more general concept.

Definition: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more general than or equal to** h_k (written $h_j \geq h_k$) if and only if

$$\forall x \in X : [\ (h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

- The relation $\geq$ imposes a partial order over the hypothesis space H that is utilized in many concept learning methods.

Instance, Hypotheses and "more general"



x_1=< Sunny,Warm,High,Strong,C

x_2=< Sunny,Warm,High,Light,Wa

h_1 is a minimal specialization of h_2

h_2 is a minimal generalization of h_1

h_3=< Sunny,?,?,?,Cool,?>

Find-S Algorithm

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint a_i in h

     If the constraint a_i in h is satisfied by x

     then do nothing

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

minimal generalization to cover x

Constraint Generalization

Arrtibute: Sky

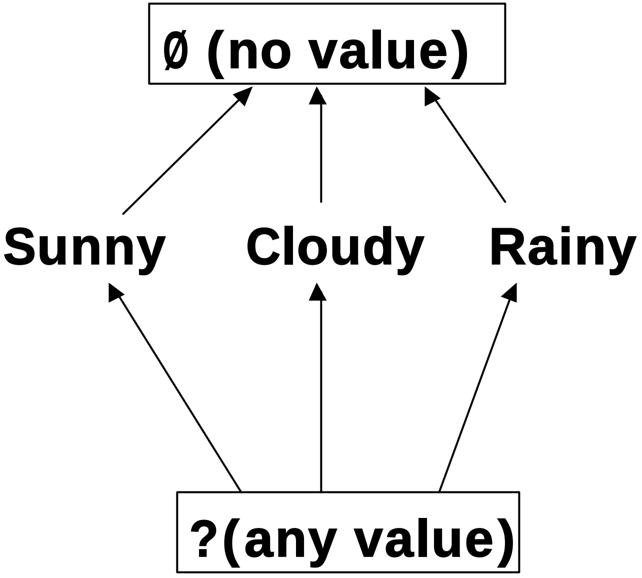
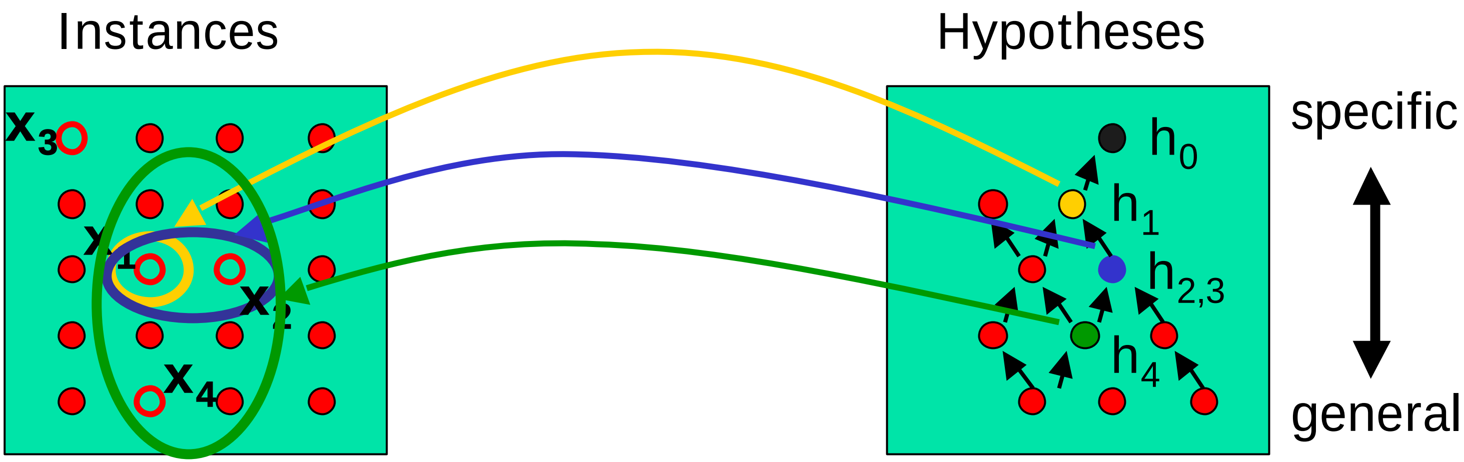


Illustration of Find-S



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=< Ø , Ø , Ø , Ø , Ø , Ø ,>

h_1=< Sunny,Warm,Normal, Strong,Warm,Same>

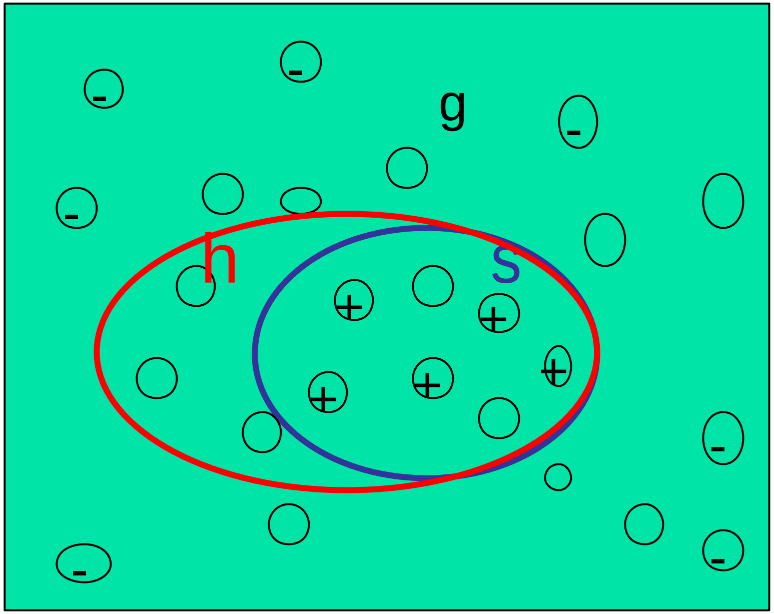
$h_{2,3}$=< Sunny,Warm,?, Strong,Warm,Same>

h_4=< Sunny,Warm,?, Strong,?,?>

Properties of Find-S

- Hypothesis space described by conjunctions of attributes
- Find-S will output the most specific hypothesis within H that is consistent with the positve training examples
- The output hypothesis will also be consistent with the negative examples, provided the target concept is contained in H. *(why?)*

Why Find-S Consistent?



h is consistent with D, then h>s;

Complaints about Find-S

- Can't tell if the learner has converged to the target concept, in the sense that it is unable to determine whether it has found the *only* hypothesis consistent with the training examples. (more examples get better approximation)
- Can't tell when training data is inconsistent, as it ignores negative training examples. (prefer to detect and tolerate errors or noise)
- Why prefer the most specific hypothesis? Why not the most general, or some other hypothesis? (more specific less likely coincident)
- What if there are multiple maximally specific hypothesis? (all of them are equally likely)

Version Spaces

- A hypothesis h is **consistent** with a set of training examples D of target concept if and only if h(x)=c(x) for each training example <x,c(x)> in D.

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

- The **version space**, $VS_{H,D}$, with respect to hypothesis space H, and training set D, is the subset of hypotheses from H consistent with all training examples:

$$VS_{H,D} = \{h \in H \mid Consistent(h,D) \}$$

List-Then Eliminate Algorithm

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

   remove from *VersionSpace* any hypothesis that is inconsistent with the training example $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

inefficient as it does not utilize the structure of the hypothesis space.

Example Version Space

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

<Sunny,?,?,Strong,?,?> <Sunny,Warm,?,?,?,?> <?,Warm,?,Strong,?,?>

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

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> +

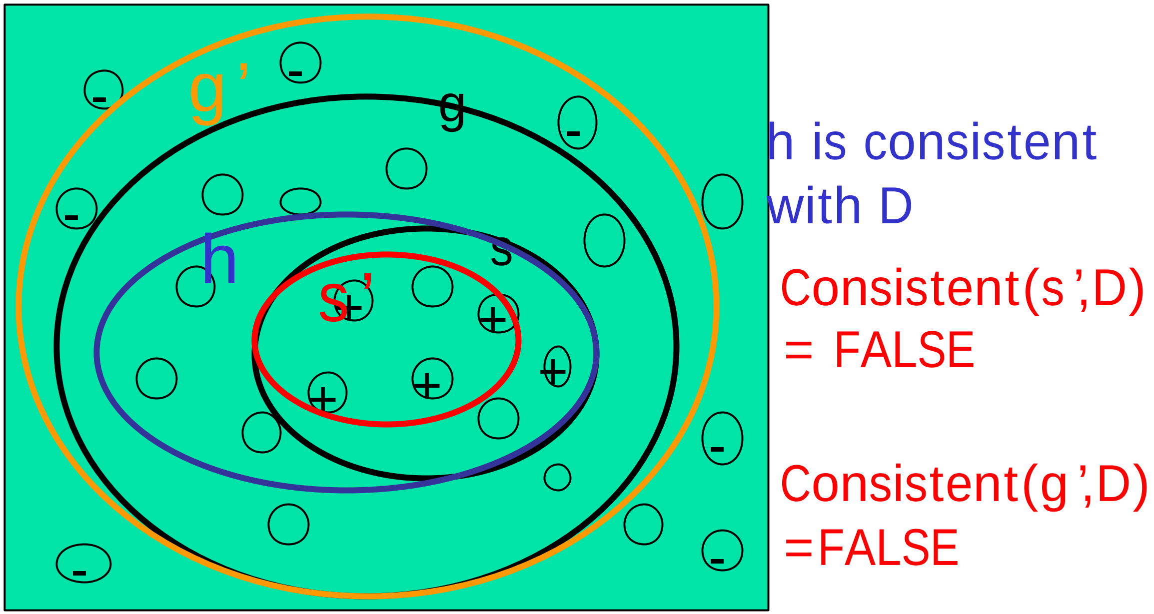
Representing Version Spaces

- The **general boundary**, G, of version space $VS_{H,D}$ is the set of maximally general hypotheses.
- The **specific boundary**, S, of version space $VS_{H,D}$ is the set of maximally specific hypotheses.
- Every hypothesis of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H \mid (\exists\, s \in S)\ (\exists\, g \in G)\ (g \geq h \geq s)$$

where $x \geq y$ means x is more general or equal than y

Boundaries of Version Space



Candidate Elimination Algorithm

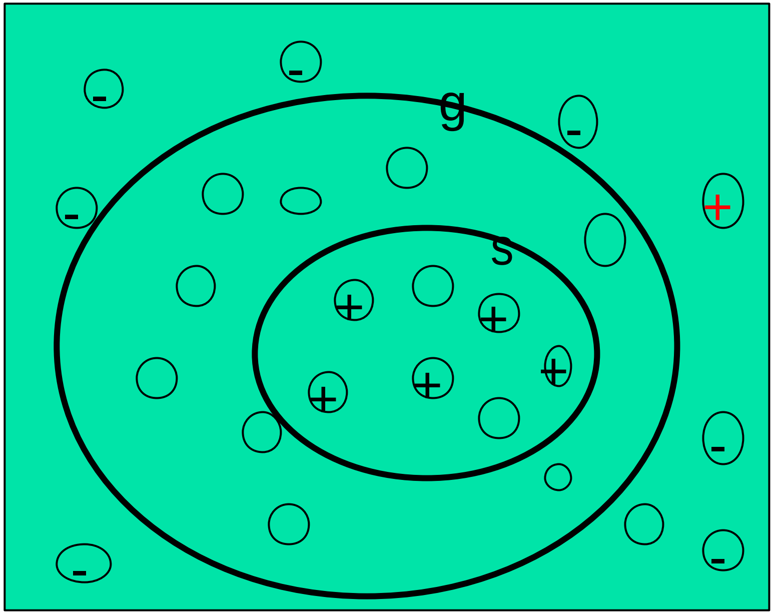
G ← maximally general hypotheses in H

S ← maximally specific hypotheses in H

For each training example d=<x,c(x)>

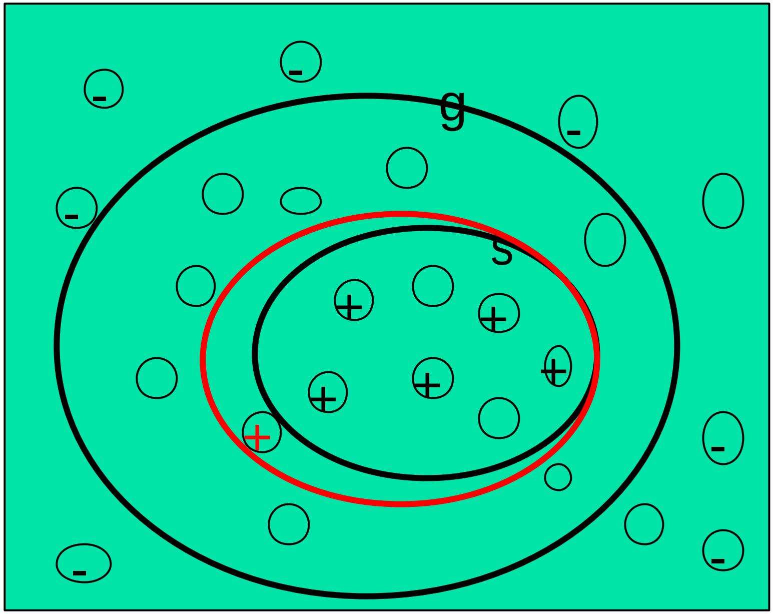
  modify G and S so that G and S are consistent with d

Positive Example: g(d)=s(d)=0



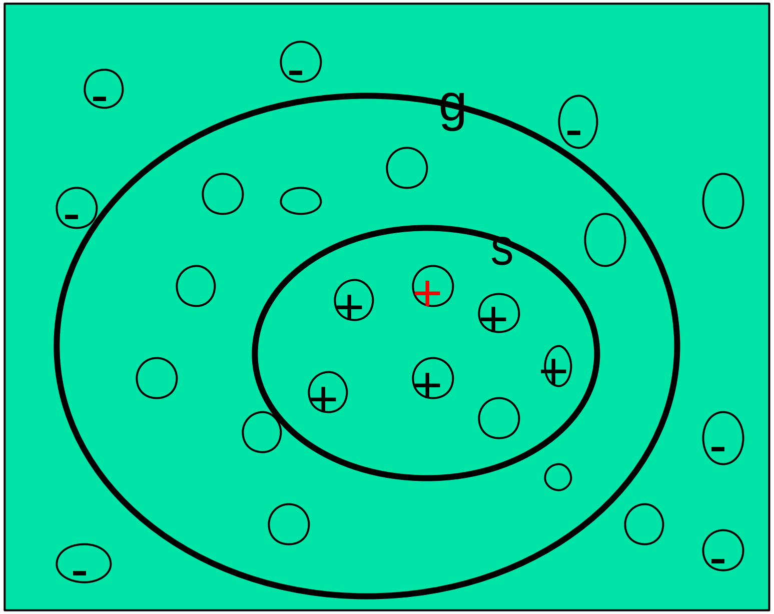
- remove g
- remove s

Possitive Example: g(d)=1 and s(d)=0

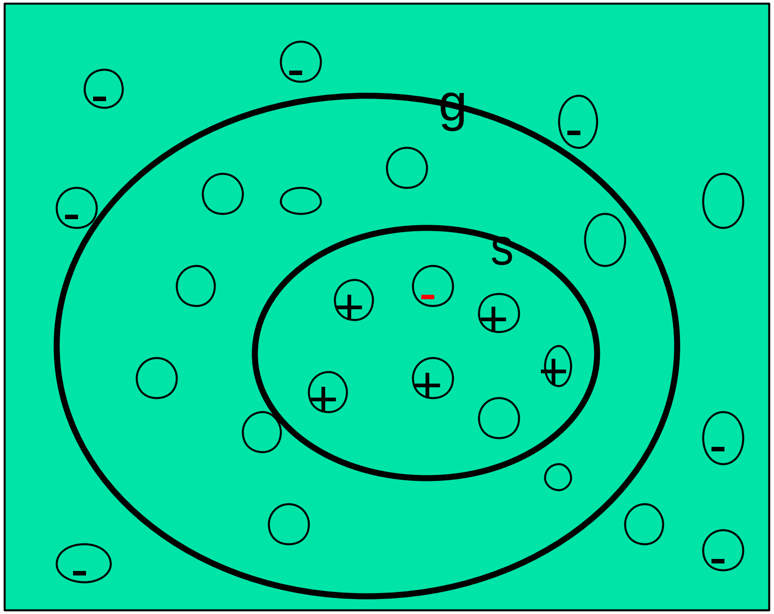


•generalize s

Possitive Example: g(d)=s(d)=1

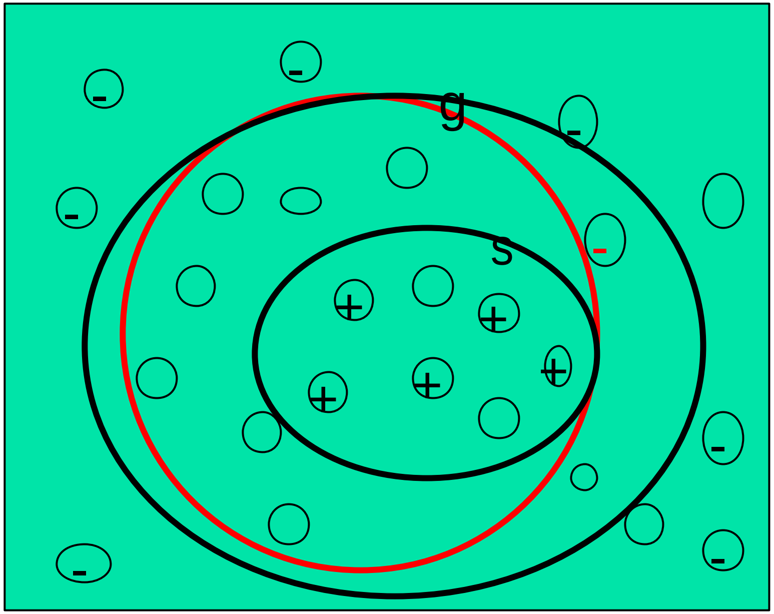


Negative Example: $g(d^-)=s(d^-)=1$



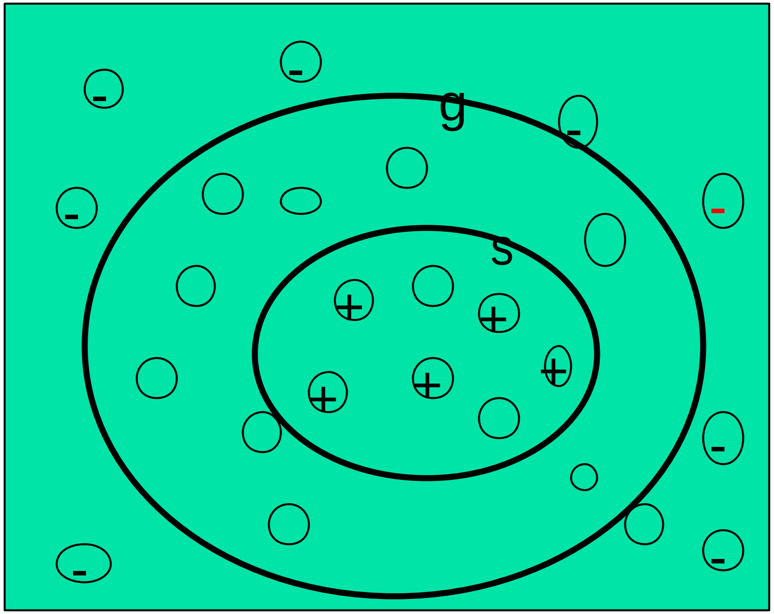
- remove s
- remove g

Negative Example: $g(d^-)=1$ and $s(d^-)=0$



- specialize g

Negative Example: $g(d^-)=s(d^-)=0$



Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H

S $\leftarrow$ maximally specific hypotheses in H

For each training example d=<x,c(x)>

If d is a positive example

Remove from G any hypothesis that is inconsistent with d

For each hypothesis s in S that is not consistent with d

- remove s from S.
- Add to S all minimal generalizations h of s such that
  - h consistent with d
  - Some member of G is more general than h
- Remove from S any hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm

If d is a negative example

Remove from S any hypothesis that is inconsistent with d

For each hypothesis g in G that is not consistent with d

- remove g from G.
- Add to G all minimal specializations h of g such that
  - h consistent with d
  - Some member of S is more specific than h
- Remove from G any hypothesis that is less general than another hypothesis in G

Example Candidate Elimination

S: ~~{<∅, ∅, ∅, ∅, ∅, ∅ >}~~

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

x_1 = <Sunny Warm Normal Strong Warm Same> +

S: ~~{< Sunny Warm Normal Strong Warm Same >}~~

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

x_2 = <Sunny Warm High Strong Warm Same> +

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

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

Example Candidate Elimination

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

G: ~~{<?, ?, ?, ?, ?, ?>}~~

x_3 = <Rainy Cold High Strong Warm Change> -

S: ~~{< Sunny Warm ? Strong Warm Same >}~~

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

x_4 = <Sunny Warm High Strong Cool Change> +

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

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

Remarks on Version Space and Candidate-Elimination

- converge to target concept when
  - no error in training examples
  - target concept is in H
- converge to an empty version space when
  - inconsistency in training data
  - target concept cannot be described by hypothesis representation
- what should be the next training example?
- how to classify new instances?

Classification of New Data

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

<Sunny,?,?,Strong,?,?> <Sunny,Warm,?,?,?,?> <?,Warm,?,Strong,?,?>

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

x_5 = <Sunny Warm Normal Strong Cool Change> + 6/0

x_6 = <Rainy Cold Normal Light Warm Same> - 0/6

x_7 = <Sunny Warm Normal Light Warm Same> ? 3/3

x_8 = <Sunny Cold Normal Strong Warm Same> ? 2/4

Inductive Leap

+ <Sunny Warm Normal Strong Cool Change>
+ <Sunny Warm Normal Light Warm Same>

S : <Sunny Warm Normal ? ? ?>

- How can we justify to classify the new example as

  + <Sunny Warm Normal Strong Warm Same>

Bias: We assume that the hypothesis space H contains the target concept c. In other words that c can be described by a conjunction of attribute constraints.

Biased Hypothesis Space

- Our hypothesis space is unable to represent a simple disjunctive target concept : (Sky=Sunny) v (Sky=Cloudy)

problem of expressibility

x_1 = <Sunny Warm Normal Strong Cool Change> +
x_2 = <Cloudy Warm Normal Strong Cool Change> +

S : { <?, Warm, Normal, Strong, Cool, Change> }

x_3 = <Rainy Warm Normal Light Warm Same> -

S : {}

Unbiased Learner

- Idea: Choose H that expresses every teachable concept, that means H is the set of all possible subsets of X called the power set P(X)
- $|X|=96$, $|P(X)|=2^{96} \sim 10^{28}$ distinct concepts
- H = disjunctions, conjunctions, negations
  - e.g. <Sunny Warm Normal ? ? ?> v <? ? ? ? ? Change>
- H surely contains the target concept.

Unbiased Learner

What are S and G in this case?

Assume positive examples (x_1, x_2, x_3) and negative examples (x_4, x_5)

S : { ($x_1 \vee x_2 \vee x_3$) }   G : { $\neg$ ($x_4 \vee x_5$) }

The only examples that are classified are the training examples themselves. In other words in order to learn the target concept one would have to present every single instance in X as a training example.

Each unobserved instance will be classified positive by precisely half the hypothesis in VS and negative by the other half.   problem of generalizability

Futility of Bias-Free Learning

- A learner that makes no prior assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

No Free Lunch!

Inductive Bias

Consider:

- Concept learning algorithm L
- Instances X, target concept c
- Training examples $D_c=\{<x,c(x)>\}$
- Let $L(x_i,D_c)$ denote the classification assigned to instance x_i by L after training on D_c.

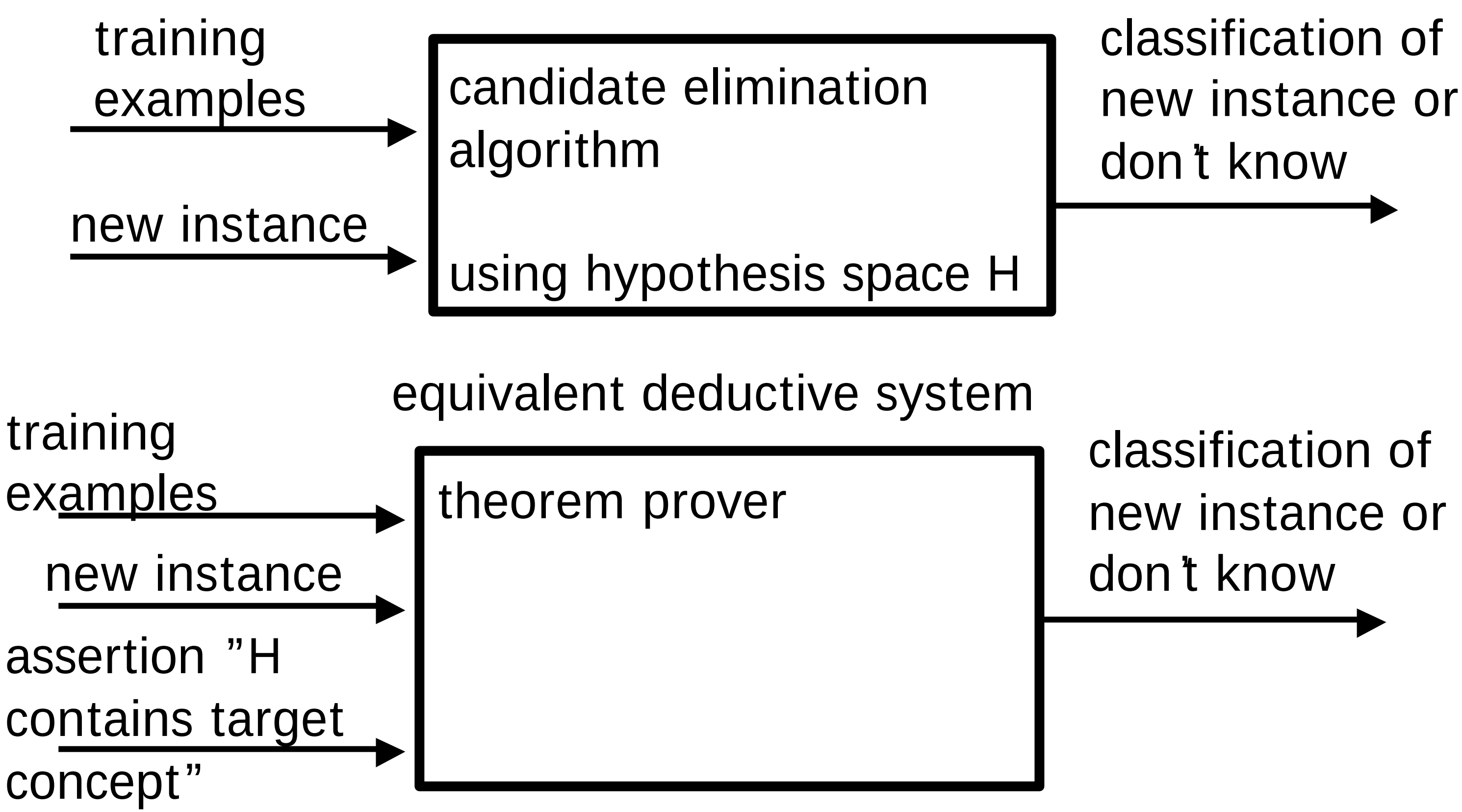
Definition:

The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training data D_c

$$(\forall x_i \in X)[B \wedge D_c \wedge x_i] \mid\!\text{--}\; L(x_i, D_c)$$

Where A |-- B means that A logically entails B.

Inductive Systems and Equivalent Deductive Systems



Three Learners with Different Biases

- Rote learner: Store examples, and classify x if and only if it matches a previously observed example.
  - No inductive bias
- Version space candidate elimination algorithm.
  - Bias: The hypothesis space contains the target concept.
- Find-S
  - Bias: The hypothesis space contains the target concept and all instances are negative instances unless the opposite is entailed by its other knowledge.

Summary

- Concept learning as search through H
- General-to-specific ordering over H
- Version space candidate elimination algorithm
- S and G boundaries characterize learner's uncertainty
- Learner can generate useful queries
- Inductive leaps possible only if learner is biased
- Inductive learners can be modelled by equivalent deductive systems