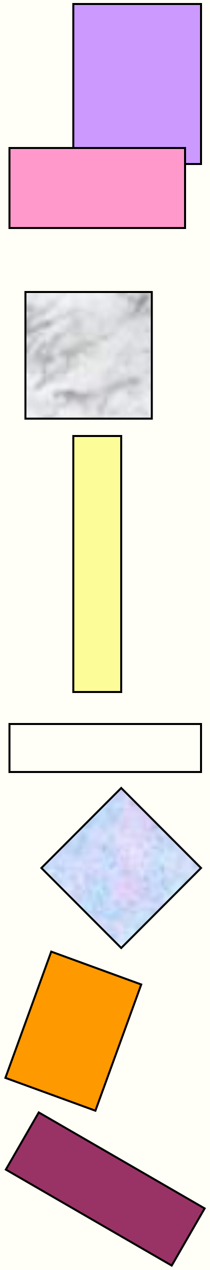
Machine Learning
Lecture2: Concept Learning and General-to specific ordering

Jie Li

Janice@163.com

Concept Learning

Analyzing Concepts



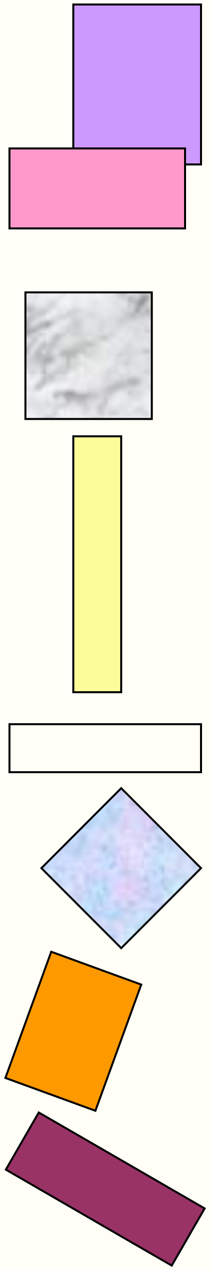
Concepts are categories of stimuli that have certain features in common.

The shapes on the left are all members of a conceptual category: **rectangle**.
Their common features are
(1) 4 lines;
(2) opposite lines parallel;
(3) lines connected at ends;
(4) lines form 4 right angles.

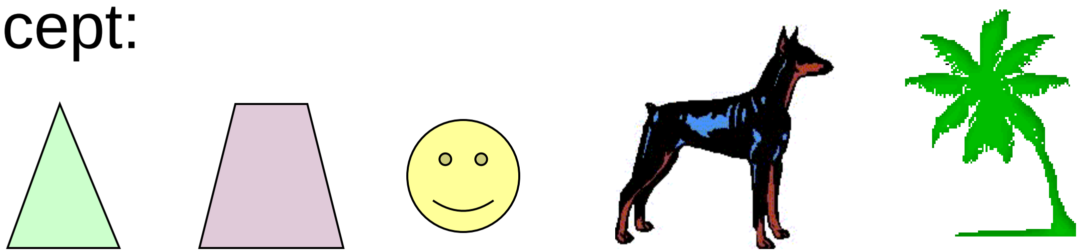
Color, size, and orientation are not defining features (irrelevant)of the concept

Concept Learning

Analyzing Concepts



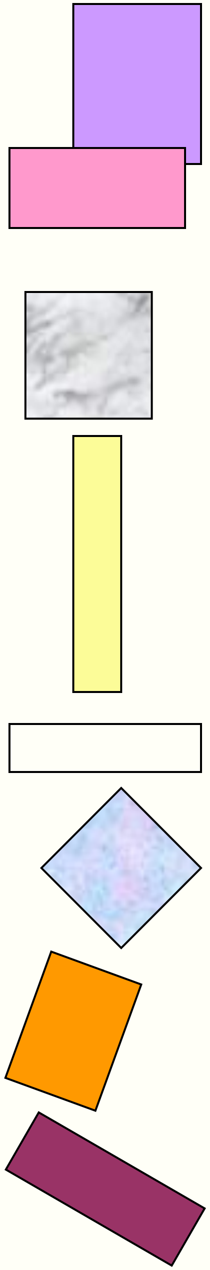
If a stimulus is a member of a specified conceptual category, it is referred to as a "positive instance". If it is not a member, it is referred to as "negative instance". These are all negative instances of the rectangle concept:



As rectangles are defined, a stimulus is a negative instance if it lacks any one of the specified features.

Concept Learning

Analyzing Concepts



Every concept has two components:

Attributes: These are features of a stimulus that one must look for to decide if that stimulus is a positive instance of the concept.

A rule: This a statement that specifies which attributes must be present or absent for a stimulus to qualify as a positive instance of the concept.

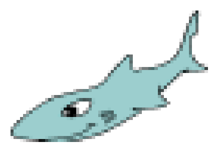
For rectangles, the attributes would be the four features discussed earlier, and the rule would be that all the attributes must be present.

Concept Learning

Analyzing Concepts

The simplest rules refer to the presence or absence of a single attribute. For example, a “vertebrate” animal is defined as an animal with a backbone. Which of these stimuli are positive instances?

+



+

–

+

Concept Learning

Analyzing Concepts

More complex conceptual rules involve two or more specified attributes. For example, the conjunction rule states that a stimulus must possess two or more specified attributes to qualify as a positive instance of the concept.

This was the rule used earlier to define the concept of a rectangle.

Concept Learning

Rote Learning vs. Concept Learning

Rote learning is learning without understanding the meaning of what is learned. For example, you can learn to make the correct response to a stimulus without discovering the conceptual category to which the stimulus belongs.

The next time you see that example, you may give the correct term. But what if you are given an example you haven't seen before?

Concept Learning

Rote Learning vs. Concept Learning

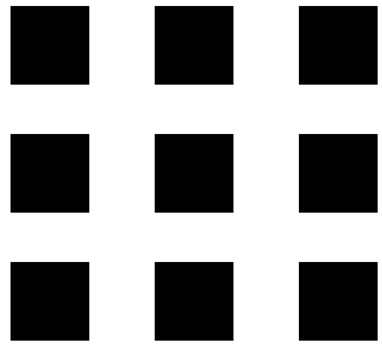
This is the real test of concept learning: Can you recognize (respond appropriately to) novel instances of the concept?

An operant conditioning experiment with chimpanzees (Kelleher, 1958) illustrates the distinction between rote learning and concept learning.

The procedure was a form of discrimination training in which there were 13 discriminative stimuli (+ instances of a concept) and 13 delta stimuli (- negative instances of the concept).

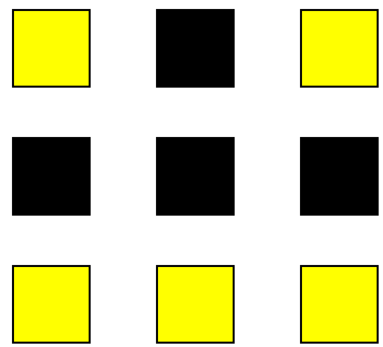
Concept Learning

Each stimulus involved 9 small windows arranged in 3 rows, with 3 windows per row:



Each window could be lit or left dark, for example:

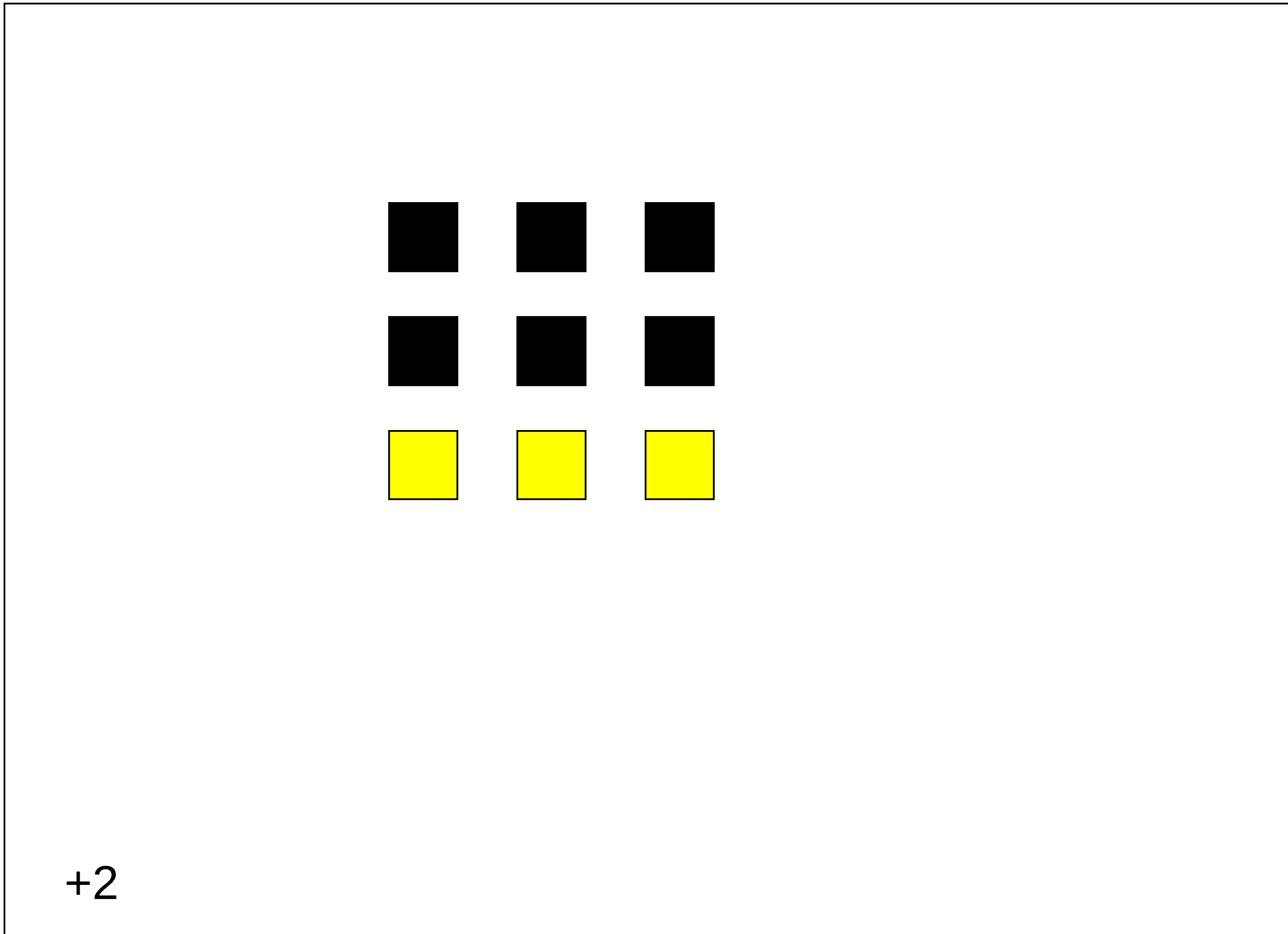
Concept Learning



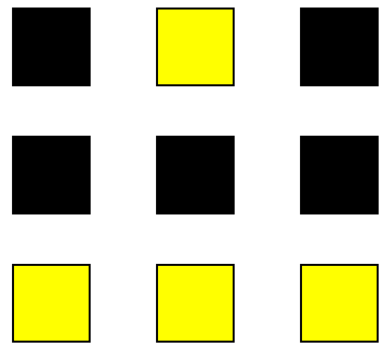
The array, above was one of the + instances of the concept. Here are two more:

Concept Learning

Rote Learning vs. Concept Learning



Concept Learning



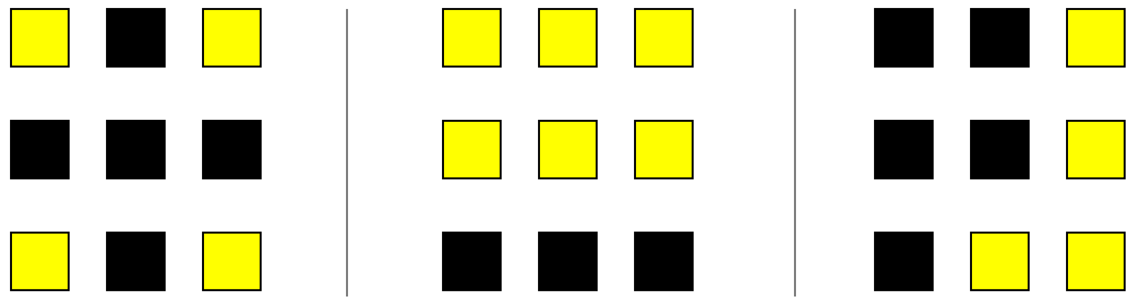
The positive instances all had common attributes. The negative instances lacked one or more of these attributes. Here are some of the negative stimuli:

+3

Concept Learning

Rote Learning vs.
Concept Learning

Negative Instances (Delta Stimuli)

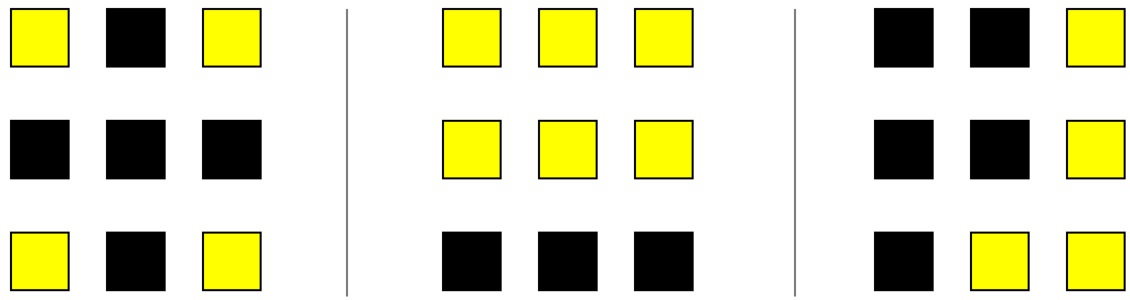


The common attributes of the positive instances defined the concept. It was...?

Concept Learning

Rote Learning vs. Concept Learning

Negative Instances (Delta Stimuli)



The common attributes of the positive instances defined the concept. It was...?

Bottom three windows lit.

Concept Learning

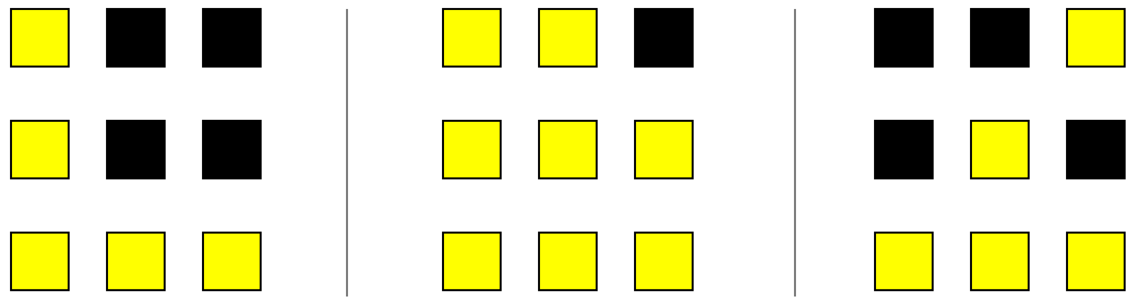
In Phase 2 of this discrimination problem, 6 positive and 6 negative stimuli were removed and replaced with new ones *while keeping the concept the same*.

Here are examples: three of the new positive instances and three of the new negative instances.

Concept Learning

Rote Learning vs. Concept Learning

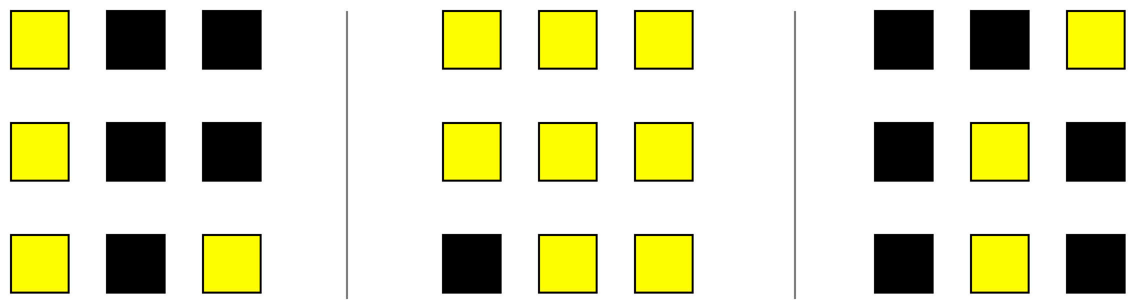
New Positive Instances (Phase 2)



Concept Learning

Rote Learning vs. Concept Learning

New Negative Instances (Phase 2)

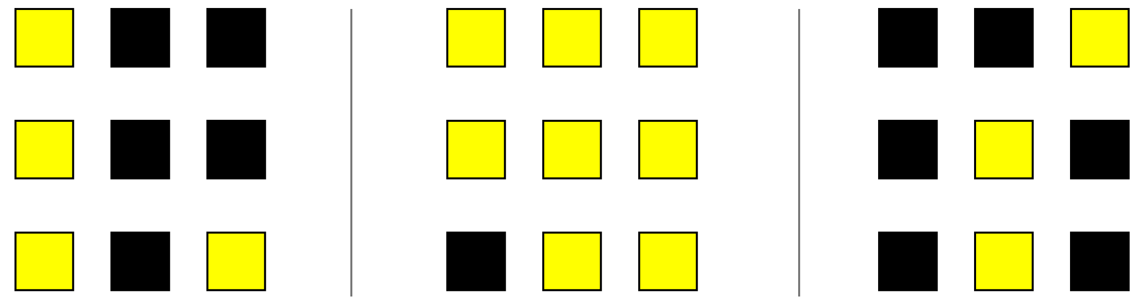


?

The question was: Would the chimps respond appropriately to the new stimuli even though they never saw those exact patterns before?

Concept Learning

New Negative Instances (Phase 2)



Result: Yes, they showed rapid responding to the + instances and little or no responding to the negative instances.

Concept Learning

Rote Learning vs. Concept Learning

This showed that the chimps had learned a concept in Phase 1. Whatever the pattern was, they looked at the bottom 3 windows, and if all 3 were lit, they pressed the button.

This illustrates the advantage of solving problems conceptually: You can respond appropriately to new situations. You focus on just the relevant attributes.

Concept Learning

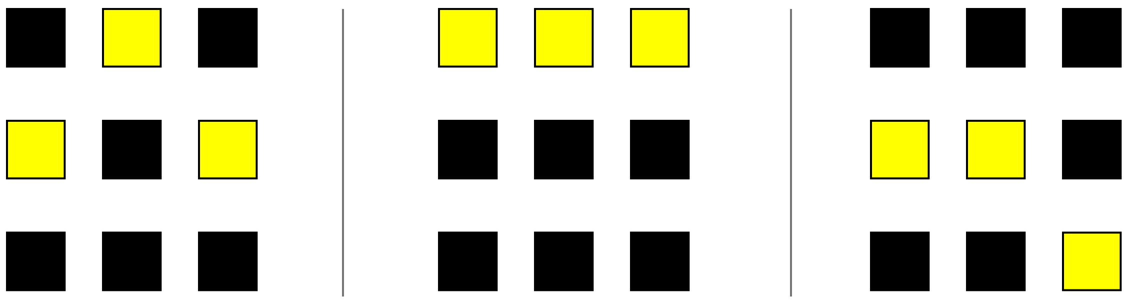
Example of Rote Learning

The chimps were given a second concept problem involving 13 new + instances and 13 new – instances. Here is a sample of these stimuli. What would you say the concept was?

Concept Learning

Rote Learning vs. Concept Learning

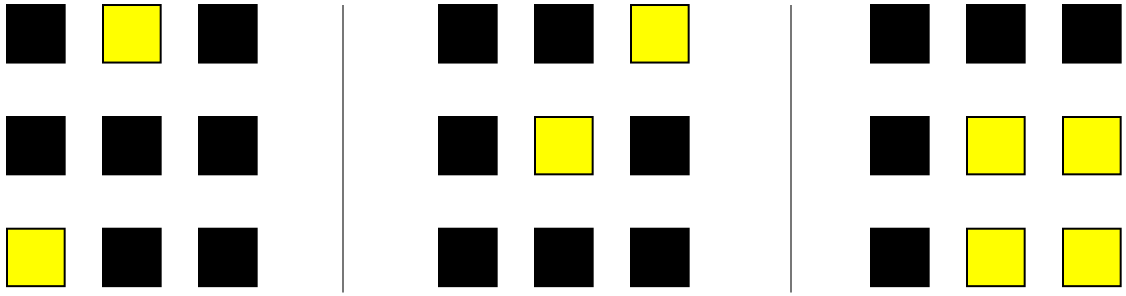
New Positive Instances (Problem 2)



Concept Learning

Rote Learning vs. Concept Learning

New Negative Instances (Problem 2)



Concept Learning

Example of Rote Learning

The concept was "any 3 windows lit." Negative instances had 2 or 4 windows lit.

The procedure was the same as in Problem 1. The 13 + and 13 – instances were presented until a strong discrimination was learned: rapid responding during + stimuli, little or no responding during – stimuli.

Concept Learning

Example of Rote Learning

Then came the test of concept learning: 6 new + instances and 6 new – instances were presented. Would the animals keep responding appropriately?

Concept Learning

Rote Learning vs.
Concept Learning

Example of Rote Learning

Performance was disrupted during the new stimuli, with frequent pauses during positive stimuli and bursts of rapid responding during negative stimuli.

The chimps solved the problem by "memorizing the answers" without discovering what the positive stimuli had in common. This may be because the concept was abstract: The 3 lit windows were not tied to a specific location. The animals had to respond to the number "3".

Concept learning

Concept learning

=

inferring a boolean-valued function from training examples of its input and output

Training Examples for Concept Enjoy Sport

Concept: "days on which my friend Aldo 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

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

instance

Representing Hypotheses

- Hypothesis h is 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:

- Instances X : Possible days described by the attributes *Sky, Temp, Humidity, Wind, Water, Forecast*
- Target function c: EnjoySport $X \rightarrow \{0,1\}$
- Hypotheses 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.

Counting Instances, Concepts, Hypotheses

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

#distinct instances : ?

#syntactically distinct hypotheses : ?

#semantically distinct hypotheses : ?

Counting 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

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

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

General to Specific Ordering

- Consider two hypotheses:
  - h_1 = <Sunny, ?, ?, Strong, ?, ?>
  - h_2 = <Sunny, ?, ?, ?, ?, ?>

- Set of instances covered by h_1 and h_2:

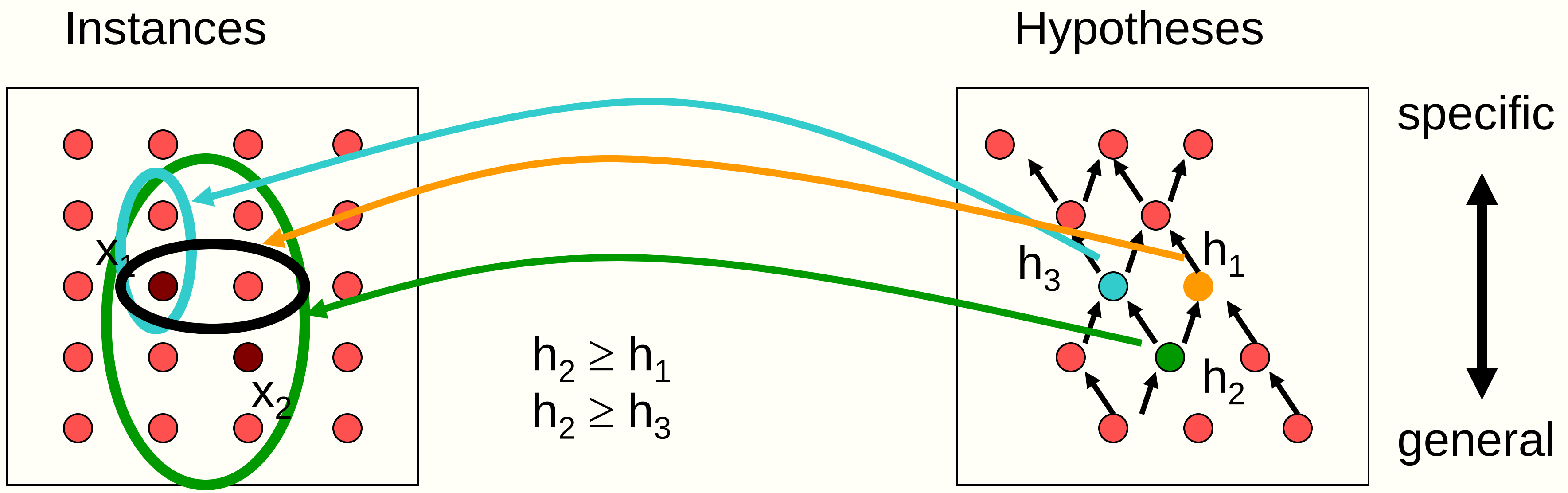
  h_2 imposes fewer constraints than h_1 and therefore classifies more instances x as positive, i.e. $h(x) = 1$

Def. 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 by many concept learning methods

General to Specific Ordering



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 Algorithm

Initialize h to the most specific hypothesis in H

For each positive training instance x

  For each attribute constraint a_i in h

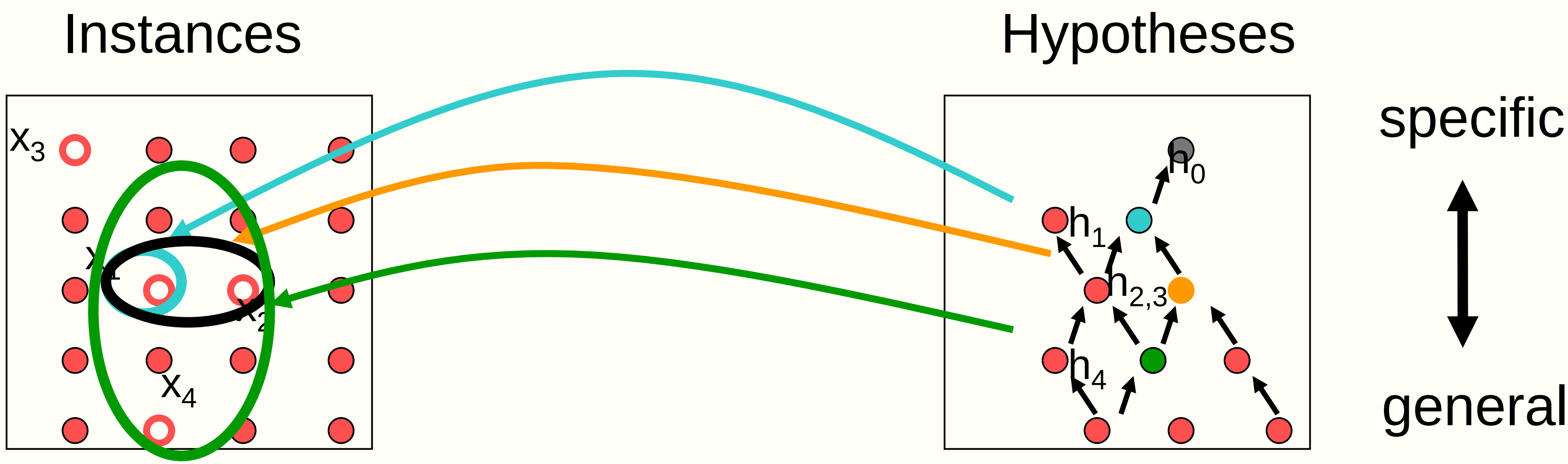
    If the constraint a_i in h is satisfied by x

      then do nothing

      else replace a_i in h by the next more general constraint that is satisfied by x

Output hypothesis h

Hypothesis Space Search by 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.

Complaints about Find-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
- Can't tell when training data is inconsistent, as it ignores negative training examples
- Why prefer the most specific hypothesis?
- What if there are multiple maximally specific hypothesis?

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.

  $$Consistent(h,D) := \forall <x,c(x)> \in D \;\; h(x)=c(x)$$

- 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 h that is inconsistent with the training example, i.e.

   $h(x) \neq c(x)$

3. Output the list of hypotheses in *VersionSpace*

Version Space: example

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 Space

- The **General boundary** *G*, of version space $VS_{H,D}$ is the set of its maximally general members

$$G \equiv \{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

- The **Specific boundary** *S*, of version space $VS_{H,D}$ is the set of its maximally specific members

$$S \equiv \{s \in H \mid Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

- Every member of the version space lies between these boundaries

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

  – Version Space Representation Theorem

Candidate Elimination Algorithm (1/3)

- $G \leftarrow$ maximally general hypotheses in H

$$G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\}$$

Should be specialized

- $S \leftarrow$ maximally specific hypotheses in H

$$S_0 \leftarrow \{\langle \phi, \phi, \phi, \phi, \phi, \phi \rangle\}$$

Should be generalized

Candidate Elimination Algorithm (2/3)

- For each training example *d*, do
  - If *d* is a positive example
    - Remove from *G* any hypothesis 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* is consistent with *d*, and
        - » some member of *G* is more general than *h*
      - Remove from *S* any hypothesis that is more general than another hypothesis in *S*
        (i.e., partial-ordering relations exist)

positive training examples force the S boundary become increasing general

Candidate Elimination Algorithm (3/3)

- If *d* is a negative example
  - Remove from *S* any hypothesis 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* is consistent with *d*, and
      - » some member of S is more specific than *h*
    - Remove from *G* any hypothesis that is less general than another hypothesis in *G*
      (i.e., partial-ordering relations exist)

negative training examples force the G boundary become increasing specific

Example Trace (1/5)

S_0: $\{\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle\}$

G_0: $\{\langle ?, ?, ?, ?, ?, ? \rangle\}$

Example Trace (2/5)

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

↓

S_1: { <*Sunny, Warm, Normal, Strong, Warm, Same*> }

↓

S_2: { <*Sunny, Warm, ?, Strong, Warm, Same*> }

G_0, G_1, G_2: { <?, ?, ?, ?, ?, ?> }

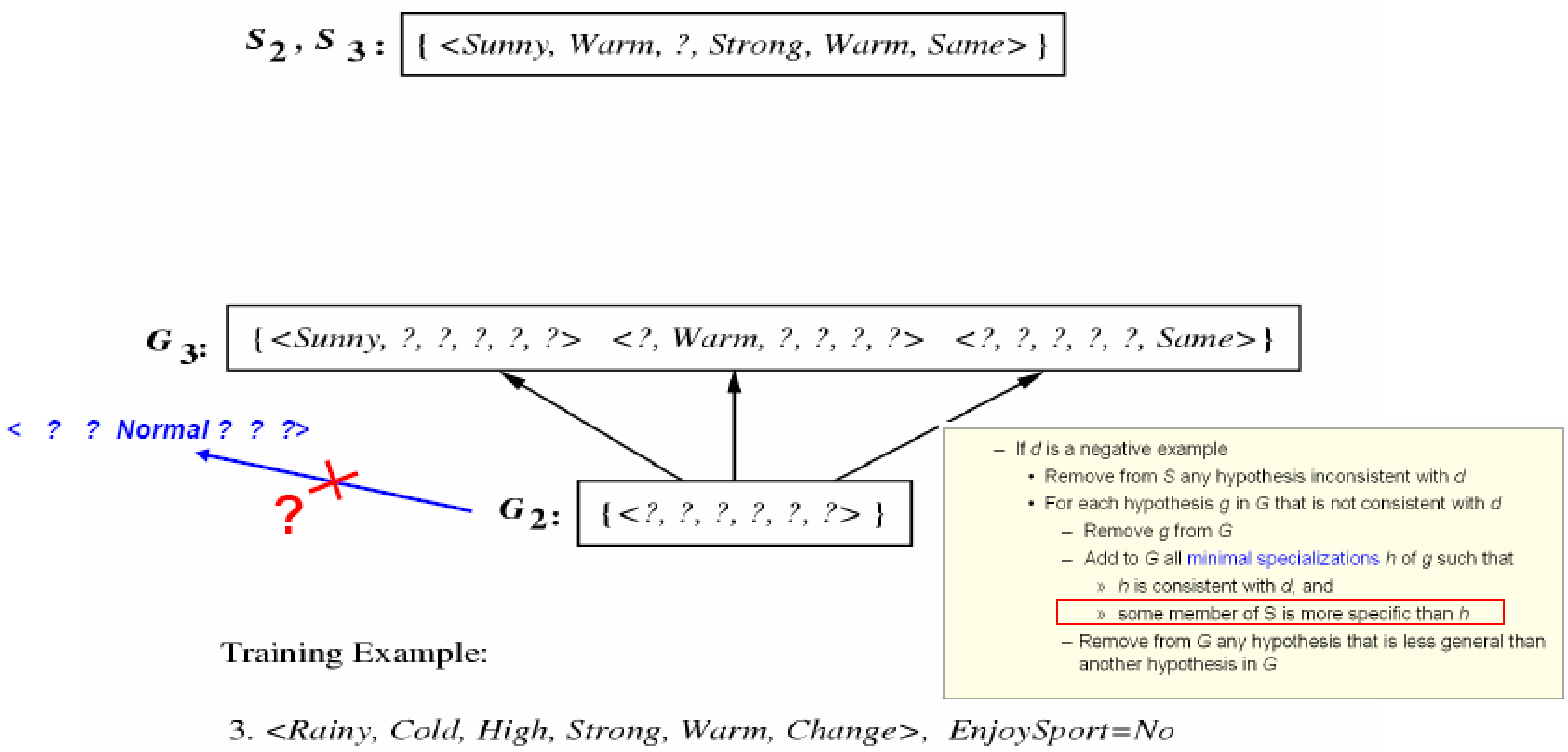
- For each training example *d*, do
  - If *d* is a positive example
    - Remove from *G* any hypothesis inconsistent with *d*
    - For each hypothesis *s* in *S* that is not consistent with *d*
      - Remove *s* from *S*
      - Add to *S* all minimal generalizations *h* of *s* such that
        - » *h* is consistent with *d*, and
        - » some member of *G* is more general than *h*
      - Remove from *S* any hypothesis that is more general than another hypothesis in *S* (i.e., partial-ordering relations exist)

Training examples:

1 . <*Sunny, Warm, Normal, Strong, Warm, Same*>, *Enjoy Sport = Yes*

2 . <*Sunny, Warm, High, Strong, Warm, Same*>, *Enjoy Sport = Yes*

Example Trace (3/5)



- G_2 has six ways to be minimally specified
  - Why <?,? ,*Normal*,?,?,? > etc. do not exist in G_3 ?

Example Trace (4/5)

S 3: {<*Sunny, Warm, ?, Strong, Warm, Same*>}

S 4: {<*Sunny, Warm, ?, Strong, ?, ?*>}

- For each training example *d*, do
  - If *d* is a positive example
    - Remove from *G* any hypothesis 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* is consistent with *d*, and
        - some member of *G* is more general than *h*
      - Remove from *S* any hypothesis that is more general than another hypothesis in *S* (i.e., partial-ordering relations exist)

G 4: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*>}

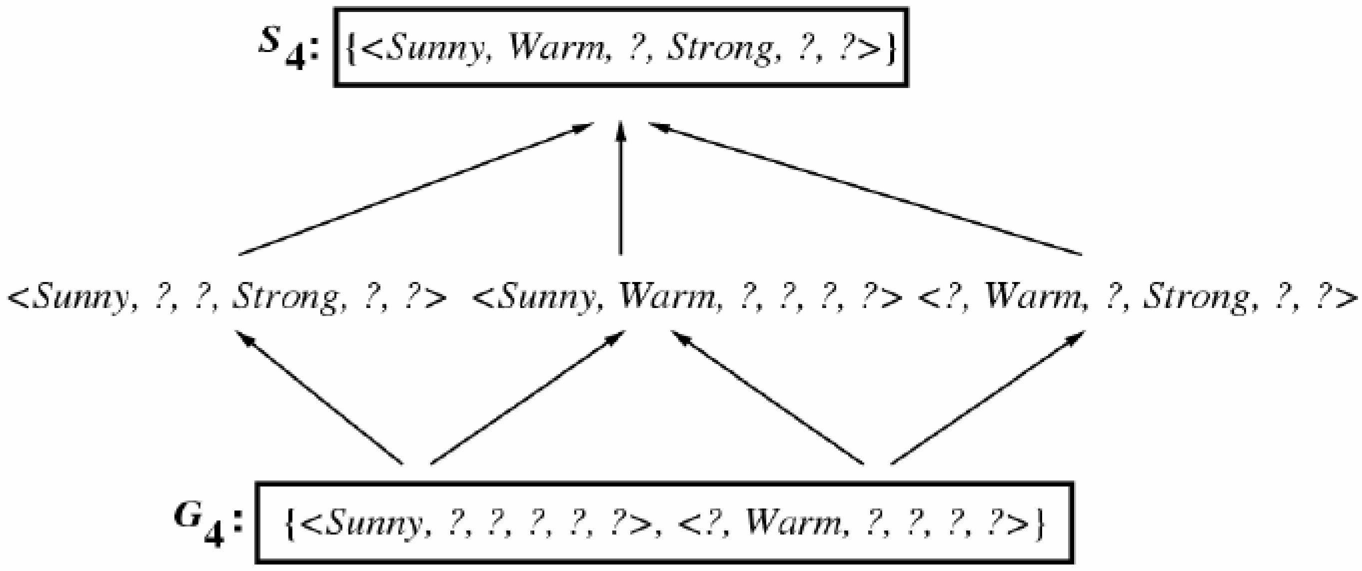
G 3: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*> <*?, ?, ?, ?, ?, Same*>}

Training Example:

4. <*Sunny, Warm, High, Strong, Cool, Change*>, *EnjoySport = Yes*

- Notice that,
  - *S* is a summary of the previously positive examples
  - *G* is a summary of the previously negative examples

Example Trace (5/5)

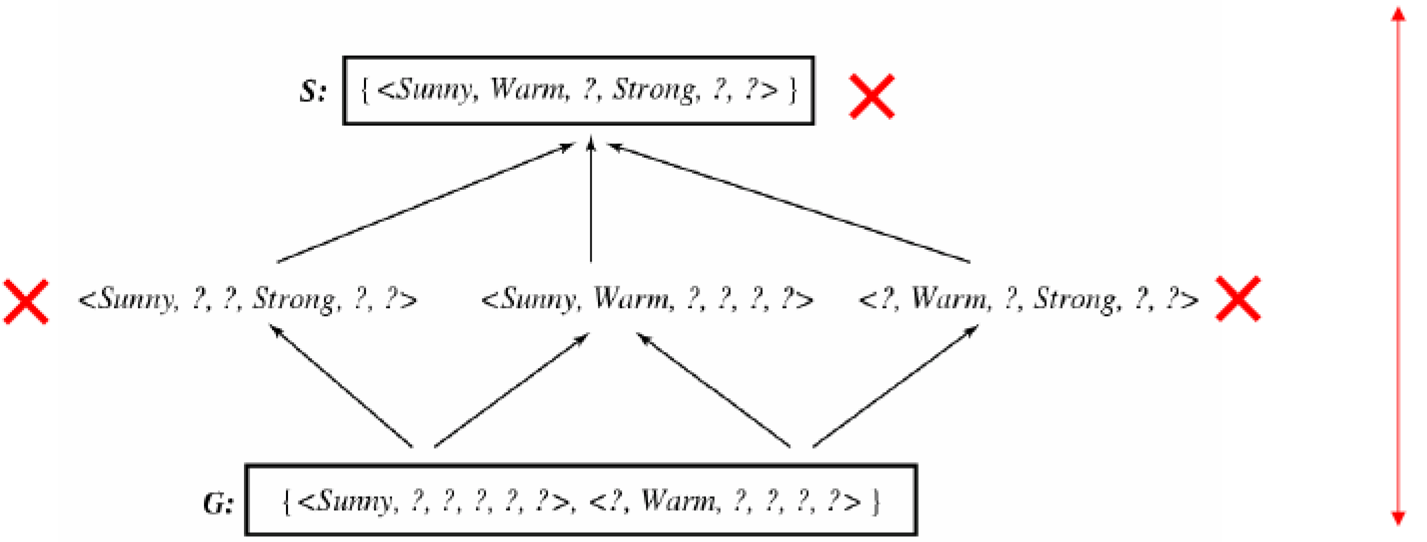


- *S* and *G* boundaries move monotonically closer to each other, delimiting a smaller and smaller version space

What Next Training Example

- Learner can generate useful queries
  - Discriminate among the alternatives competing hypotheses in the current version space

Specific



General

If a positive hypothesis is posed:

<Sunny, Warm, Normal, Light, Warm, Same >

What if it is a negative one ?

Version Space: example

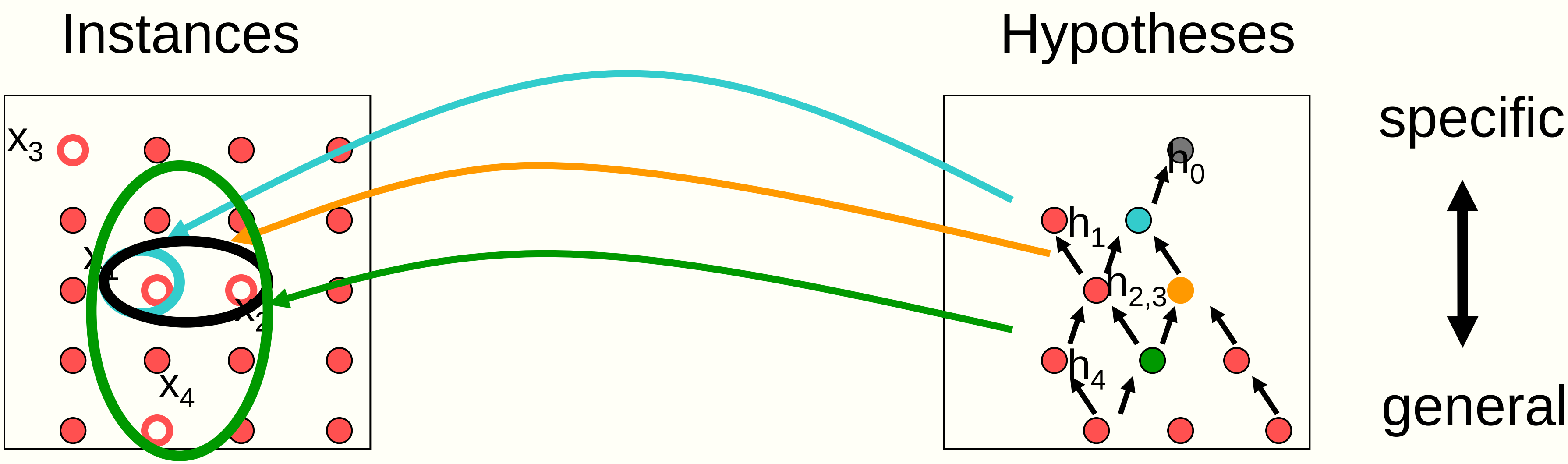
S: {<Sunny,Warm,?,Strong,?,?>}

<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> +

Hypothesis Space Search by 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> +

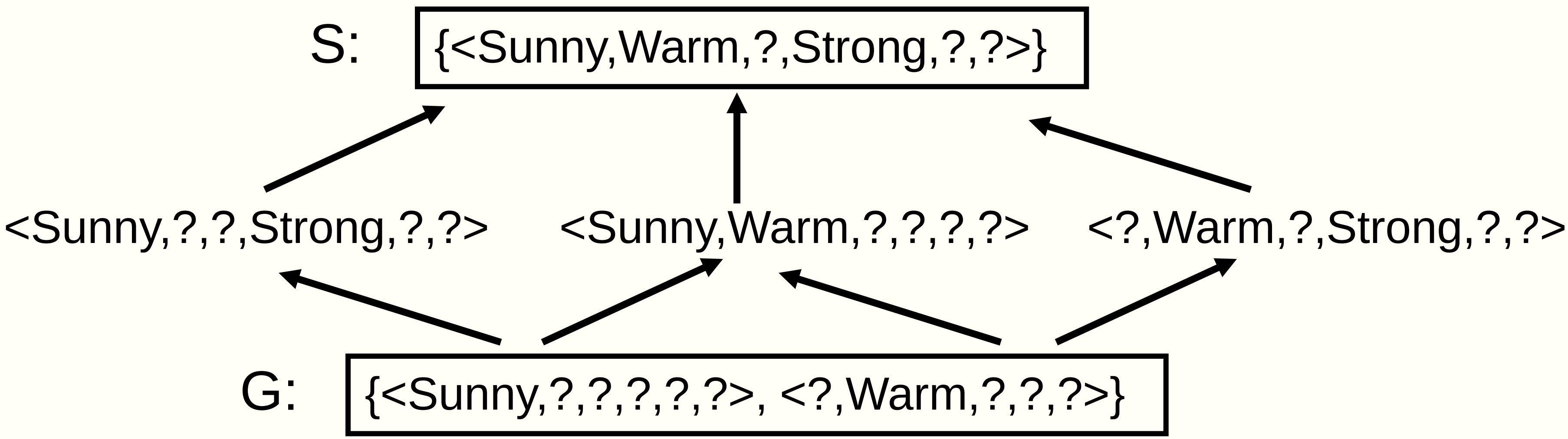
h_0=<Ø, Ø, Ø, Ø, Ø, Ø,>

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

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

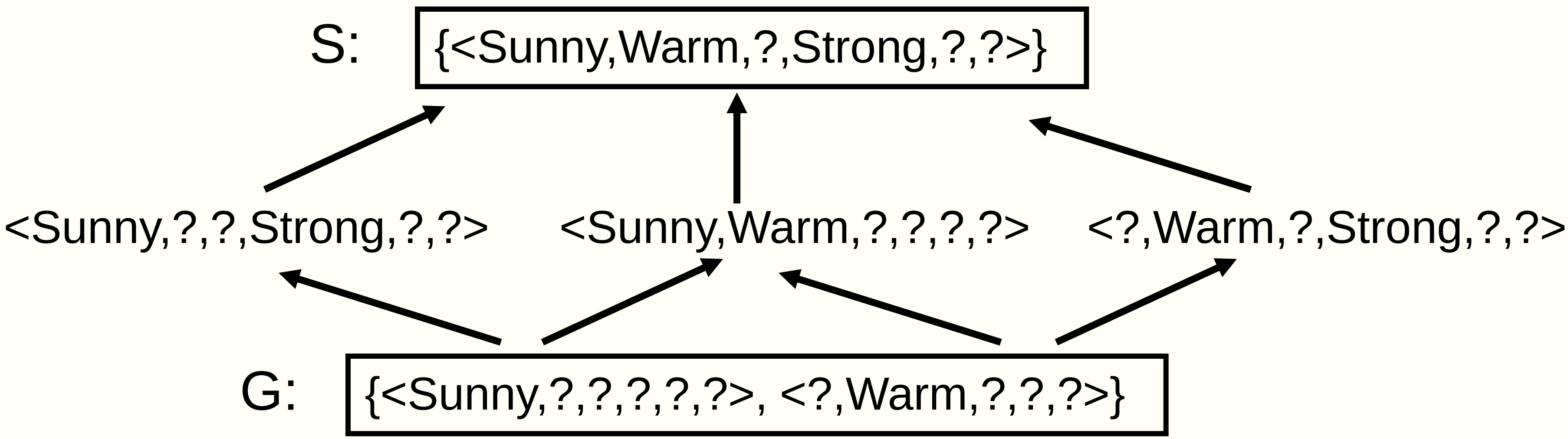
h_4=<Sunny,Warm,?,
Strong,?,?>

Classification of New Data



x_5 = <Sunny Warm Normal Strong Cool Change>
x_6 = <Rainy Cold Normal Light Warm Same>
x_7 = <Sunny Warm Normal Light Warm Same>
x_8 = <Sunny Cold Normal Strong Warm Same>

Classification of New Data



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 ?

Biased Hypothesis Space

- Our hypothesis space is unable to represent a simple disjunctive target concept :

(Sky=Sunny) v (Sky=Cloudy)

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 Strong, Cool, Change > -

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

Counting 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

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

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

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.

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

$\boxed{\text{No Free Lunch!}}$

Inductive Bias

Consider:

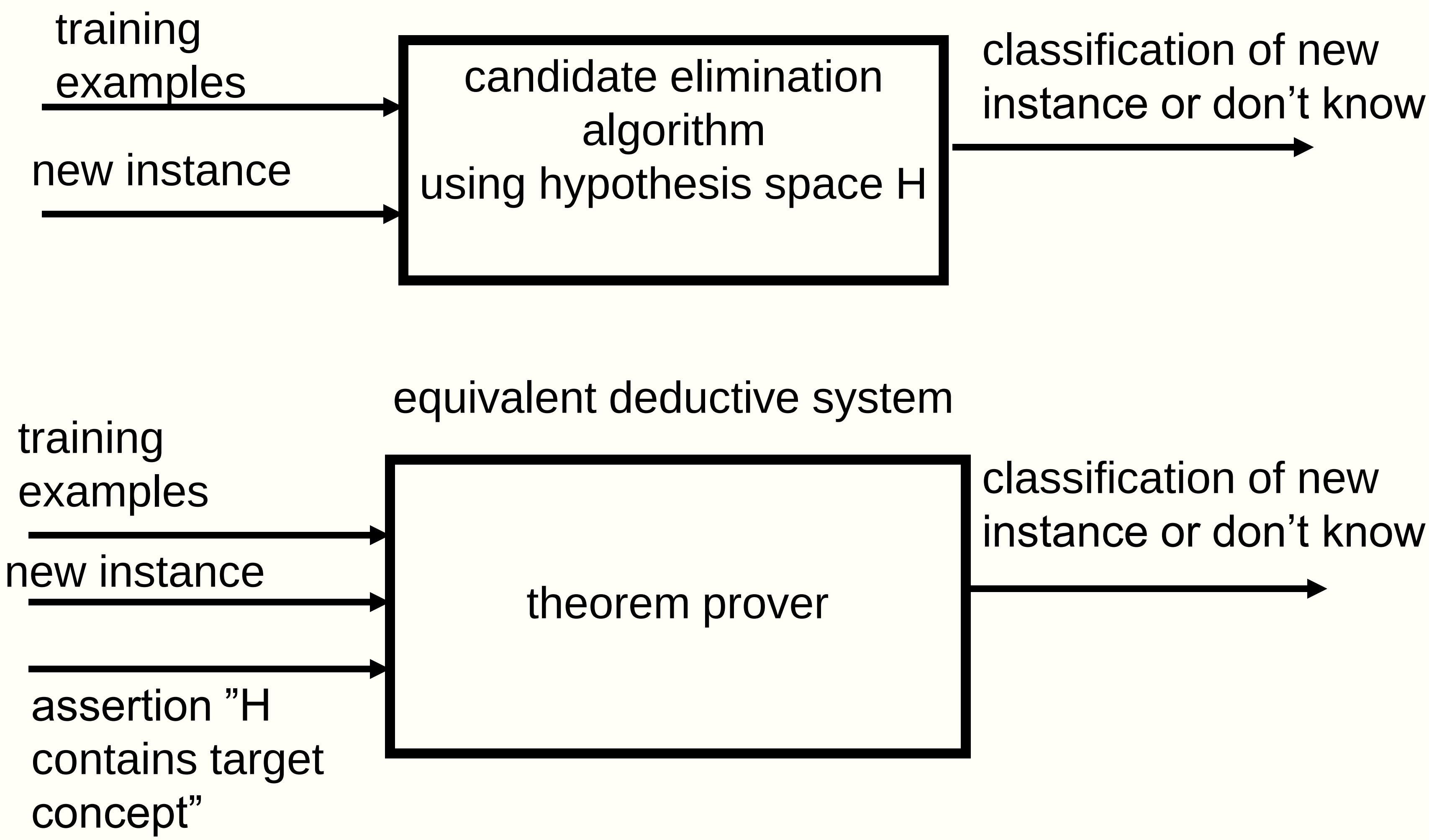
- Concept learning algorithm L
- Instances X, target concept c
- Training examples $D_c = \{<x, c(x)>\}$
- Let $L(x_i, D_c)$ denote the classification assigned to instance x_i by L after training on D_c.

Def. 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 classify x if and only if it matches a previously observed example.
  –
- Version space candidate elimination algorithm.
  –
- Find-S
  –

Three Learners with Different Biases

- Rote learner: Store examples 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.