Chapter ML:II (continued)

II. Machine Learning Basics

- Rule-Based Learning of Simple Concepts
- From Regression to Classification
- Evaluating Effectiveness

Rule-Based Learning of Simple Concepts

Classification Problem

Setting:

- X is a multiset of feature vectors.
- $C = \{\text{no}, \text{yes}\}$ is a set of two classes.
  Similarly: $\{0, 1\}$, $\{-1, 1\}$, $\{\ominus, \oplus\}$, "belongs to a concept or not", etc.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a decision rule $y()$ for feature value combinations.

Rule-Based Learning of Simple Concepts

Example Learning Task

X contains vectors encoding the weather in the six dimensions “Sky”, “Temperature”, “Humidity”, “Wind”, “Water”, and “Forecast”.

D contains examples of weather conditions $\mathbf{x} \in X$ along with a statement whether or not our friend will enjoy her favorite sport (surfing):

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing	
1	sunny	warm	normal	strong	warm	same	yes	1
2	sunny	warm	high	strong	warm	same	yes	1
3	rainy	cold	high	strong	warm	change	no	0
4	sunny	warm	high	strong	cool	change	yes	1

What is the *extensional* definition of the concept (class) “EnjoySurfing”?

→ What is an *intensional* definition of the concept (class) “EnjoySurfing”?

→ What are *hypotheses* that capture the concept (class) “EnjoySurfing”?

Remarks:

- Domains of the features in the learning task:

Sky	Temperature	Humidity	Wind	Water	Forecast
sunny	warm	normal	strong	warm	same
rainy	cold	high	light	cool	change
cloudy					

Rule-Based Learning of Simple Concepts

Concepts and Hypotheses

Definition 1 (Concept, Hypothesis, Hypothesis Space)

Let O be a set of objects, $\mathbf{X}$ the feature space constructed with a model formation function $\alpha : O \to \mathbf{X}$, and $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors.

A concept is a subset of O and hence induces a subset $X' \subseteq X$. Concept learning means to learn from an example set D a function $y()$, $y : X \to \{0, 1\}$, which returns 1 if $\mathbf{x} \in X'$ and 0 otherwise.

Rule-Based Learning of Simple Concepts

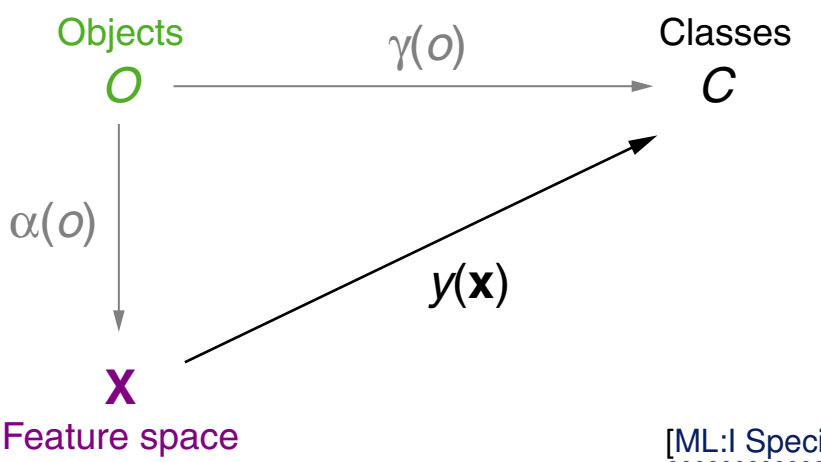
Concepts and Hypotheses (continued)

Definition 1 (Concept, Hypothesis, Hypothesis Space)

Let O be a set of objects, $\mathbf{X}$ the feature space constructed with a model formation function $\alpha : O \to \mathbf{X}$, and $X = \{\mathbf{x} \mid \mathbf{x} = \alpha(o), o \in O\}$ be a multiset of feature vectors.

A concept is a subset of O and hence induces a subset $X' \subseteq X$. Concept learning means to learn from an example set D a function $y()$, $y : X \to \{0, 1\}$, which returns 1 if $\mathbf{x} \in X'$ and 0 otherwise.

A concrete function $y()$ is called hypothesis. The set H of all considered hypotheses (model functions) is called hypothesis space.



[ML:I Specification of Learning Problems]

Remarks (concept) :

- A concept (a class) can be described by an intensional or an extensional definition, which gives meaning to the concept (the class). [Wikipedia]

  An intensional definition gives meaning by specifying necessary and sufficient conditions for the concept. An extensional definition gives meaning to a concept by specifying its extension, that is, every object that falls under the definition of the concept in question.

- Within a learning task, a concept is (usually incompletely) defined by means of D, which contains examples from which a subset falls under the definition of the concept in question. I.e., D gives us an incomplete, extensional definition of the concept.

  In the exemplary learning task the population is comprised of the combinations of possible weather conditions in the six dimensions (see the feature domains above). The concept is the subset of the population that contains exactly those weather conditions when surfing is enjoyed.

Remarks (hypothesis) :

- Recap. A hypothesis is a proposed explanation for a phenomenon. [Wikipedia]
  Here, a hypothesis "explains" (= fits) the data D. Hence, a concrete model function $y()$, $\mathbf{y}()$, or, if the function type is clear from the context, its parameters $\mathbf{w}$ or $\boldsymbol{\theta}$ are called "hypothesis". The variable name h (similarly: h_1, h_2, h_i, h', etc.) may be used to refer to a specific instance of a model function or its parameters.
- A hypothesis is expected to "capture a (target) concept", to "explain a (target) concept", or to "predict a (target) concept" in terms of the feature expressions of the objects.
- The "quality", the "persuasiveness", or the "power" of a hypothesis depends on its capability to represent (= to fit) a given set of observations, which are called examples here.
- In our learning setting, a hypothesis cannot be inferred or proven by deductive reasoning. Rather, a hypothesis is a finding or an insight gained by *inductive reasoning*.

Rule-Based Learning of Simple Concepts

Consistent Hypotheses

The example set D, $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\}$, contains usually both positive ($c = 1$) and negative ($c = 0$) examples. [learning task]

Definition 2 (Classified as Positive, Consistent)

Let $y()$ be a concrete model function (hypothesis).

An example $(\mathbf{x}, c)$ is classified as positive by $y()$ iff $y(\mathbf{x}) = 1$.

$y()$ is consistent with an example $(\mathbf{x}, c)$ iff $y(\mathbf{x}) = c$.

$y()$ is consistent with a set D of examples, denoted as *consistent*$(y(), D)$, iff:

$$\forall (\mathbf{x}, c) \in D : \; y(\mathbf{x}) = c$$

Remarks:

- The string “Iff” or “iff” is an abbreviation for “If and only if”, which means “necessary and sufficient”. It is a textual representation for the logical biconditional, also known as material biconditional or iff-connective. The respective symbol is “$\leftrightarrow$”. [Wolfram] [Wikipedia]
- An example that is classified as positive by a hypothesis can also be said to fulfill the hypothesis.
- The fact that a hypothesis is consistent with an example can also be described the other way round: an example is consistent with a hypothesis.
- Given an example $(\mathbf{x}, c)$, notice the difference between (1) classified as positive and (2) being consistent with a hypothesis. The former asks for $y(\mathbf{x}) = 1$, disregarding the actual target concept value c. The latter asks for the identity between the target concept c and $y(\mathbf{x})$.
- The consistency of $y(\mathbf{x})$ can be analyzed for a single example as well as for a set D of examples. Given the latter, consistency requires that $y(\mathbf{x}) = 1$ iff $c = 1$, for all $(\mathbf{x}, c) \in D$. This is equivalent with the condition that $y(\mathbf{x}) = 0$ iff $c = 0$, for all $(\mathbf{x}, c) \in D$.
- Learning means to determine a model function $y() \in H$ that is consistent with D.
  Similarly: Machine learning means to systematically search the hypothesis space.

Rule-Based Learning of Simple Concepts

Decision Rules as Model Functions

Structure of our decision rule $y()$:

- a propositional logic conjunction of feature value matches
- three kinds of parameter values θ : literal, *?* (wildcard), $\perp$ (contradiction)

$$y(\mathbf{x}) \;=\; \mathcal{I}(\theta_1 \doteq x_1 \;\wedge\; \theta_2 \doteq x_2 \;\wedge\; \ldots \;\wedge\; \theta_6 \doteq x_6)$$

$\mathcal{I}(\theta \doteq x)$	θ	x
1	$\text{literal}_\theta = \text{literal}_x$	
0	$\text{literal}_\theta \neq \text{literal}_x$	
1	*?*	literal_x
0	$\perp$	literal_x

Rule-Based Learning of Simple Concepts

Decision Rules as Model Functions

Structure of our decision rule $y()$:

- a propositional logic conjunction of feature value matches
- three kinds of parameter values θ : literal, *?* (wildcard), $\perp$ (contradiction)

$$y(\mathbf{x}) \;=\; \mathcal{I}(\theta_1 \doteq x_1 \;\wedge\; \theta_2 \doteq x_2 \;\wedge\; \ldots \;\wedge\; \theta_6 \doteq x_6)$$

$\mathcal{I}(\theta \doteq x)$	θ	x
1	$\text{literal}_\theta = \text{literal}_x$	
0	$\text{literal}_\theta \neq \text{literal}_x$	
1	*?*	literal_x
0	$\perp$	literal_x

A hypothesis h for EnjoySurfing [learning task] : $\boldsymbol{\theta} = (\textit{sunny, warm, ?, strong, ?, ?}\ )$

$$\rightsquigarrow\ \forall(\mathbf{x}, c) \in D :\ y(\mathbf{x}) = c$$

Remarks:

- The binary relation »$\doteq$« corresponds to the $=$-relation between strings plus the two extra cases for »*?*« (always true) and »$\perp$« (never true).
- $\mathcal{I}$ denotes the standard interpretation function, which returns for a propositional formula α its truth value: $\alpha \mapsto \mathcal{I}(\alpha),\ \mathcal{I}(\alpha) \in \{0, 1\}$.
- Depending on the learning task—more specifically: on the structure of the feature space—a hypothesis (the model function, the model) can take different forms. Examples:
  - rules (as done here)
  - analytical functions (typical for regression settings)
  - decision trees (typical for categorical domains)
  - probability mass functions (in Bayesian learning)

Rule-Based Learning of Simple Concepts

Extremal Hypotheses

Definition 3 (Maximally Specific / General Hypothesis)

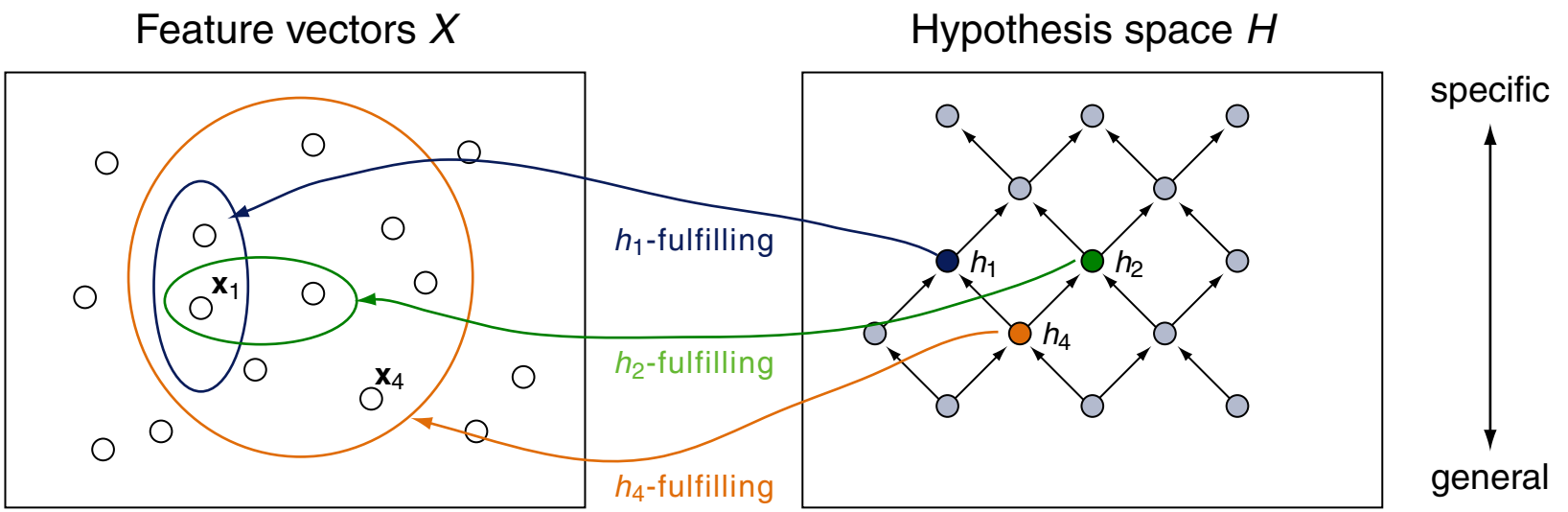
The model functions $y_{s_0}() \equiv 0$ and $y_{g_0}() \equiv 1$ are called maximally specific and maximally general hypothesis respectively. No $\mathbf{x} \in X$ is classified as positive by $y_{s_0}()$, and all $\mathbf{x} \in X$ are classified as positive by $y_{g_0}()$.

Parameters of maximally specific / general hypothesis in the example [learning task] :

- $h_{s_0} : \boldsymbol{\theta} = (\bot, \bot, \bot, \bot, \bot, \bot)$ (never enjoy sport)
- $h_{g_0} : \boldsymbol{\theta} = (\,?, ?, ?, ?, ?, ?\,)$ (always enjoy sport)

Rule-Based Learning of Simple Concepts

Order of Hypotheses



$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$

$h_1 : y_1()$ with $\boldsymbol{\theta} = (\textit{sunny, ?, normal, ?, ?, ?})$

$h_2 : y_2()$ with $\boldsymbol{\theta} = (\textit{sunny, ?, ?, ?, warm, ?})$

$h_4 : y_4()$ with $\boldsymbol{\theta} = (\textit{sunny, ?, ?, ?, ?, ?})$

Rule-Based Learning of Simple Concepts

Order of Hypotheses (continued)

Definition 4 (More General Relation)

Let X be a multiset of feature vectors and let $y_1()$ and $y_2()$ be two boolean-valued model functions with domain X. Then $y_1()$ is called more general than $y_2()$, denoted as $y_1() \geq_g y_2()$, iff:

$$\forall \mathbf{x} \in X : \ (\ y_2(\mathbf{x}) = 1 \ \text{ implies } \ y_1(\mathbf{x}) = 1 \ )$$

$y_1()$ is called strictly more general than $y_2()$, denoted as $y_1() >_g y_2()$, iff:

$$y_1() \geq_g y_2() \ \text{ and } \ y_2() \not\geq_g y_1()$$

In the illustration:

$\forall \mathbf{x} \in X : y_2(\mathbf{x}) = 1$ implies that $y_4(\mathbf{x}) = 1$. I.e., $y_4()$ is more general than $y_1()$.

Remarks:

- If $y_1()$ is more general than $y_2()$, then $y_2()$ can also be called being more specific than $y_1()$.
- The relations $\geq_g$ and $>_g$ are independent of a target concept. They depend only on the fact that examples are classified as positive by a hypothesis, i.e., whether $y(\mathbf{x}) = 1$, $(\mathbf{x}, c) \in D$. It is not required that $c = 1$.
- h_{s_0} is minimum and h_{g_0} is maximum with regard to $\geq_g$: no hypothesis is more specific than h_{s_0}, and no hypothesis is more general than h_{g_0}.

  We will consider only hypothesis spaces that contain h_{s_0} and h_{g_0}.
- The $\geq_g$-relation defines a partial order on the hypothesis space H : $\geq_g$ is reflexive, anti-symmetric, and transitive. The order is *partial* since (unlike in a total order) not all hypothesis pairs stand in the relation. [Wikipedia partial, total]

  I.e., we are given hypotheses $y_i()$, $y_j()$, for which neither $y_i() \geq_g y_j()$ nor $y_j() \geq_g y_i()$ holds, such as the hypotheses h_1 and h_2 in the illustration.

Remarks (entailment) :

- The semantics of the implication, in words "a implies b", denoted as $a \to b$, is as follows. $a \to b$ is true if either (1) a is true and b is true, or (2) if a is false and b is true, or (3) if a is false and b is false—in short: "if a is true then b is true as well", or, "the truth of a implies the truth of b".
- "$\to$" can be understood as "causality connective": Let a and b be two events where a is a cause for b. If we interpret the occurrence of an event as true and its non-occurrence as false, we will observe only occurrence combinations such that the formula $a \to b$ is true. The connective is also known as material conditional, material implication, material consequence, or simply, implication or conditional.
- Note in particular that the connective "$\to$" does not mean "entails", which would be denoted as either $\Rightarrow$ or $\models$. Logical entailment (synonymously: logical inference, logical deduction, logical consequence) allows to infer or to prove a formula β given a formula α.

  Consider for instance the More-General-Definition: From the formula $\alpha =$ "$y_2(\mathbf{x}) = 1$" we cannot infer or prove the formula $\beta =$ "$y_1(\mathbf{x}) = 1$".
- In the More-General-Definition the implication specifies a condition that is to be fulfilled by the definiendum (= the thing to be defined). The implication is used to check whether or not a thing belongs to the set of things specified by the definiens (= the expression that defines):

  Each pair of functions, $y_1()$, $y_2()$, is a thing that belongs to the set of things specified by the definition of the $\geq_g$-relation (i.e., stands in the $\geq_g$-relation) if and only if the implication $y_2() = 1 \ \to \ y_1() = 1$ is true for all $\mathbf{x} \in X$.

Remarks (entailment) : (continued)

- In a nutshell: distinguish carefully between "α requires β", denoted as $\alpha \to \beta$, on the one hand, and "from α follows β", denoted as $\alpha \Rightarrow \beta$, on the other hand. $\alpha \to \beta$ is considered as a sentence from the *object language* (language of discourse) and stipulates a computing operation, whereas $\alpha \Rightarrow \beta$ is a sentence from the *meta language* and makes an assertion *about* the sentence $\alpha \to \beta$, namely: "$\alpha \to \beta$ is a tautology".
- Finally, consider the following sentences from the object language, which are synonymous:

  "$\alpha \to \beta$"
  "α implies β"
  "if α then β"
  "α causes β"
  "α requires β"
  "α is sufficient for β"
  "β is necessary for α"

Rule-Based Learning of Simple Concepts

Inductive Learning Hypothesis

"Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples."

[p.23, Mitchell 1997]

Rule-Based Learning of Simple Concepts

Find-S Algorithm [Mitchell 1997] [algorithms: LMS BGD$_\sigma$ PT BGD IGD]

Algorithm:	Find-S	Find Specific Hypothesis.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in X$ and target class $c \in \{0, 1\}$.
Output:	$\boldsymbol{\theta}$	Parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ from H. (= hypothesis)

Find-S(D)

```
1.  θ = h_{s_0}  // Start with the maximally specific hypothesis in H.
2.  FOREACH (x, c) ∈ D DO
3.    IF c = 1 THEN // Learn only from positive examples.
4.      y(x) = I(θ_1 ≐ x_1 ∧ ... ∧ θ_p ≐ x_p)
5.      IF y(x) = 0 DO
6.        θ = min_generalization(θ, x)
7.      ENDIF
8.    ENDIF
9.  ENDDO
10. return(θ)
```

Rule-Based Learning of Simple Concepts

Find-S Algorithm [Mitchell 1997] [algorithms: LMS BGD$_\sigma$ PT BGD IGD]

Algorithm:	Find-S	Find Specific Hypothesis.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in X$ and target class $c \in \{0, 1\}$.
Output:	$\boldsymbol{\theta}$	Parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ from H. (= hypothesis)

Find-S(D)

```
1.  θ = h_{s_0}  // Start with the maximally specific hypothesis in H.
2.  FOREACH (x, c) ∈ D DO
3.    IF c = 1 THEN // Learn only from positive examples.
4.      [          ]   Model function evaluation.
5.      [          ]   Check for consistency.
6.      [          ]   Parameter vector update ≙ least more general hypothesis.
7.    ENDIF
8.  ENDIF
9.  ENDDO
10. return(θ)
```

Remarks:

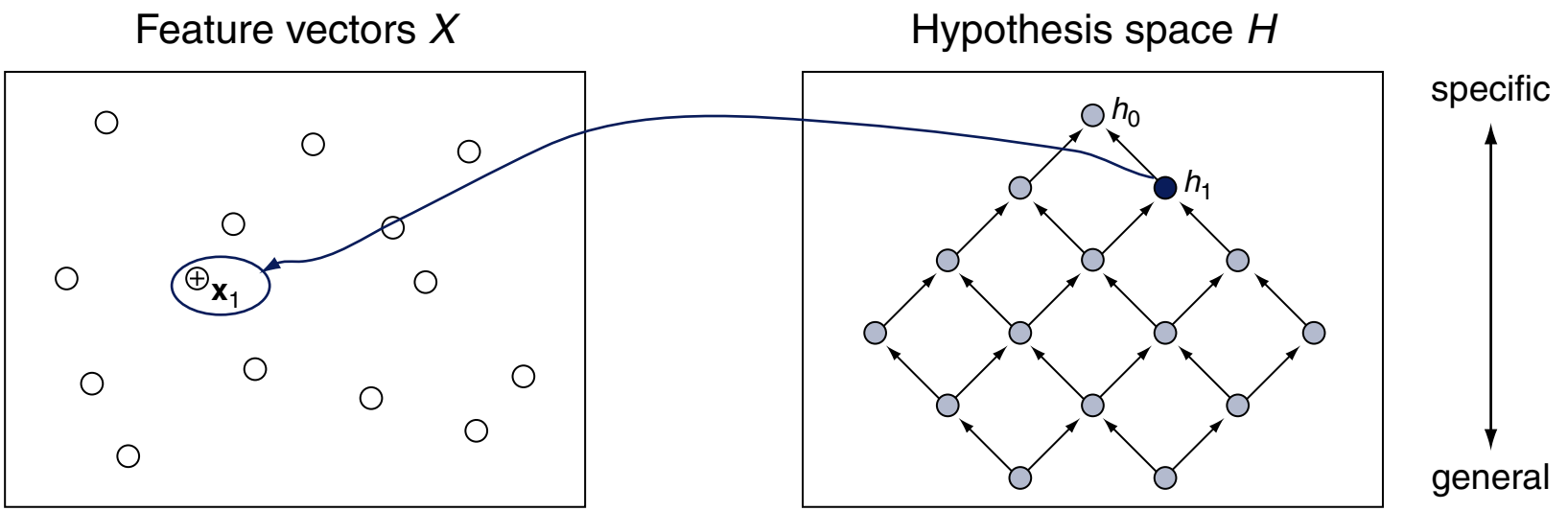
- Except for the first step, generalization means to substitute question marks (wildcards) for literals. Another term for “generalization” is “relaxation”.
- *min_generalization*$(\boldsymbol{\theta}, \mathbf{x})$ returns a parameter vector $\boldsymbol{\theta}'$ that is minimally generalized wrt. $\boldsymbol{\theta}$ and that is consistent with $(\mathbf{x}, 1)$. Formally:

  $y'() \geq_g y()$ and $y'(\mathbf{x}) = 1$, and there is no $y''()$ with $y'(\mathbf{x}) >_g y''() \geq_g y()$ with $y''(\mathbf{x}) = 1$.
- For more complex hypothesis structures the relaxation of $y()$, *min_generalization*$()$, may not be unique. In such a case one of the alternatives has to be chosen.
- If a function $y()$ needs to be relaxed towards some $y'()$ with $y'() \notin H$, the maximally general hypothesis h_{g_0} can be added to H.
- Similar to *min_generalization*$()$, a function *min_specialization*$()$ can be defined, which returns a minimally specialized, consistent hypotheses for negative examples.

Rule-Based Learning of Simple Concepts

Find-S Algorithm: Illustration

See the example set D for the concept *EnjoySurfing*.



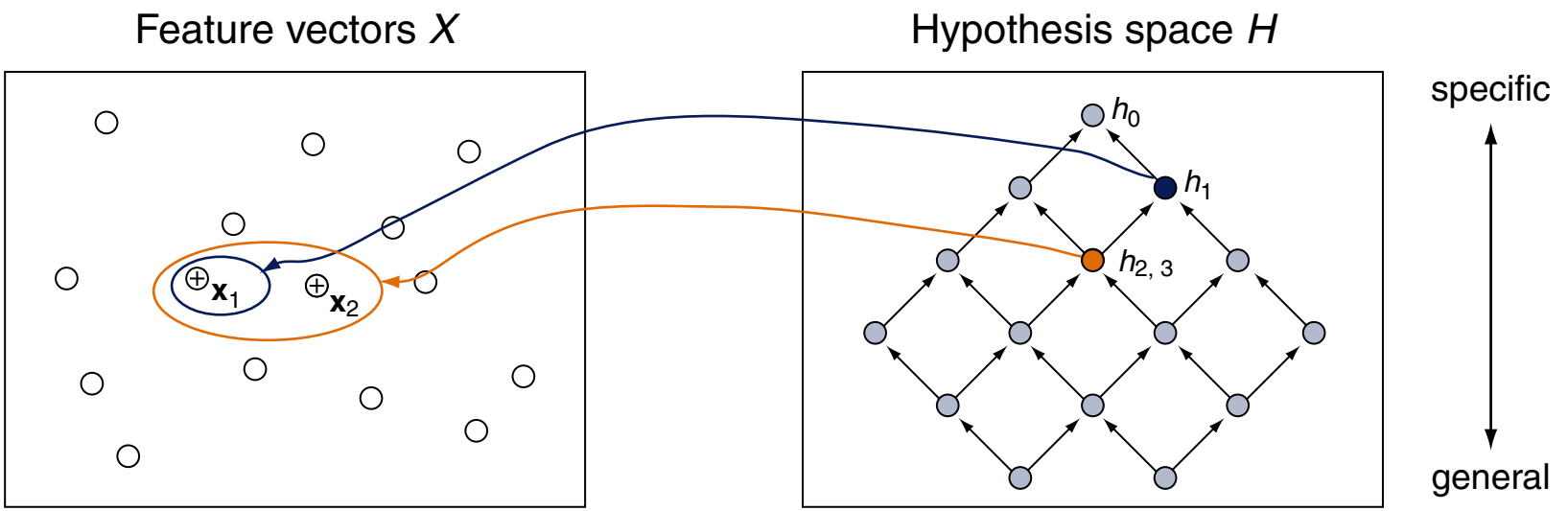
$h_0 = h_{s_0}$: $(\perp, \perp, \perp, \perp, \perp, \perp)$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $h_1 : (\textit{sunny, warm, normal, strong, warm, same})$

Rule-Based Learning of Simple Concepts

Find-S Algorithm: Illustration

See the example set D for the concept *EnjoySurfing*.



$h_0 = h_{s_0}$: $(\perp, \perp, \perp, \perp, \perp, \perp)$

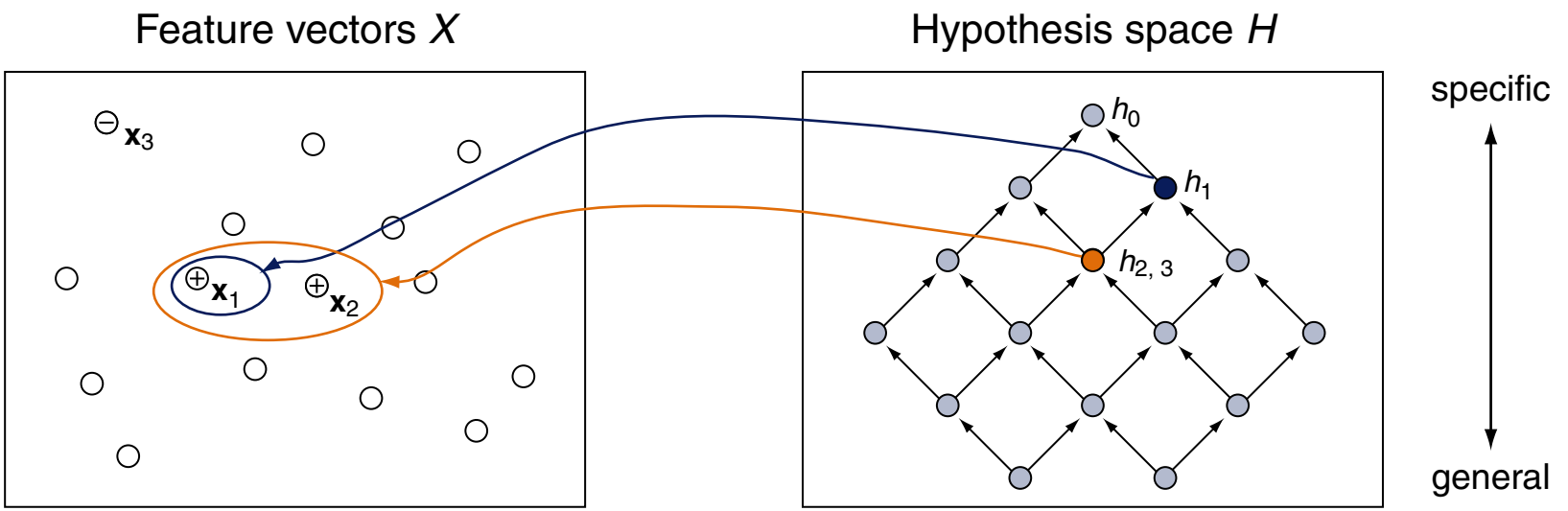
$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $h_1 : (\textit{sunny, warm, normal, strong, warm, same})$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $h_2 : (\textit{sunny, warm, ?, strong, warm, same})$

Rule-Based Learning of Simple Concepts

Find-S Algorithm: Illustration

See the example set D for the concept *EnjoySurfing*.



$h_0 = h_{s_0}$: $(\perp, \perp, \perp, \perp, \perp, \perp)$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $h_1 : (\textit{sunny, warm, normal, strong, warm, same})$

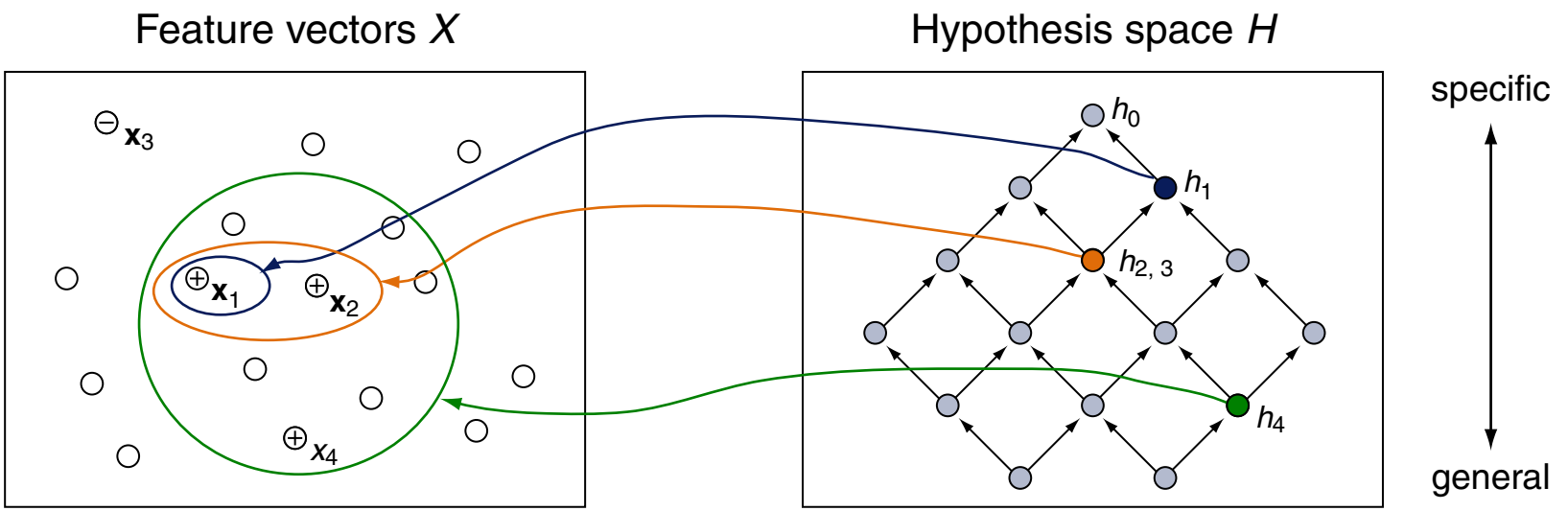
$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $h_2 : (\textit{sunny, warm, ?, strong, warm, same})$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $h_3 : (\textit{sunny, warm, ?, strong, warm, same})$

Rule-Based Learning of Simple Concepts

Find-S Algorithm: Illustration

See the example set D for the concept *EnjoySurfing*.



$h_0 = h_{s_0}$: $(\perp, \perp, \perp, \perp, \perp, \perp)$

$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $h_1 : (\textit{sunny, warm, normal, strong, warm, same})$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $h_2 : (\textit{sunny, warm, ?, strong, warm, same})$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $h_3 : (\textit{sunny, warm, ?, strong, warm, same})$

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$ $h_4 : (\textit{sunny, warm, ?, strong, ?, ?})$

Rule-Based Learning of Simple Concepts

Find-S Algorithm: Discussion

1. Did we learn the only concept—or are there others?
2. Why should one pursuit the maximally specific hypothesis?
3. What if several maximally specific hypotheses exist?
4. Inconsistencies in the example set D remain undetected.
5. An inappropriate hypothesis structure or space H remains undetected.

Rule-Based Learning of Simple Concepts

The Set of Consistent Hypothesis

Definition 5 (Version Space)

The version space H_D, $H_D \subseteq H$, of a hypothesis space H and a example set D is comprised of all model functions $y() \in H$ that are consistent with the set D of examples:

$$H_D = \{y() \mid y() \in H \ \wedge \ (\ \forall(\mathbf{x}, c) \in D : \ y(\mathbf{x}) = c \ ) \ \}$$

Rule-Based Learning of Simple Concepts

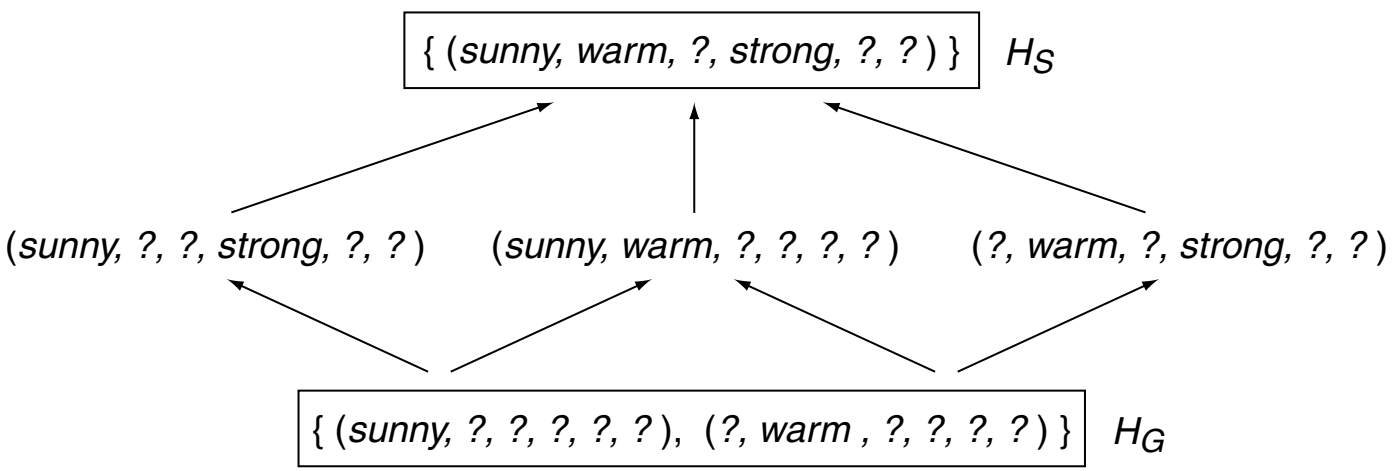
The Set of Consistent Hypothesis (continued)

Definition 5 (Version Space)

The version space H_D, $H_D \subseteq H$, of a hypothesis space H and a example set D is comprised of all model functions $y() \in H$ that are consistent with the set D of examples:

$$H_D = \{y() \mid y() \in H \ \wedge \ (\ \forall(\mathbf{x}, c) \in D : \ y(\mathbf{x}) = c \ ) \ \}$$

Illustration of H_D for the example set D :



Remarks:

- The term “version space” reflects the fact that H_D represents the set of all consistent versions of the target concept that are encoded in D.
- A naive approach for the construction of the version space is the following: (1) enumeration of all members of H, and, (2) elimination of those $y() \in H$ where $y(\mathbf{x}) \neq c$ for some $(\mathbf{x}, c) \in D$ holds. This approach presumes a finite hypothesis space H and is feasible only for toy problems.

Rule-Based Learning of Simple Concepts

The Set of Consistent Hypothesis (continued)

Definition 6 (Boundary Sets of a Version Space)

Let H be hypothesis space and let D be set of examples. Then, based on the $\geq_g$-relation, the set of maximally general hypotheses, H_G, is defined as follows:

$$H_G = \{\, y() \mid y() \in H \wedge \textit{consistent}(y(), D) \wedge (\nexists y'() : y'() \in H \wedge y'() >_g y() \wedge \textit{consistent}(y'(), D)) \,\}$$

Similarly, the set of maximally specific (i.e., minimally general) hypotheses, H_S, is defined as follows:

$$H_S = \{\, y() \mid y() \in H \wedge \textit{consistent}(y(), D) \wedge (\nexists y'() : y'() \in H \wedge y() >_g y'() \wedge \textit{consistent}(y'(), D)) \,\}$$

Rule-Based Learning of Simple Concepts

The Set of Consistent Hypothesis (continued)

Theorem 7 (Version Space Representation)

Let X be a multiset of feature vectors, $C = \{0, 1\}$ be a set of classes, and H be a set of boolean-valued functions with domain X. Moreover, let $D \subseteq X \times C$ be a multiset of examples.

Then, based on the $\geq_g$-relation, each member of the version space H_D lies between two members of H_G and H_S respectively:

$$H_D = \{y() \mid \; y() \in H \wedge \\ (\exists y_g() \in H_G \;\; \exists y_s() \in H_S : \;\; y_g() \geq_g y() \geq_g y_s() \;) \}$$

Remarks:

- Theorem 7 allows us to characterize the set of all consistent hypotheses by the two boundary sets H_G and H_S. The correctness of the theorem is not obvious.

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm [Mitchell 1997]

Algorithm:	CE	Candidate Elimination Algorithm.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in X$ and target class $c \in \{0, 1\}$.
Output:	(H_G, H_S)	Version space representation of version space H_D under example set D.

CE(D)

1. Initialize $H_G = \{h_{g_0}\}$ and $H_S = \{h_{s_0}\}$. Then the examples $(\mathbf{x}, c)$ in D are iterated:
2. If $\mathbf{x}$ is a positive example:
   - Remove from H_G any hypothesis that is not consistent with $\mathbf{x}$.
   - For each hypothesis $y_s()$ in H_S that is not consistent with $\mathbf{x}$:
     - Remove $y_s()$ from H_S and add to H_S all minimal generalizations $y()$ of $y_s()$ where: (1) $y()$ is consistent with $\mathbf{x}$, and (2) some member of H_G is more general than $y()$.
     - Remove from H_S any hypothesis that is less specific than another hypothesis in H_S.

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm [Mitchell 1997]

Algorithm:	CE	Candidate Elimination Algorithm.
Input:	D	Training examples $(\mathbf{x}, c)$ with $\mathbf{x} \in X$ and target class $c \in \{0, 1\}$.
Output:	(H_G, H_S)	Version space representation of version space H_D under example set D.

CE(D)

1. Initialize $H_G = \{h_{g_0}\}$ and $H_S = \{h_{s_0}\}$. Then the examples $(\mathbf{x}, c)$ in D are iterated:
2. If $\mathbf{x}$ is a positive example:
   - Remove from H_G any hypothesis that is not consistent with $\mathbf{x}$.
   - For each hypothesis $y_s()$ in H_S that is not consistent with $\mathbf{x}$:
     - Remove $y_s()$ from H_S and add to H_S all minimal generalizations $y()$ of $y_s()$ where: (1) $y()$ is consistent with $\mathbf{x}$, and (2) some member of H_G is more general than $y()$.
     - Remove from H_S any hypothesis that is less specific than another hypothesis in H_S.
3. If $\mathbf{x}$ is a negative example:
   - Remove from H_S any hypothesis that is not consistent with $\mathbf{x}$.
   - For each hypothesis $y_g()$ in H_G that is not consistent with $\mathbf{x}$:
     - Remove $y_g()$ from H_G and add to H_G all minimal specializations $y()$ of $y_g()$ where: (1) $y()$ is consistent with $\mathbf{x}$, and (2) some member of H_S is more specific than $y()$.
     - Remove from H_G any hypothesis that is less general than another hypothesis in H_G.
4. return(H_G, H_S)

Remarks:

- All hypothesis "between" the sets H_G and H_S are consistent with all examples seen so far; i.e., they accept the positive examples and reject the negative examples.
- The basic idea of Candidate Elimination is as follows:
  - Deal with false positives. A maximally general hypothesis $y_g() \in H_G$ tolerates the negative examples in first instance. Hence, $y_g()$ needs to be constrained (= specialized) with regard to each negative example that is not consistent with $y_g()$.
  - Deal with false negatives. A maximally specific hypothesis $y_s() \in H_S$ restricts the positive examples in first instance. Hence, $y_s()$ needs to be relaxed (= generalized) with regard to each positive example that is not consistent with $y_s()$.
- The boundary set H_G of the version space summarizes the information from the previously encountered negative examples. The boundary set H_S forms a summary of the previously encountered positive examples.

Rule-Based Learning of Simple Concepts

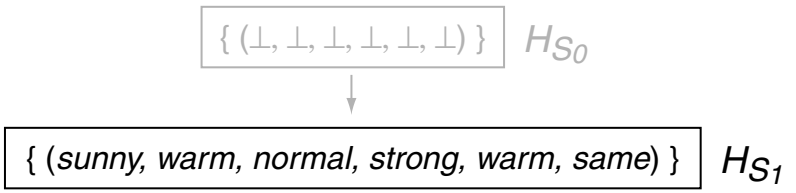
Candidate Elimination Algorithm: Illustration

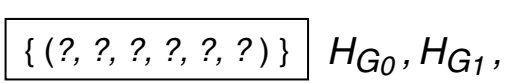
$$\boxed{\{\,(\bot, \bot, \bot, \bot, \bot, \bot)\,\}} \quad H_{S_0}$$

$$\boxed{\{\,(?, ?, ?, ?, ?, ?)\,\}} \quad H_{G_0},$$

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm: Illustration

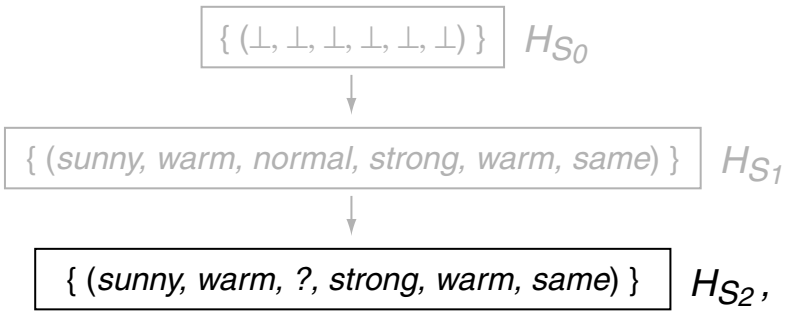


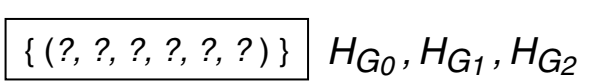


$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad\quad$ $\textit{EnjoySurfing}(\mathbf{x}_1) = 1$

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm: Illustration



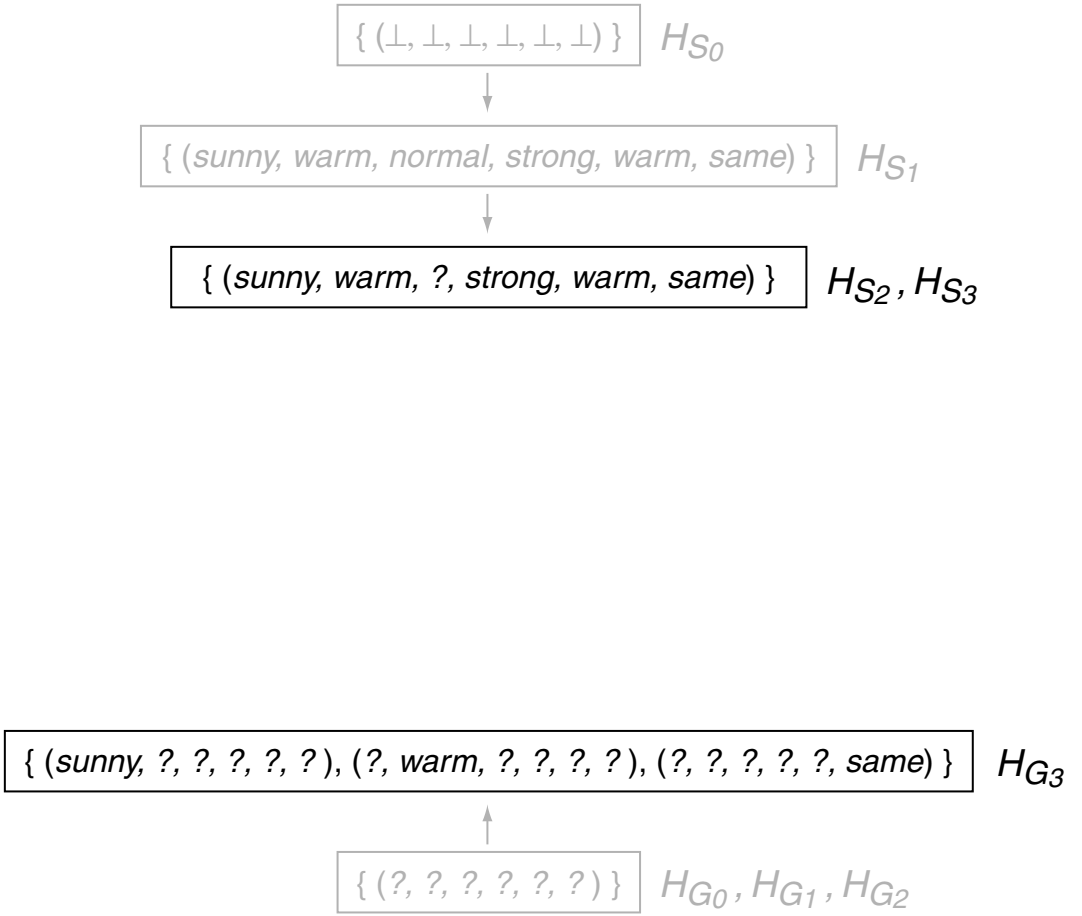


$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_1) = 1$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_2) = 1$

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm: Illustration



$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_1) = 1$

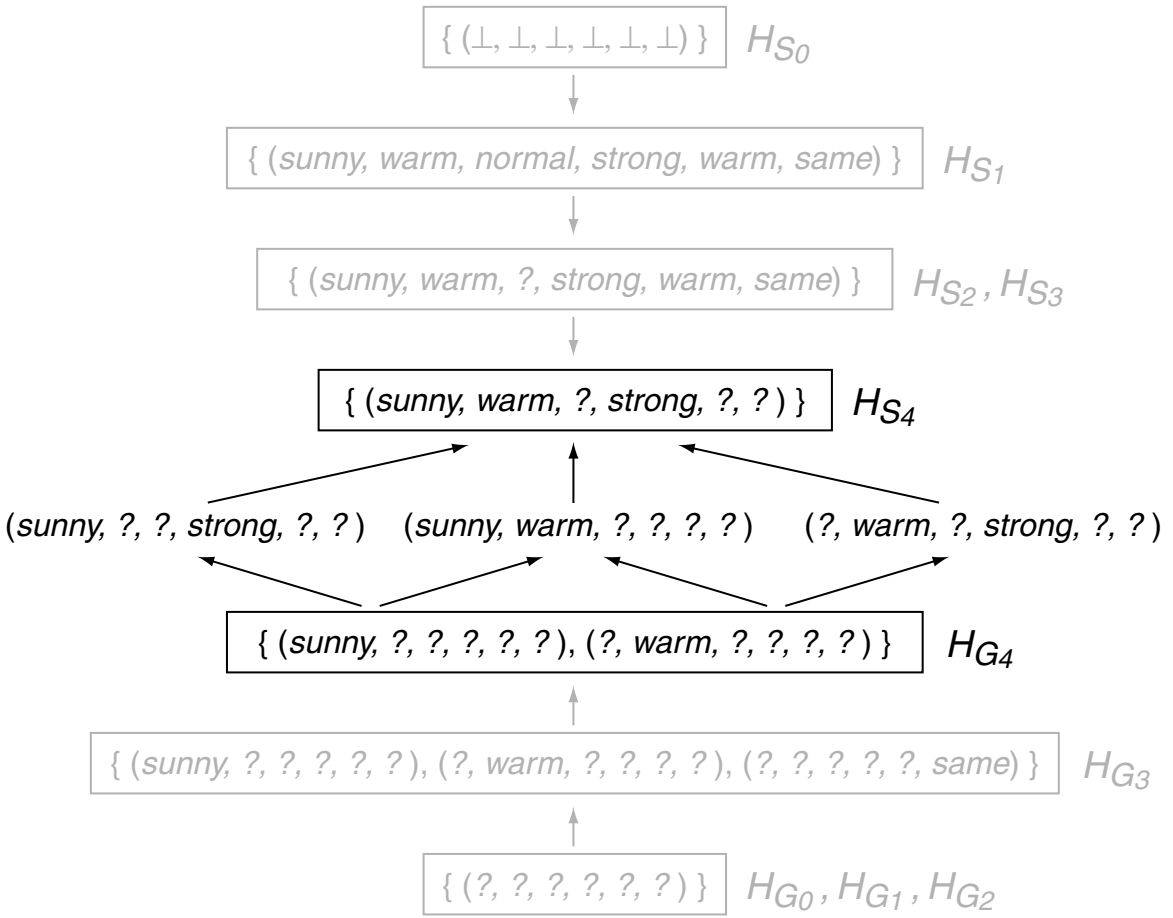
$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_2) = 1$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_3) = 0$

[feature domains] [algorithm]

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm: Illustration



$\mathbf{x}_1 = (\textit{sunny, warm, normal, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_1) = 1$

$\mathbf{x}_2 = (\textit{sunny, warm, high, strong, warm, same})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_2) = 1$

$\mathbf{x}_3 = (\textit{rainy, cold, high, strong, warm, change})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_3) = 0$

$\mathbf{x}_4 = (\textit{sunny, warm, high, strong, cool, change})$ $\quad$ $\textit{EnjoySurfing}(\mathbf{x}_4) = 1$

[feature domains] [algorithm]

Rule-Based Learning of Simple Concepts

Candidate Elimination Algorithm: Discussion

1. What about selecting examples from D according to a certain strategy?
   Keyword: active learning

2. What are partially learned concepts and how to exploit them?
   Keyword: ensemble classification

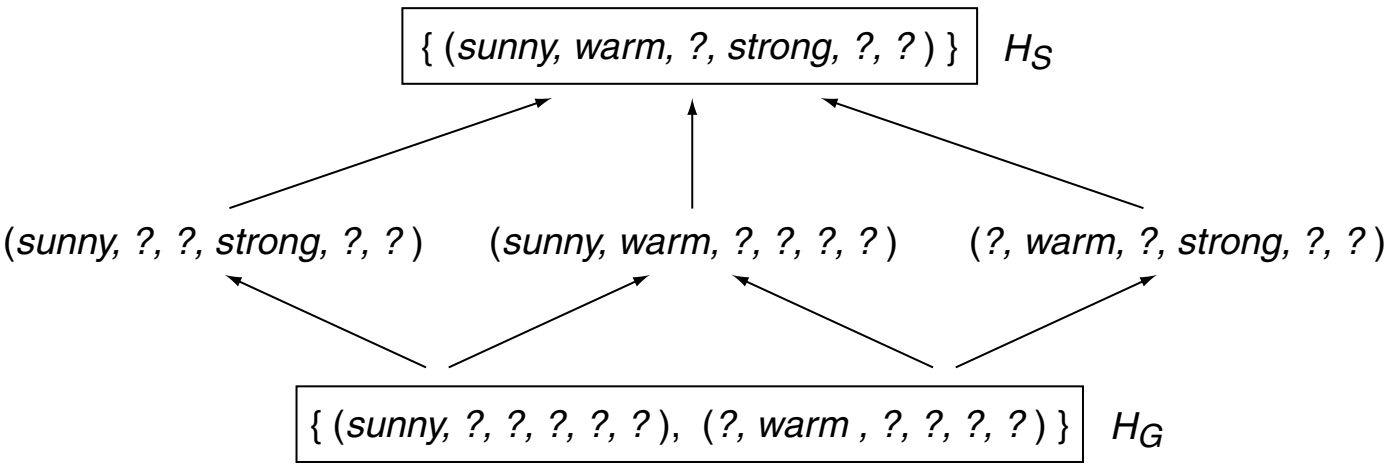
3. The version space as defined here is “biased”. What does this mean?
   Keywords: representation bias, search bias

4. Will Candidate Elimination converge towards the correct hypothesis?

5. When does one end up with an empty version space?

Rule-Based Learning of Simple Concepts

Active Learning: Selecting Examples

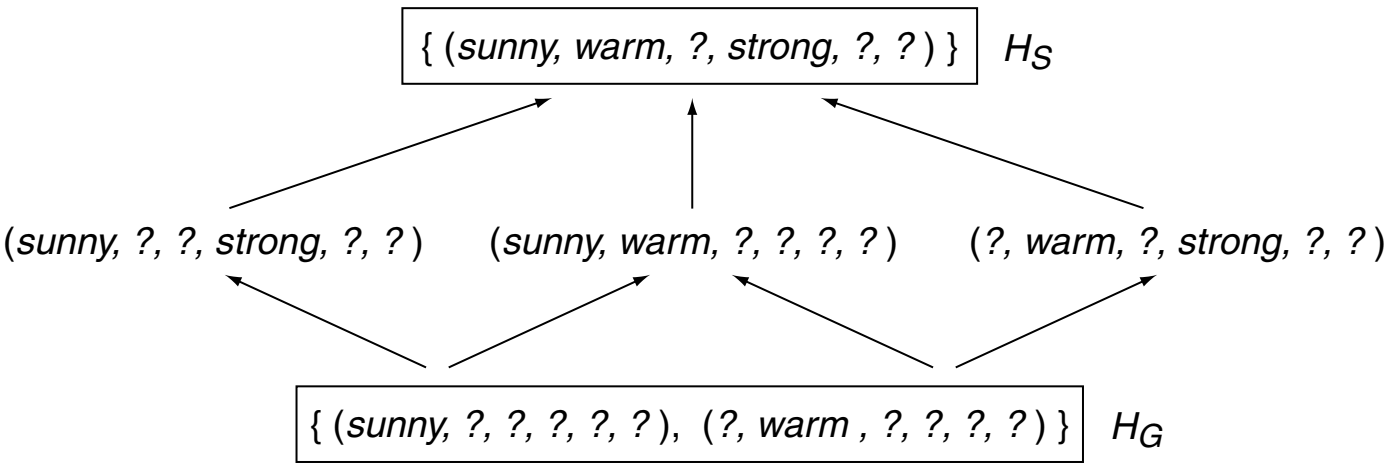


An example from which we can “maximally” learn:

$\mathbf{x}_7 = (\textit{sunny, warm, normal, light, warm, same})$

Rule-Based Learning of Simple Concepts

Active Learning: Selecting Examples (continued)



An example from which we can “maximally” learn:

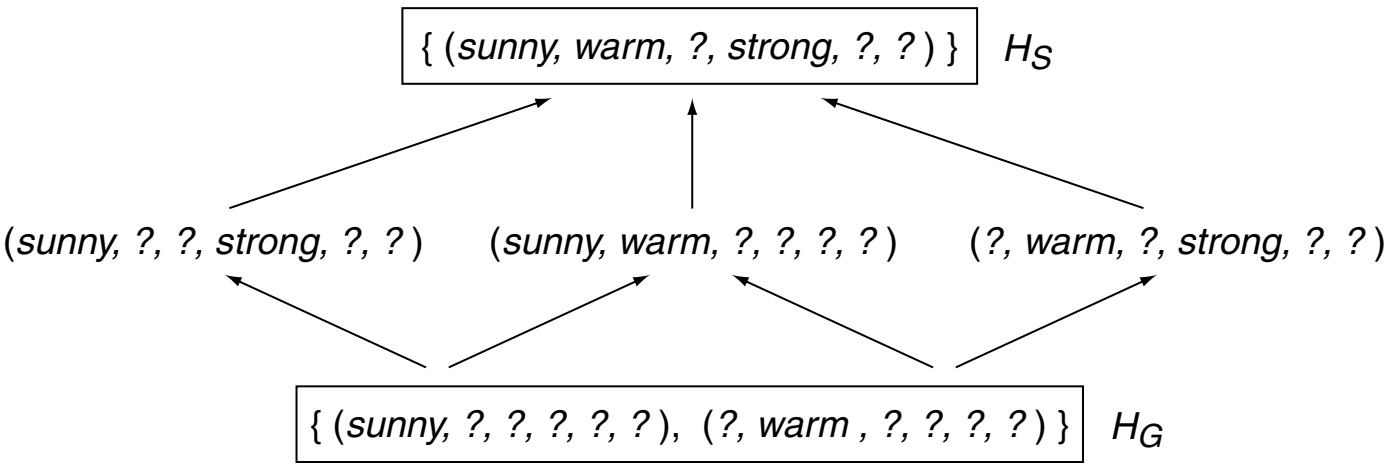
$\mathbf{x}_7 = (\textit{sunny, warm, normal, light, warm, same})$

Irrespective the value of c, $(\mathbf{x}_7, c)$ is consistent with 3 of the 6 hypotheses:

- If $\textit{EnjoySurfing}(\mathbf{x}_7) = 1$ H_S can be further generalized.
- If $\textit{EnjoySurfing}(\mathbf{x}_7) = 0$ H_G can be further specialized.

Rule-Based Learning of Simple Concepts

Ensembles: Exploiting Partially Learned Concepts

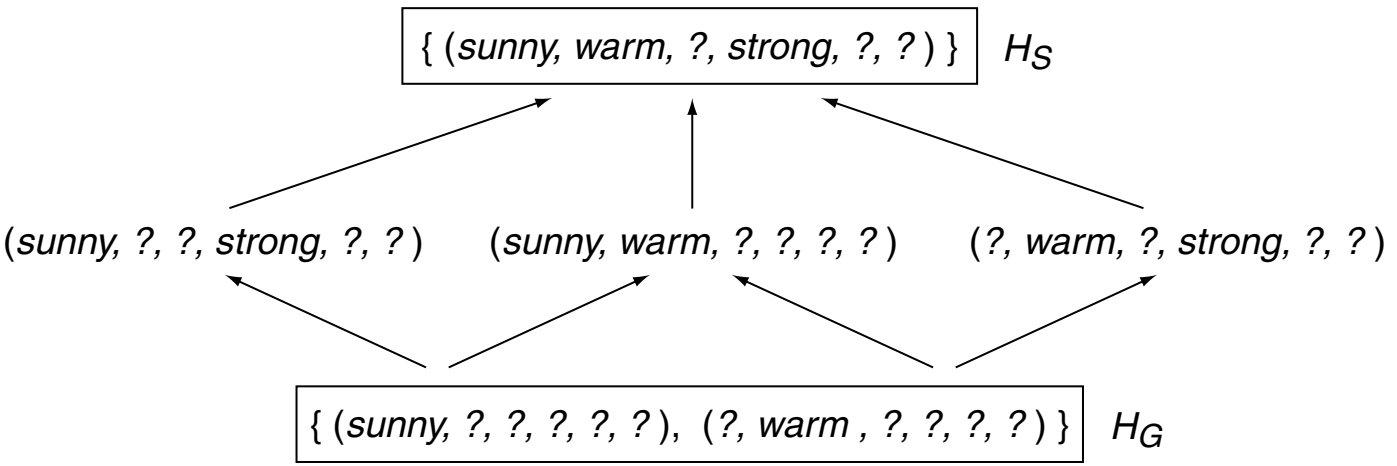


Combine the 6 classifiers in the version space to decide about new examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
5	sunny	warm	normal	strong	cool	change	
6	rainy	cold	normal	light	warm	same	
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

Rule-Based Learning of Simple Concepts

Ensembles: Exploiting Partially Learned Concepts (continued)

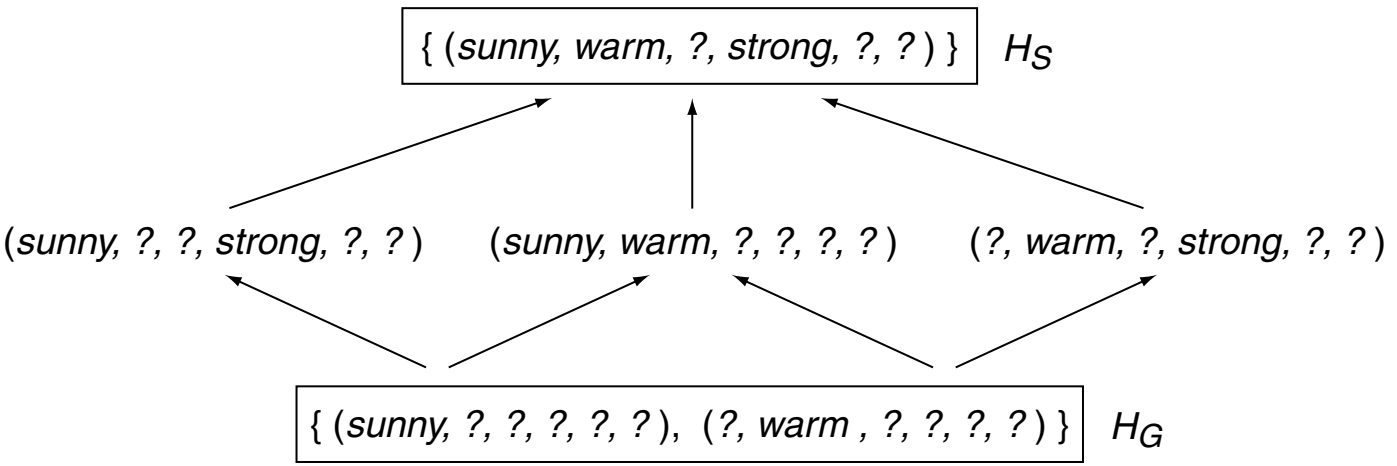


Combine the 6 classifiers in the version space to decide about new examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

Rule-Based Learning of Simple Concepts

Ensembles: Exploiting Partially Learned Concepts (continued)

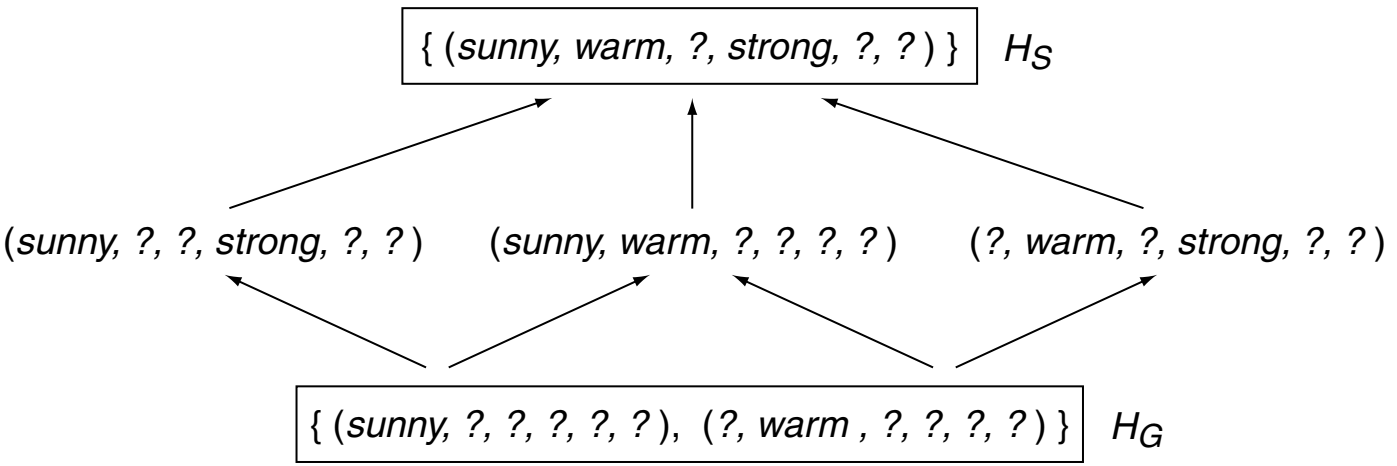


Combine the 6 classifiers in the version space to decide about new examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	
8	sunny	cold	normal	strong	warm	same	

Rule-Based Learning of Simple Concepts

Ensembles: Exploiting Partially Learned Concepts (continued)

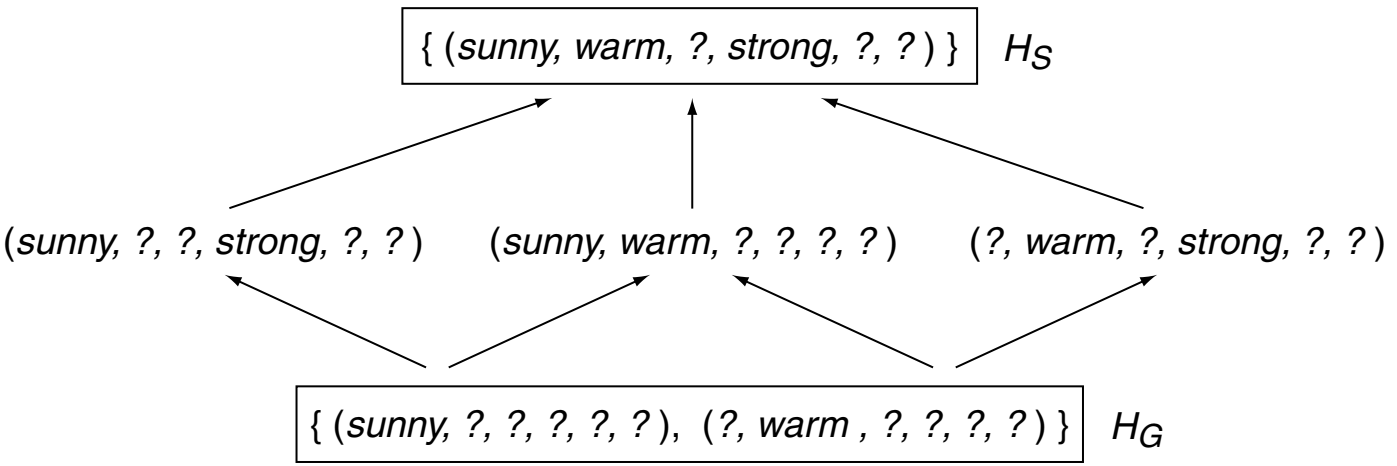


Combine the 6 classifiers in the version space to decide about new examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	3+ : 3–
8	sunny	cold	normal	strong	warm	same	

Rule-Based Learning of Simple Concepts

Ensembles: Exploiting Partially Learned Concepts (continued)



Combine the 6 classifiers in the version space to decide about new examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
5	sunny	warm	normal	strong	cool	change	6+ : 0–
6	rainy	cold	normal	light	warm	same	0+ : 6–
7	sunny	warm	normal	light	warm	same	3+ : 3–
8	sunny	cold	normal	strong	warm	same	2+ : 4–

Rule-Based Learning of Simple Concepts

Inductive Bias

Additional training examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

➜ $H_S = \{ (?, \textit{warm}, \textit{normal}, \textit{strong}, \textit{cool}, \textit{change}) \}$

Rule-Based Learning of Simple Concepts

Inductive Bias (continued)

Additional training examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

➜ $H_S = \{ (?, \textit{warm}, \textit{normal}, \textit{strong}, \textit{cool}, \textit{change}) \}$

⋮

11	rainy	warm	normal	strong	cool	change	no

➜ $H_S = \{ \}$

Discussion:

- What assumptions about the target concept are met by the learner a priori?

Rule-Based Learning of Simple Concepts

Inductive Bias (continued)

Additional training examples:

Example	Sky	Temperature	Humidity	Wind	Water	Forecast	EnjoySurfing
9	sunny	warm	normal	strong	cool	change	yes
10	cloudy	warm	normal	strong	cool	change	yes

➜ $H_S = \{ (?, \textit{warm}, \textit{normal}, \textit{strong}, \textit{cool}, \textit{change}) \}$

⋮

11	rainy	warm	normal	strong	cool	change	no

➜ $H_S = \{ \}$

Discussion:

- What assumptions about the target concept are met by the learner a priori?

➜ H may be designed to contain more elaborate model functions:
$(\textit{sunny}, ?, ?, ?, ?, ?) \vee (\textit{cloudy}, ?, ?, ?, ?, ?)$

Rule-Based Learning of Simple Concepts

Inductive Bias (continued)

"The policy by which a [learning] *algorithm generalizes from observed training examples to classify unseen instances is its inductive bias.* [. . .]

Inductive bias is the set of assumptions that,
together with the training data,
deductively justify the classification by the learner to future instances."

[p.43, Mitchell 1997]

Rule-Based Learning of Simple Concepts

Inductive Bias (continued)

- In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| = 2^{|X|}$ elements.
- A learning algorithm that considers all possible hypotheses as equally likely makes no a priori assumption with regard to the target concept.
- A learning algorithm without a priori assumptions has no *inductive bias*.

Rule-Based Learning of Simple Concepts

Inductive Bias (continued)

- In a binary classification problem the unrestricted (= unbiased) hypothesis space contains $|\mathcal{P}(X)| = 2^{|X|}$ elements.
- A learning algorithm that considers all possible hypotheses as equally likely makes no a priori assumption with regard to the target concept.
- A learning algorithm without a priori assumptions has no *inductive bias*.

➜ A learning algorithm without inductive bias has no directive to classify unseen examples. Put another way: the learner cannot *generalize*.

➜ A learning algorithm without inductive bias can only *memorize*.

Which algorithm (Find-S, Candidate Elimination) has a stronger inductive bias?