Introduction to Machine Learning

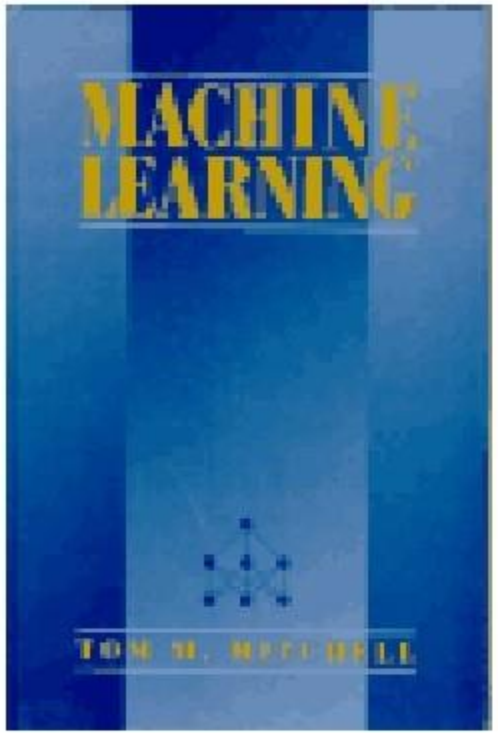
Mahesh G Huddar

Assistant Professor

Dept. of CSE, HIT, Nidasoshi

Books

Machine Learning by Tom M. Mitchell



What is machine learning?

- A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.

- As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.

What is Machine Learning ???

Machine Learning (ML) is concerned with the question of how to construct computer programs that automatically improves with experience.

Definition (Learning)

A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**. (Tom Mitchell, 1998)

A Handwritten Recognition Problem

- Task T : recognizing and classifying handwritten words within images

- Performance measure P: % of words correctly classified

- Training experience E : a database of handwritten words with given classifications

A Robot Driving Learning Problem

- Task T : driving on public four-lane highways using vision sensors

Performance measure P: average distance traveled before an error (as judged by human overseer)

- Training experience E : a sequence of images and steering commands recorded while observing a human driver

Checker Learning Problem

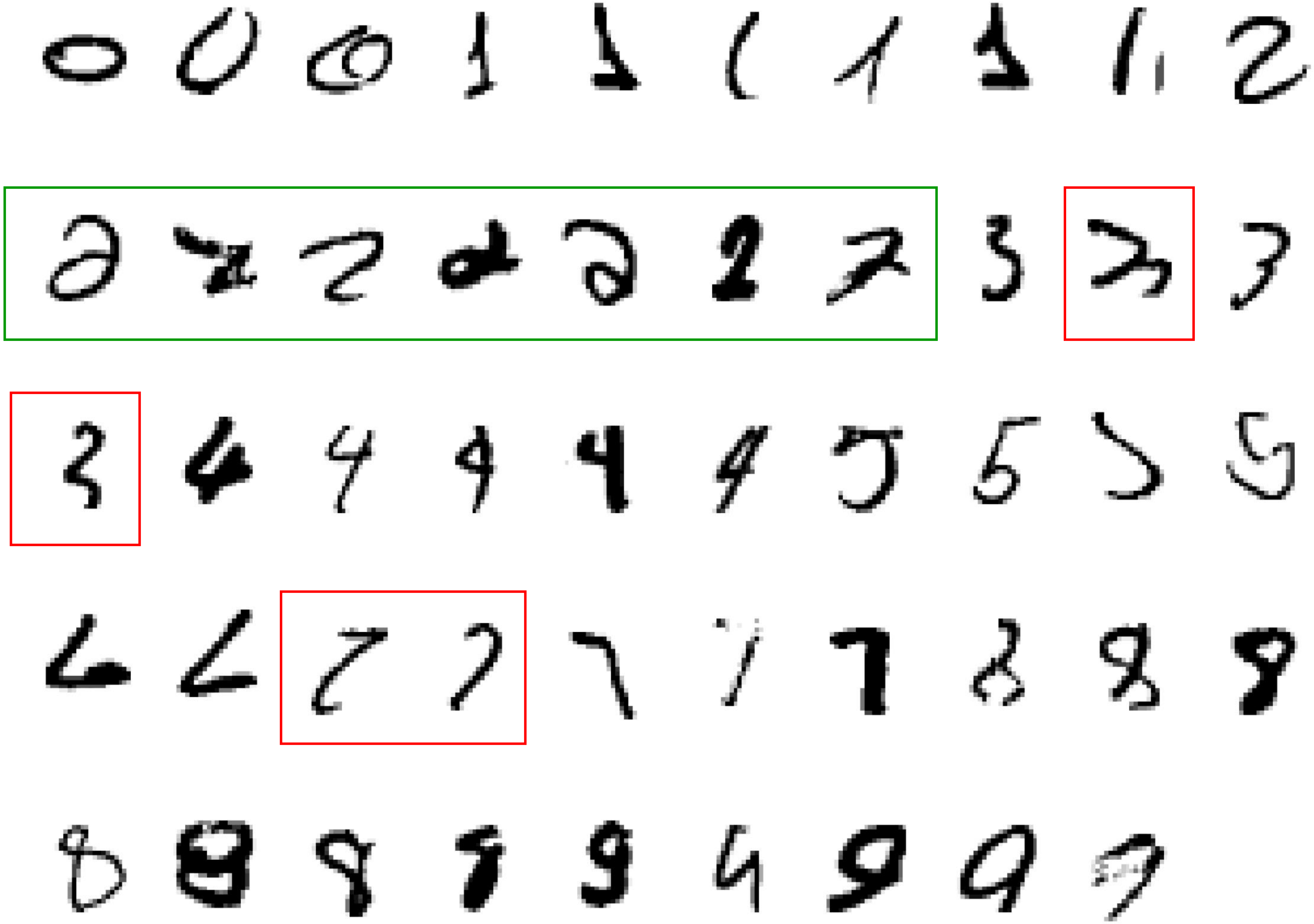
A computer program that learns to play checkers might improve its performance as ***measured by its ability to win*** at the class of tasks involving ***playing checkers games*****,** through experience ***obtained*** by ***playing games against itself***

- Task **T** : playing checkers
- Performance measure **P**: % of game won against opponents
- Training experience **E** : playing practice game against itself

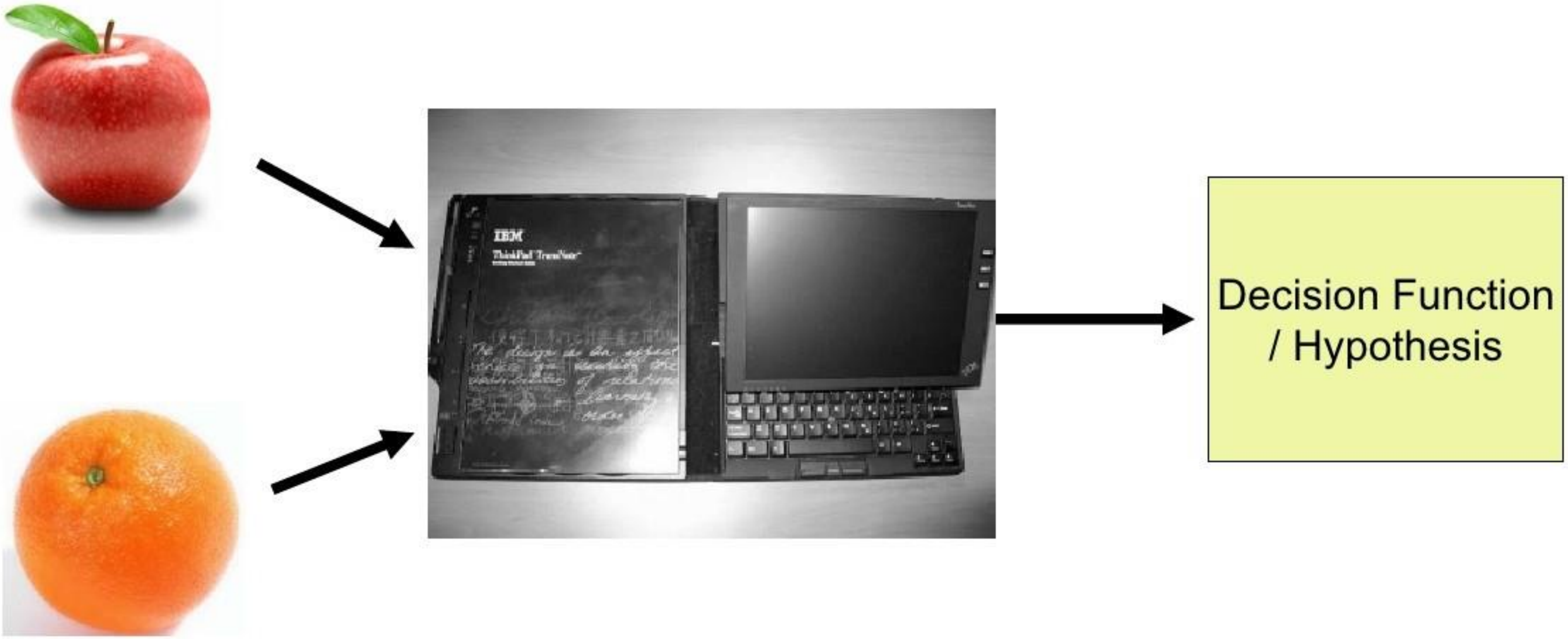
When Machine Learning ???

- Computers applied for solving a complex problem
- No known method for computing output is present
- When computation is expensive

A classic example of a task that requires machine learning: It is very hard to say what makes a 2

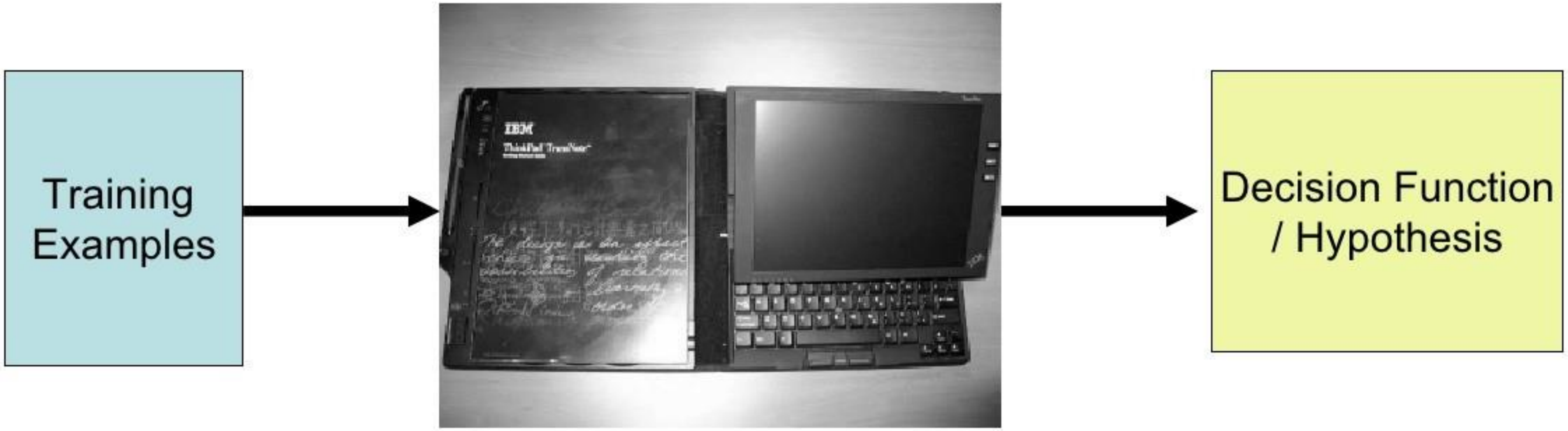


What Machine Learning does???



Decision Function / Hypothesis

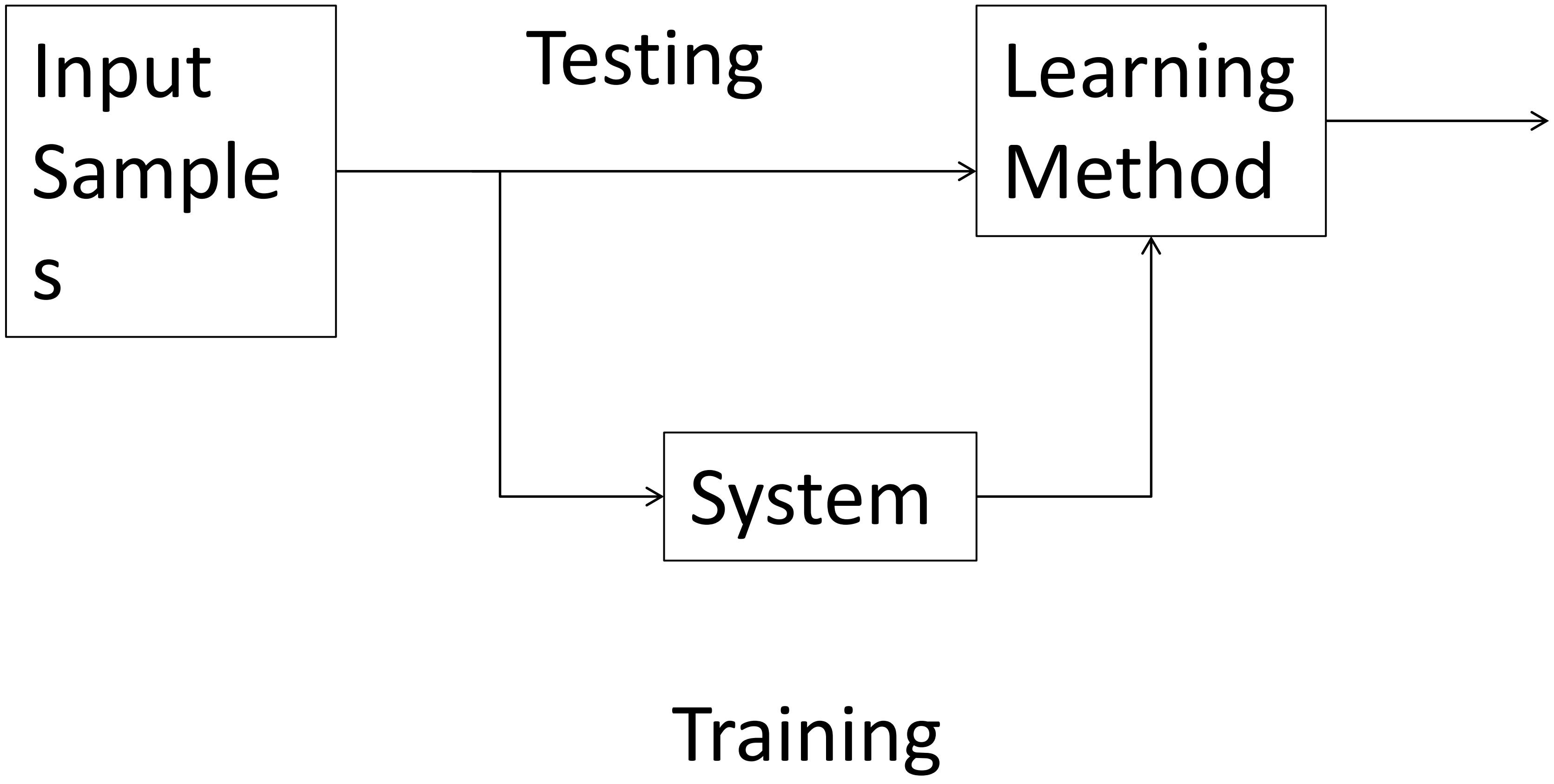
What Machine Learning does???



Training Examples

Decision Function / Hypothesis

Learning system model



Input
Sample
s

Testing

Learning
Method

System

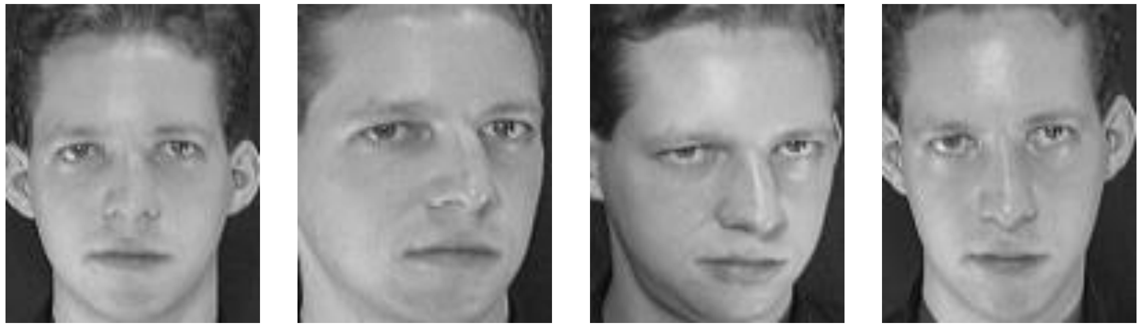
Training

Applications

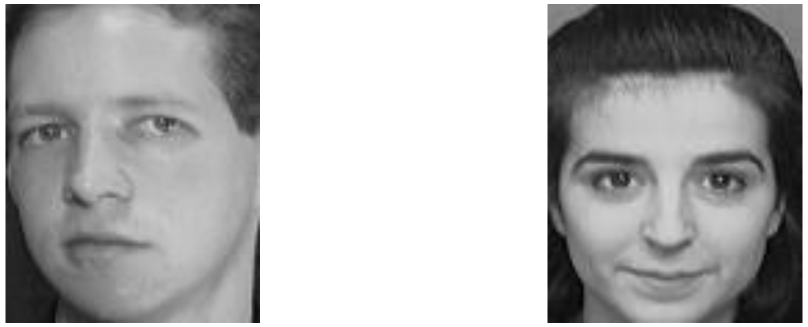
- Face detection
- Object detection and recognition
- Image segmentation
- Multimedia event detection
- Economical and commercial usage
- Natural Language Processing
- Spam Filtering
- Medical Diagnosis
- Learning to drive vehicles
- Game Playing
- and many more….

Face Recognition

Training examples of a person

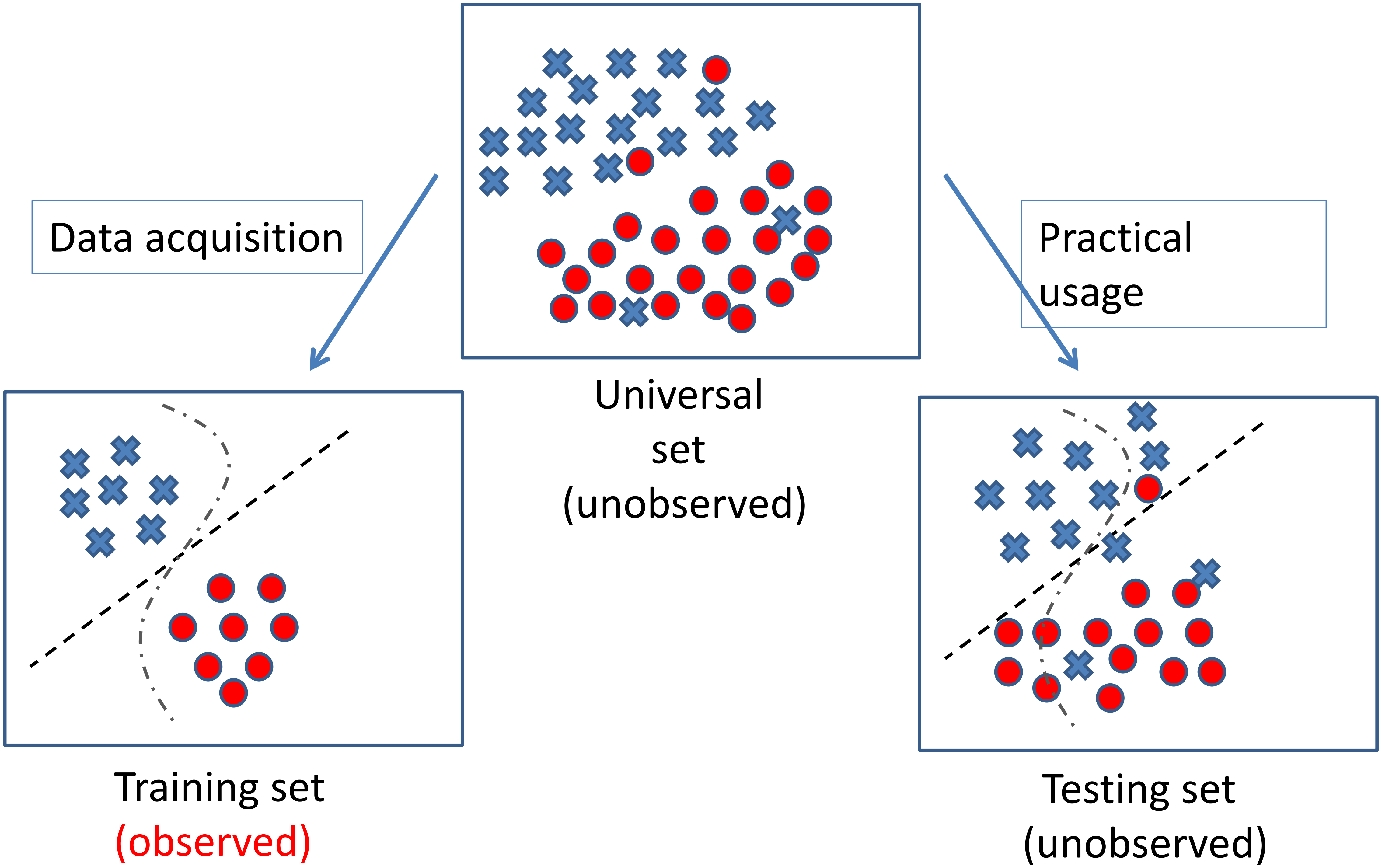


Test images



AT&T Laboratories, Cambridge UK
http://www.uk.research.att.com/facedatabase.html

Training and testing



Data acquisition

Practical usage

Universal set (unobserved)

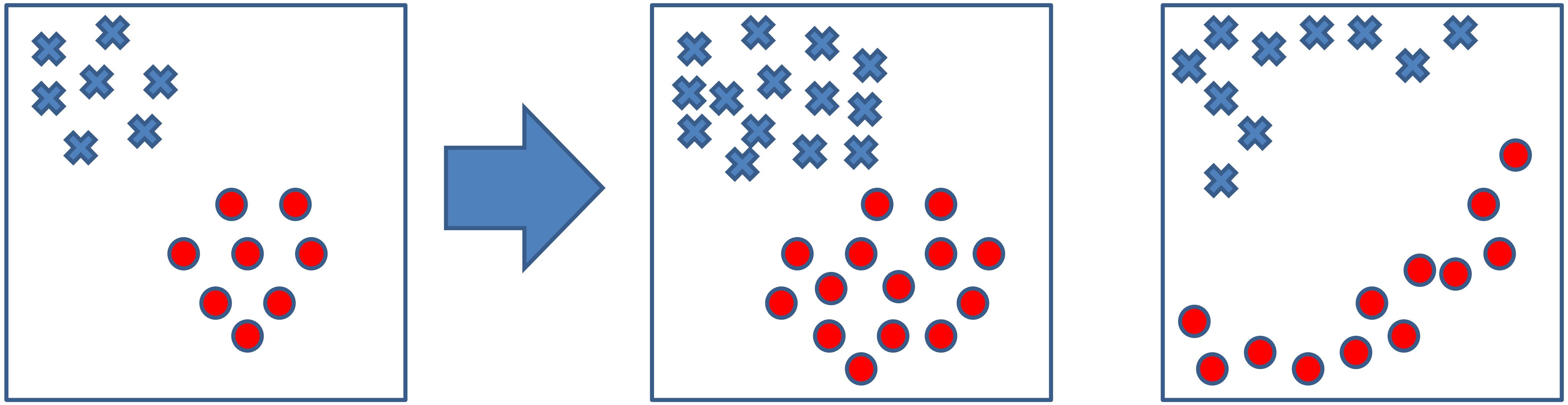
Training set (observed)

Testing set (unobserved)

Training and testing

- Training is the process of making the system able to learn.

- No free lunch rule:
  - Training set and testing set come from the same distribution
  - Need to make some assumptions or bias



Performance

- There are several factors affecting the performance:
  - **Types of training** provided
  - The form and extent of any initial **background knowledge**
  - The **type of feedback** provided
  - The **learning algorithms** used

- Two important factors:
  - Modeling
  - Optimization

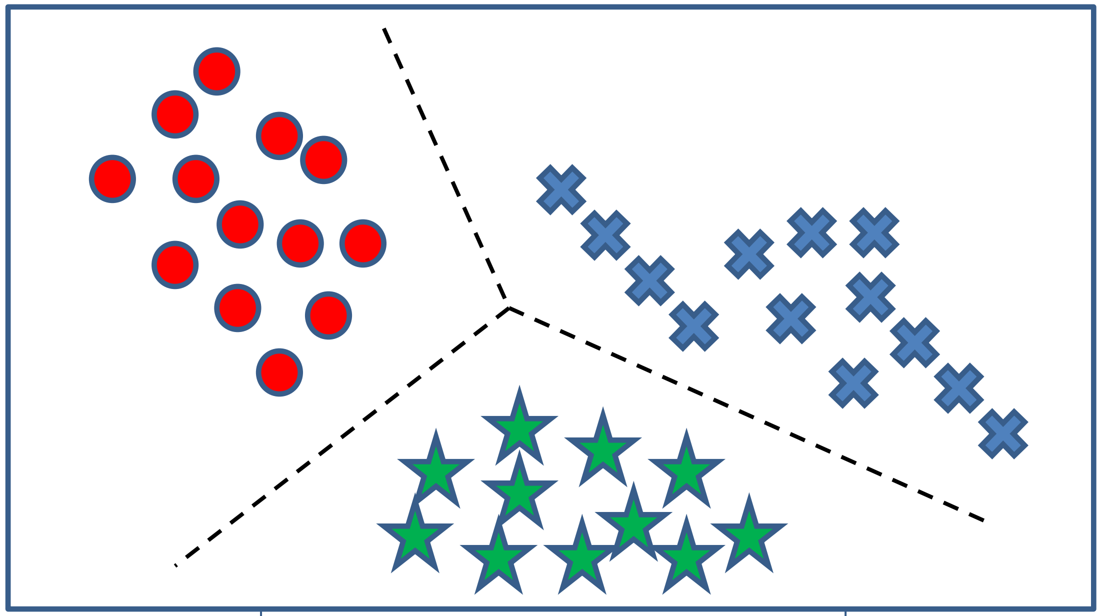
Algorithms

- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

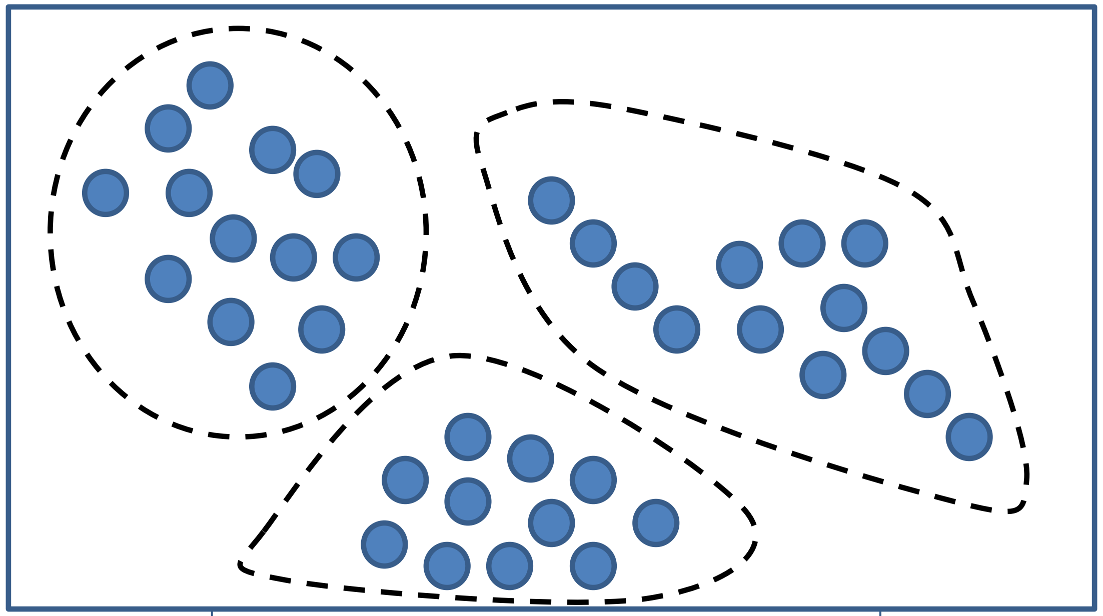
Algorithms

- **Supervised learning** ($\{x_n \in R^d, y_n) \in R\}_{n=1}^{N}$
  - Prediction
  - Classification (discrete labels), Regression (real values)
- **Unsupervised learning** ($\{x_n \in) R^d\}_{n=1}^{N}$
  - Clustering
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction
- **Semi-supervised learning**
- **Reinforcement learning**
  - Decision making (robot, chess machine)

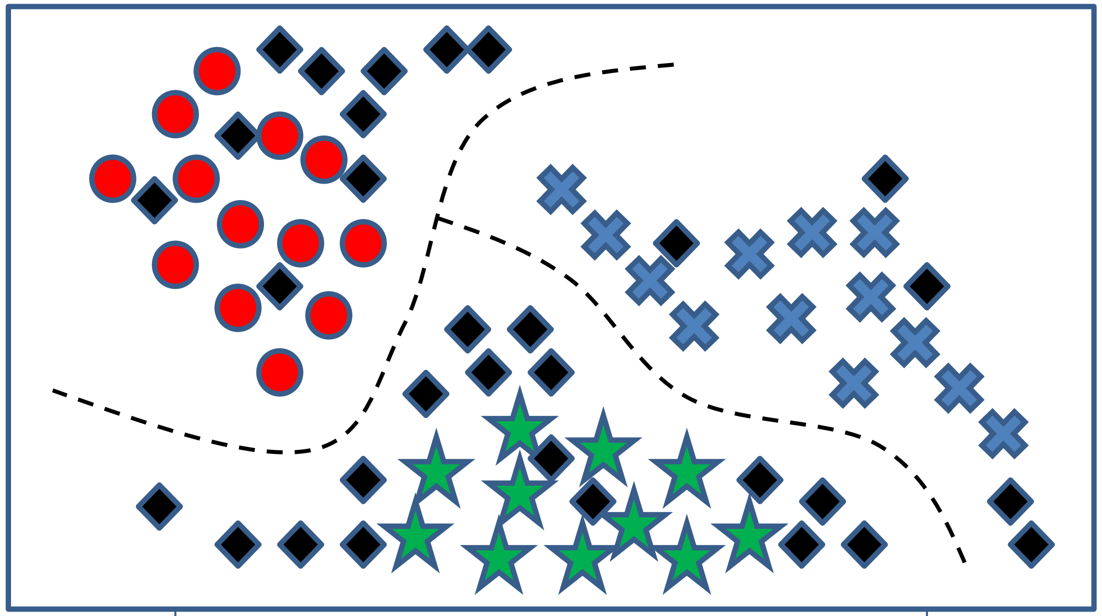
Algorithms



Supervised learning

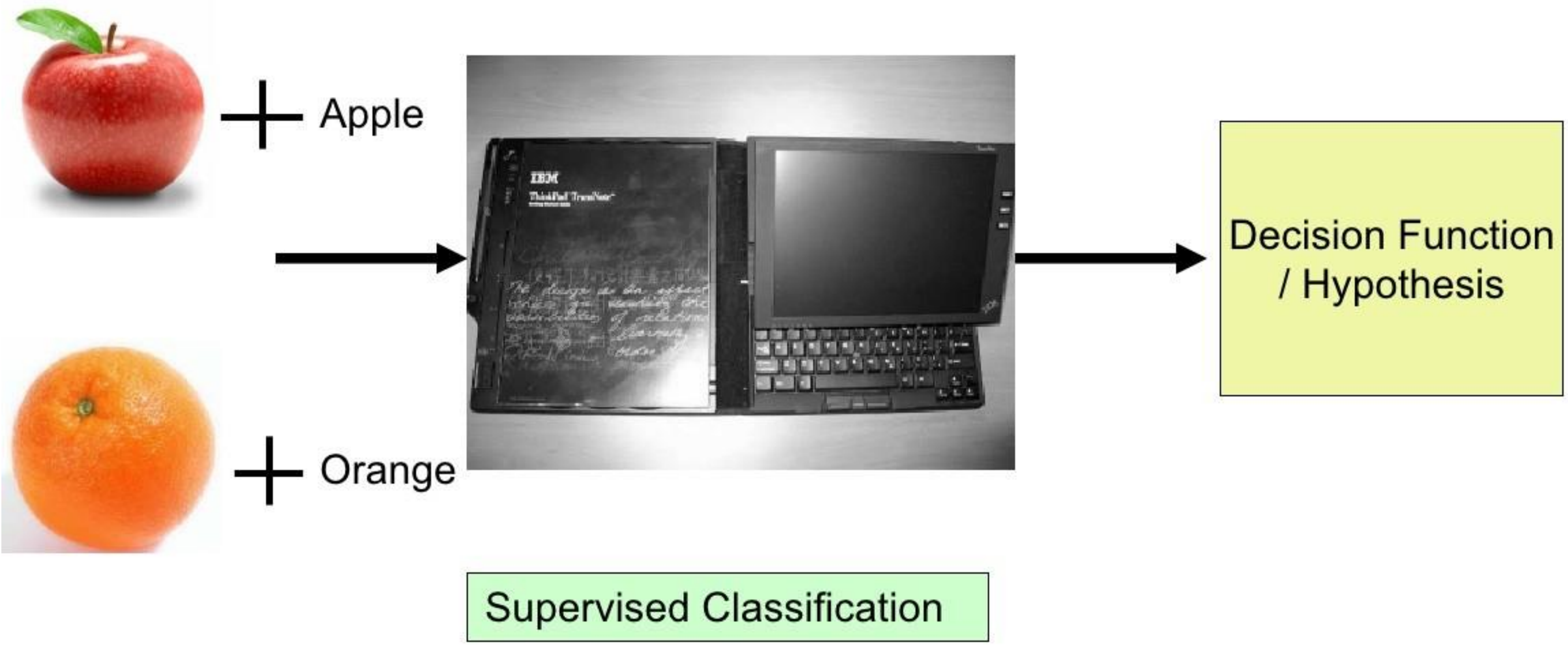


Unsupervised learning



Semi-supervised learning

Supervised Learning



+ Apple

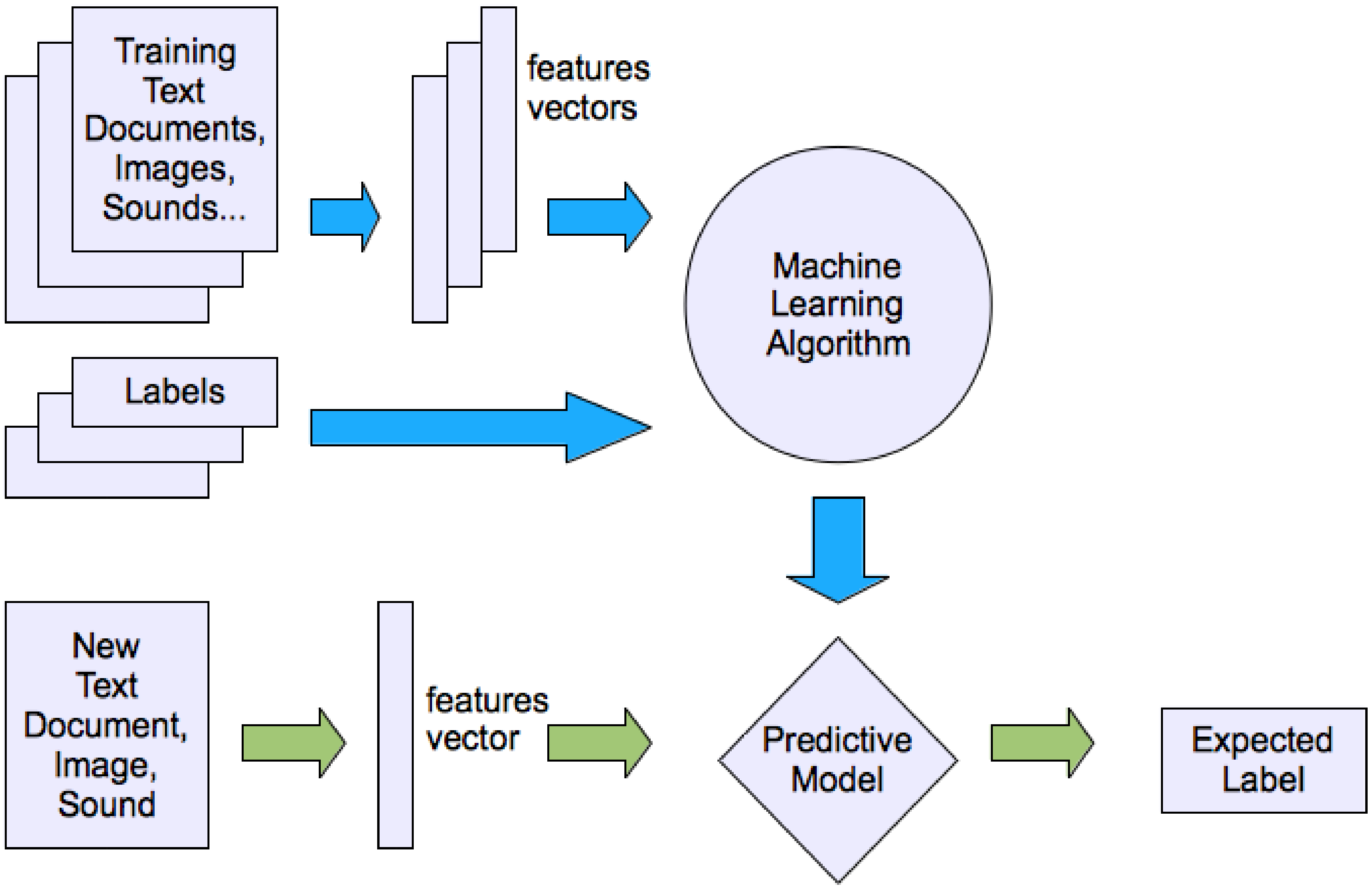
+ Orange

Decision Function / Hypothesis

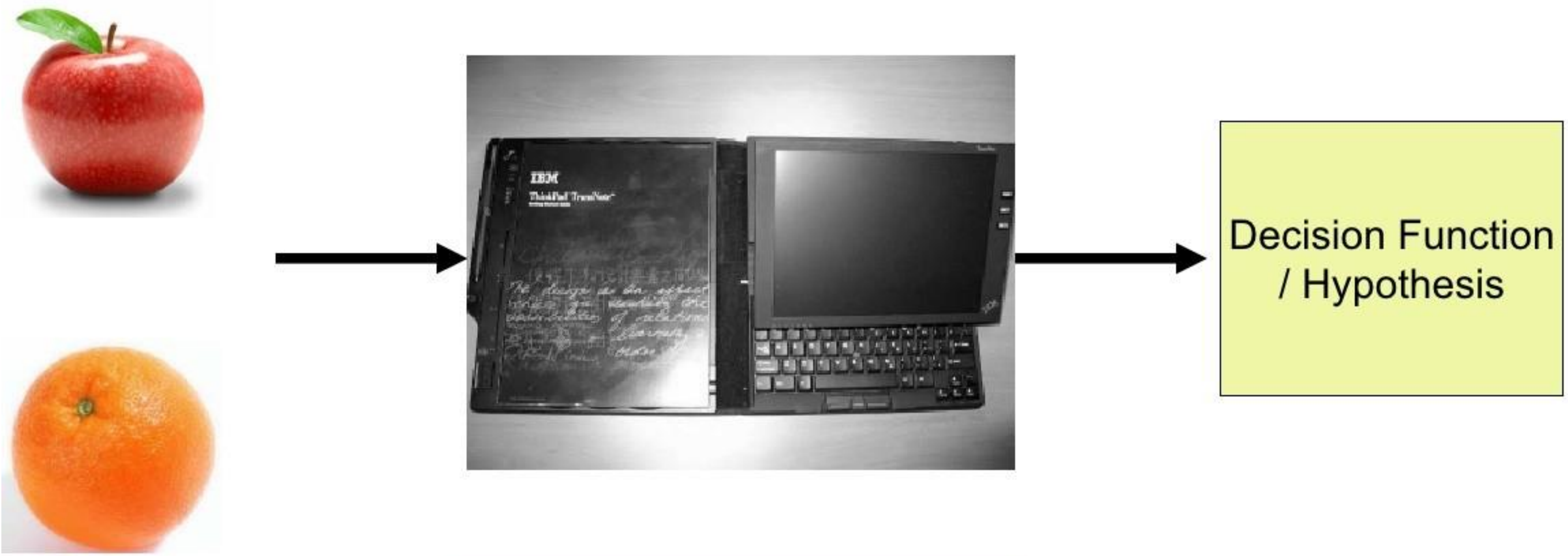
Supervised Classification

Machine learning structure

- Supervised learning


Training Text Documents, Images, Sounds...
features vectors
Machine Learning Algorithm
Labels
New Text Document, Image, Sound
features vector
Predictive Model
Expected Label

Unsupervised Learning

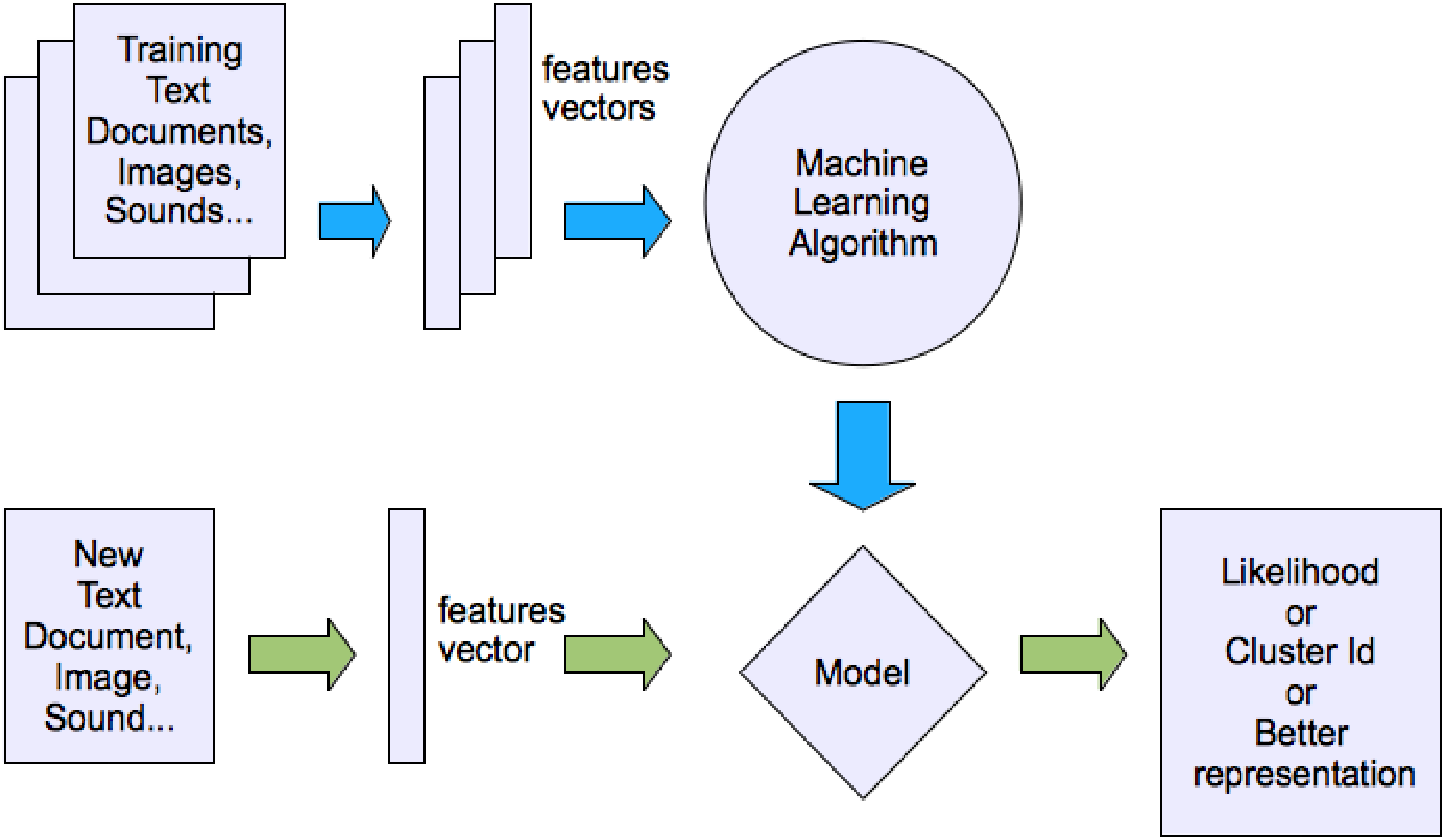


Decision Function / Hypothesis

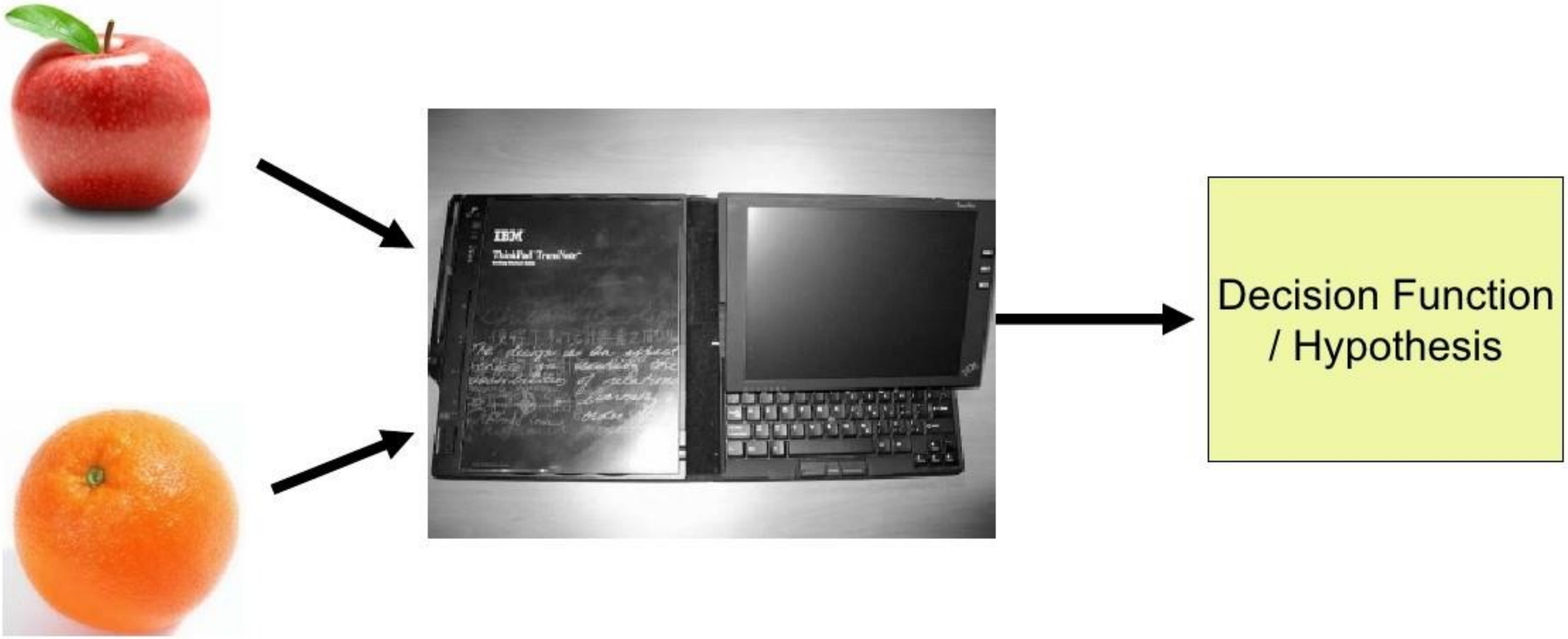
Unsupervised Classification

Machine learning structure

- Unsupervised learning

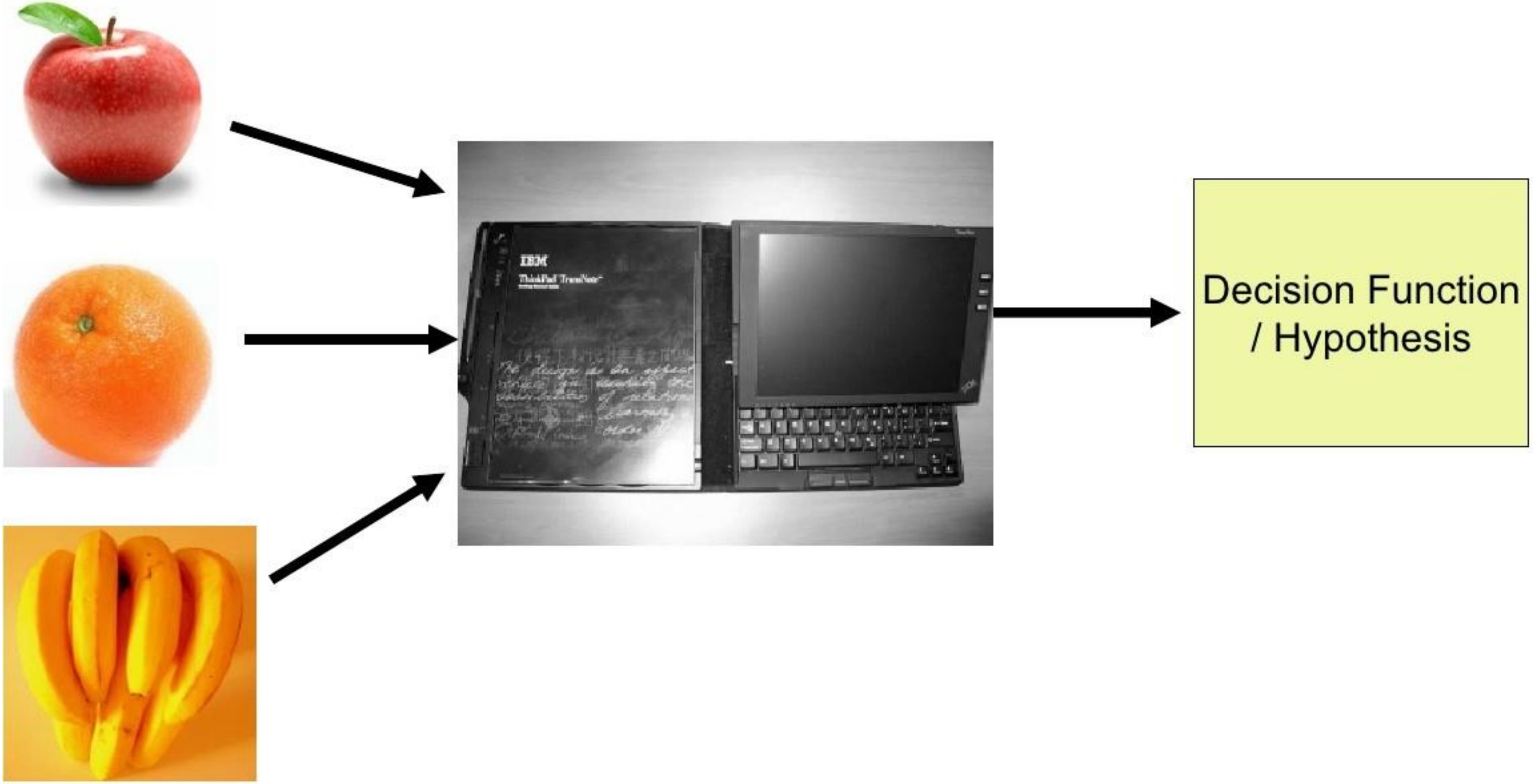


Binary Classification



Decision Function / Hypothesis

Multiclass Classification



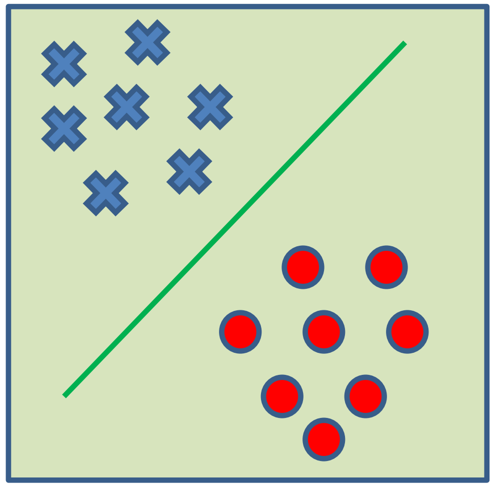
Decision Function / Hypothesis

Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

- Linear classifier



$$g(x_n) = sign(w^T x_n)$$

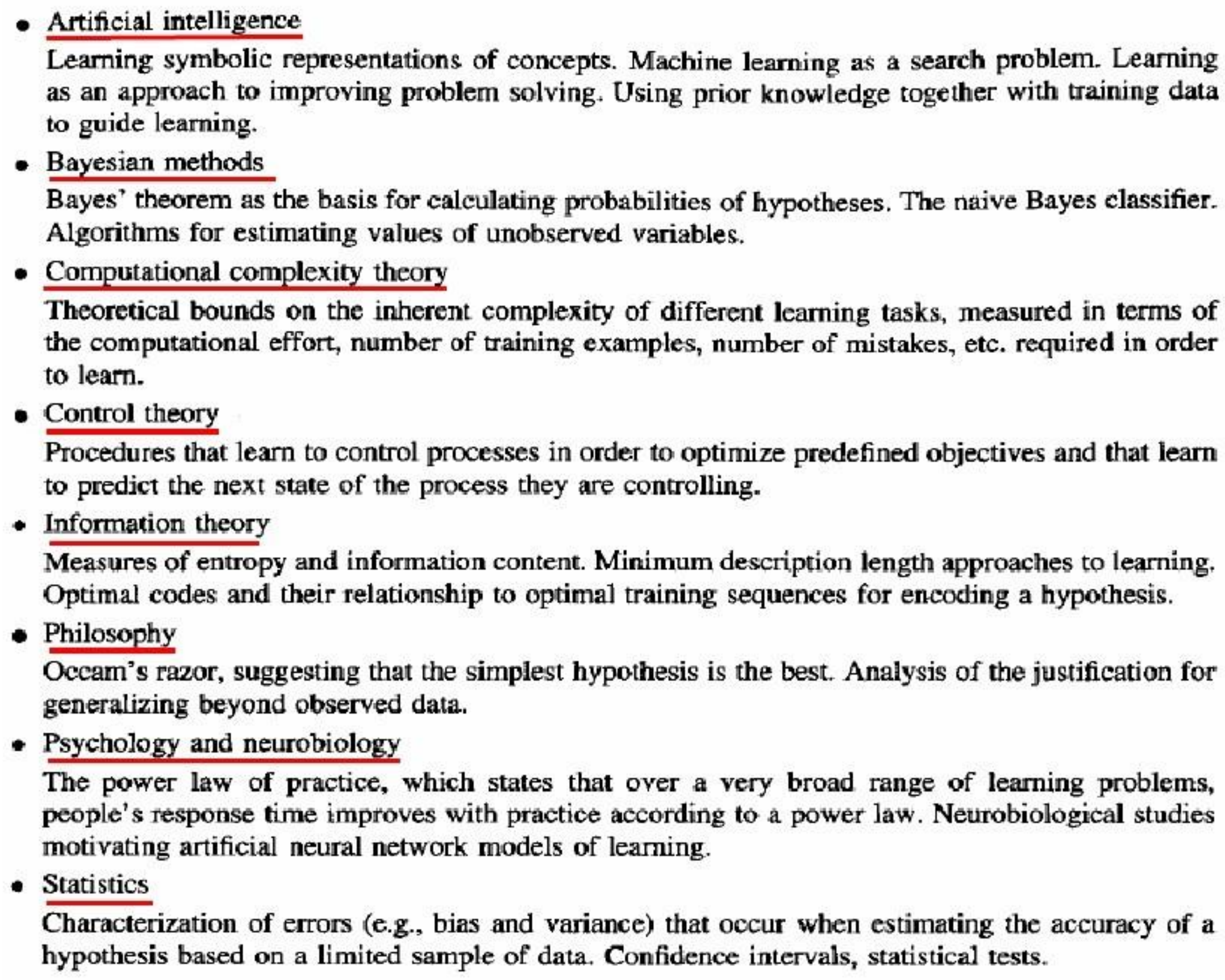
, where *w* is an *d*-dim vector (learned)

- Techniques:
  - Perceptron
  - Logistic regression
  - Support vector machine (SVM)
  - Ada-line
  - Multi-layer perceptron (MLP)

Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density Estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

Disciplines that contribute to development ML Field



- Artificial intelligence

  Learning symbolic representations of concepts. Machine learning as a search problem. Learning as an approach to improving problem solving. Using prior knowledge together with training data to guide learning.

- Bayesian methods

  Bayes' theorem as the basis for calculating probabilities of hypotheses. The naive Bayes classifier. Algorithms for estimating values of unobserved variables.

- Computational complexity theory

  Theoretical bounds on the inherent complexity of different learning tasks, measured in terms of the computational effort, number of training examples, number of mistakes, etc. required in order to learn.

- Control theory

  Procedures that learn to control processes in order to optimize predefined objectives and that learn to predict the next state of the process they are controlling.

- Information theory

  Measures of entropy and information content. Minimum description length approaches to learning. Optimal codes and their relationship to optimal training sequences for encoding a hypothesis.

- Philosophy

  Occam's razor, suggesting that the simplest hypothesis is the best. Analysis of the justification for generalizing beyond observed data.

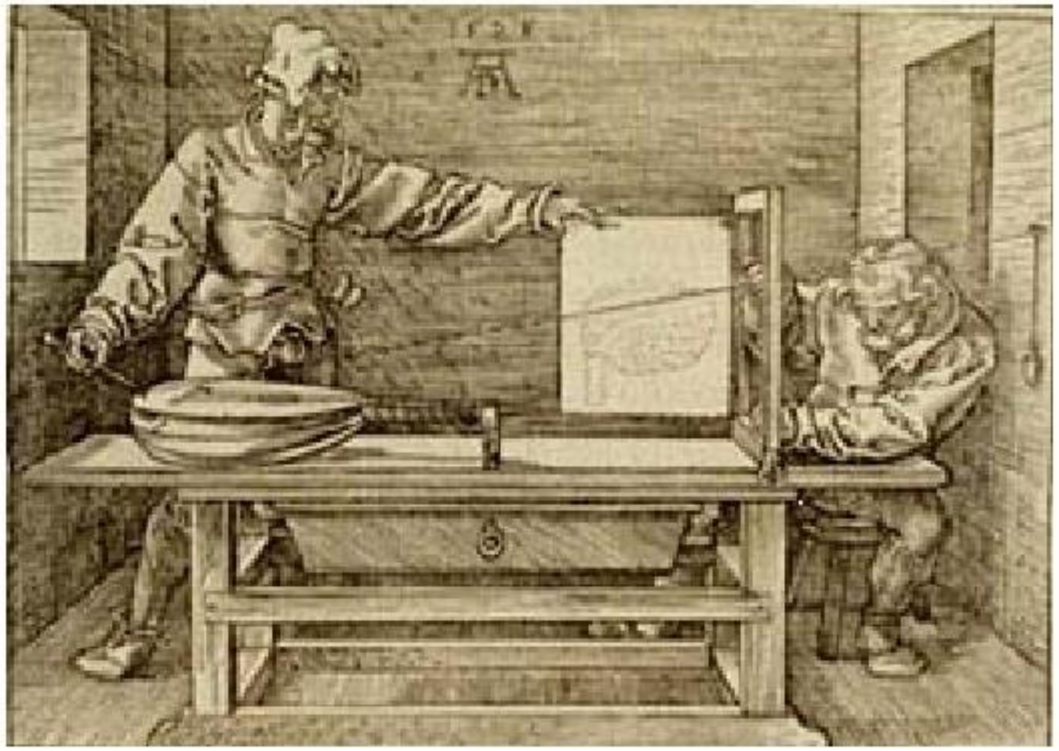
- Psychology and neurobiology

  The power law of practice, which states that over a very broad range of learning problems, people's response time improves with practice according to a power law. Neurobiological studies motivating artificial neural network models of learning.

- Statistics

  Characterization of errors (e.g., bias and variance) that occur when estimating the accuracy of a hypothesis based on a limited sample of data. Confidence intervals, statistical tests.

Designing a Learning System



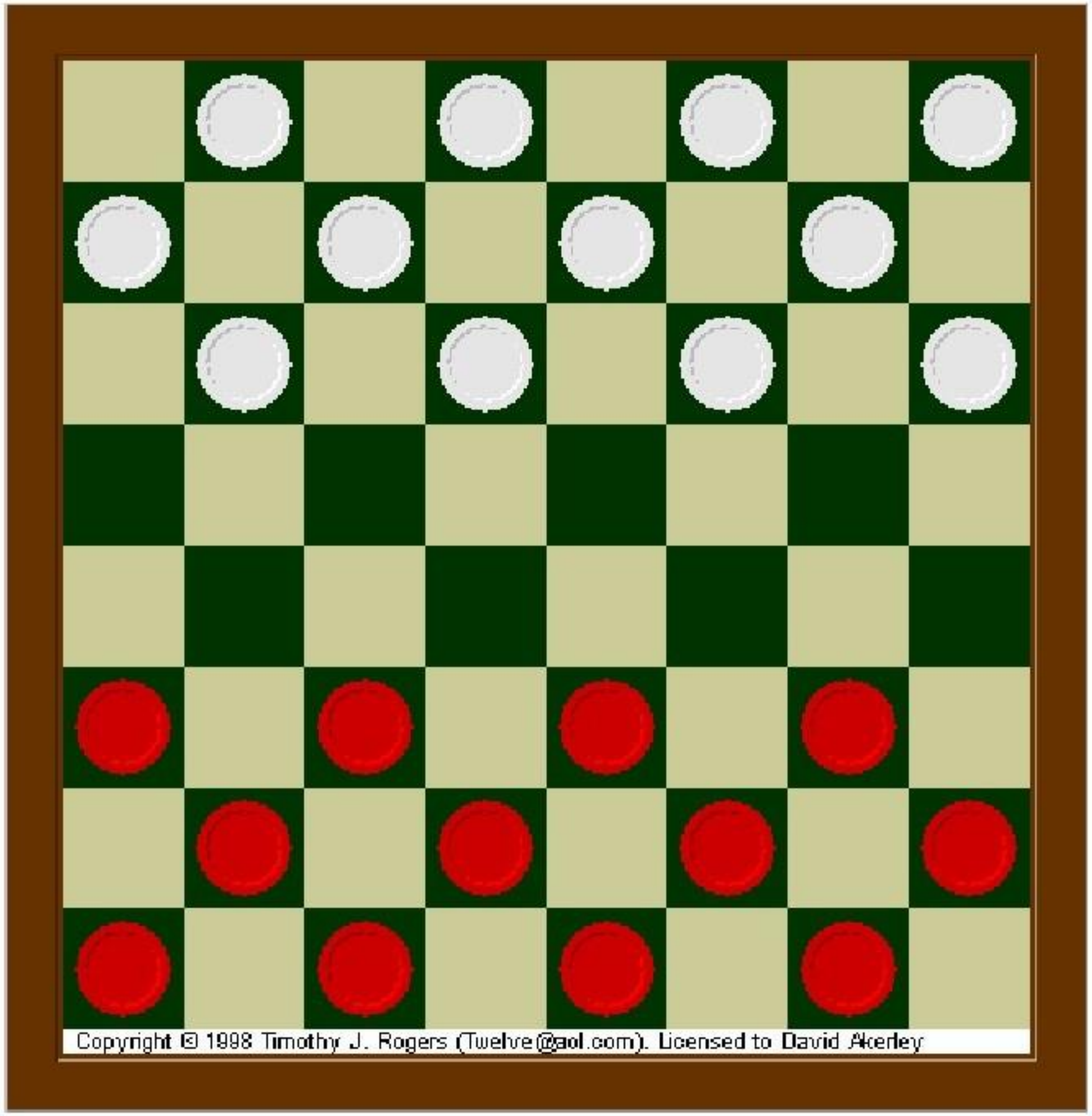
Checker Learning Problem

- Task **T** : playing checkers

- Performance measure **P**: % of game won against opponents

- Training experience **E** : playing practice game against itself

Checker



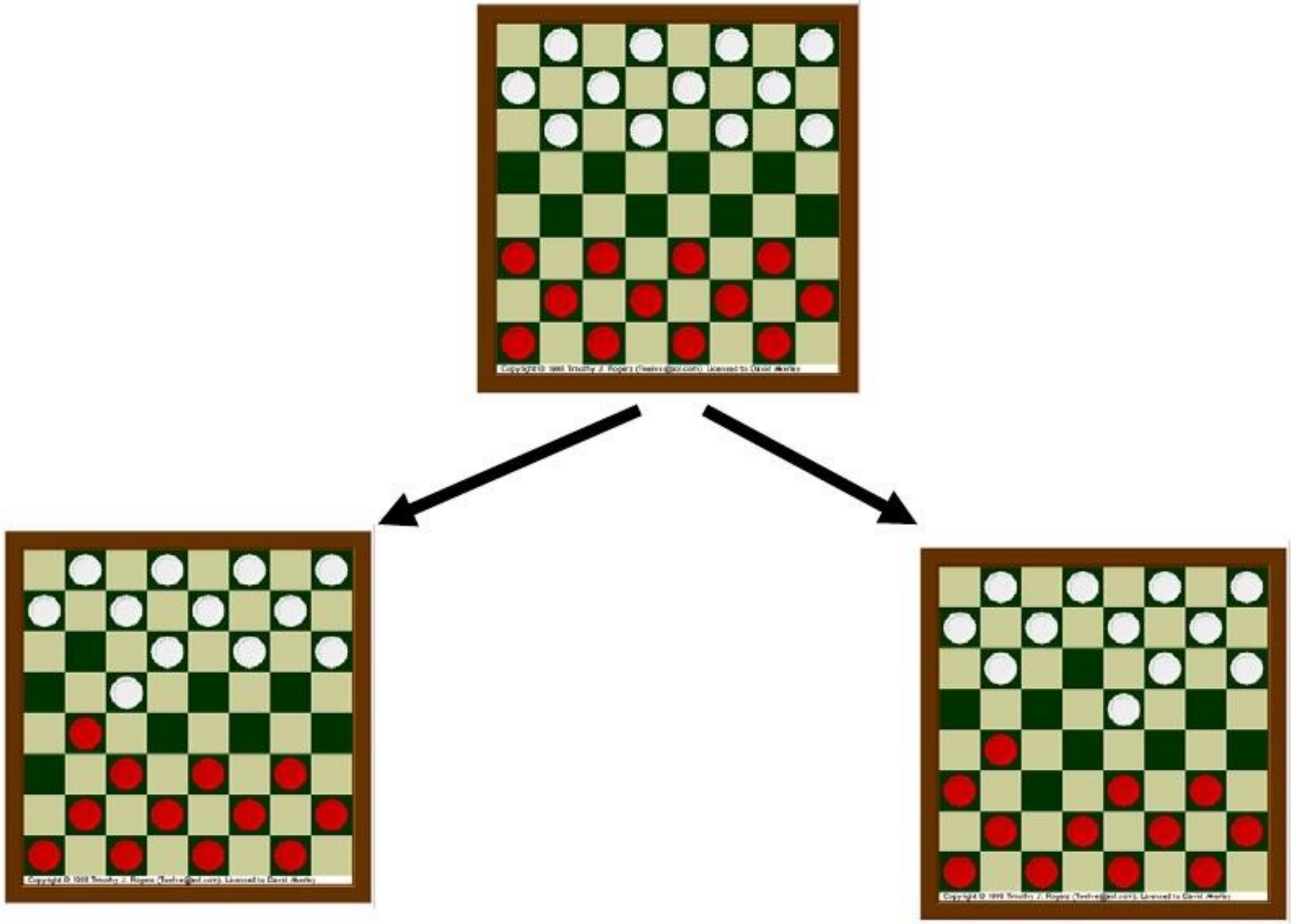
Choosing the Training Experience

The type of training experience **E** available to a system can have significant impact on success or failure of the learning system.

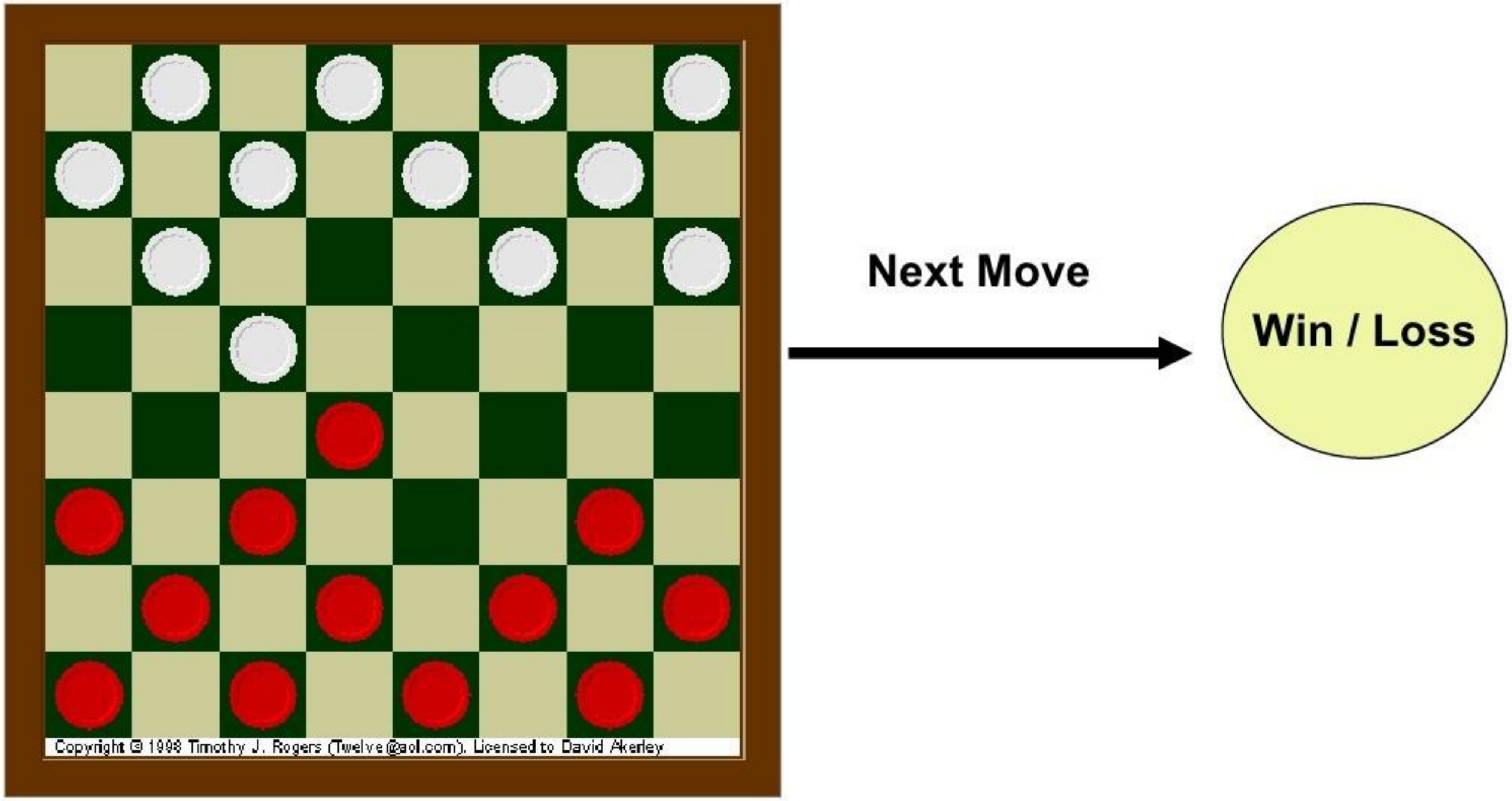
Choosing the Training Experience

One key attribute is whether the training experience provides **direct or indirect feedback regarding the choices made** by the performance system

Direct Learning



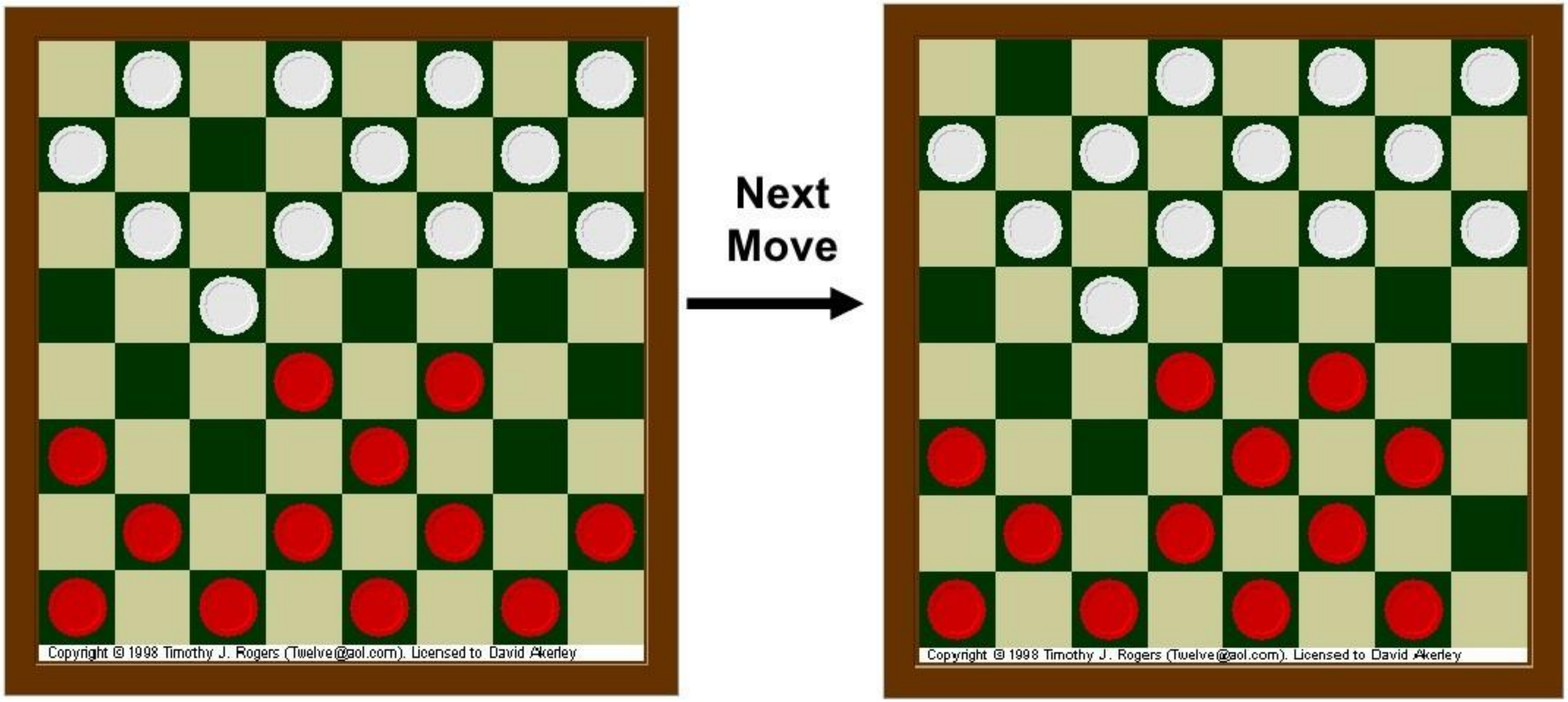
Indirect Learning


Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley

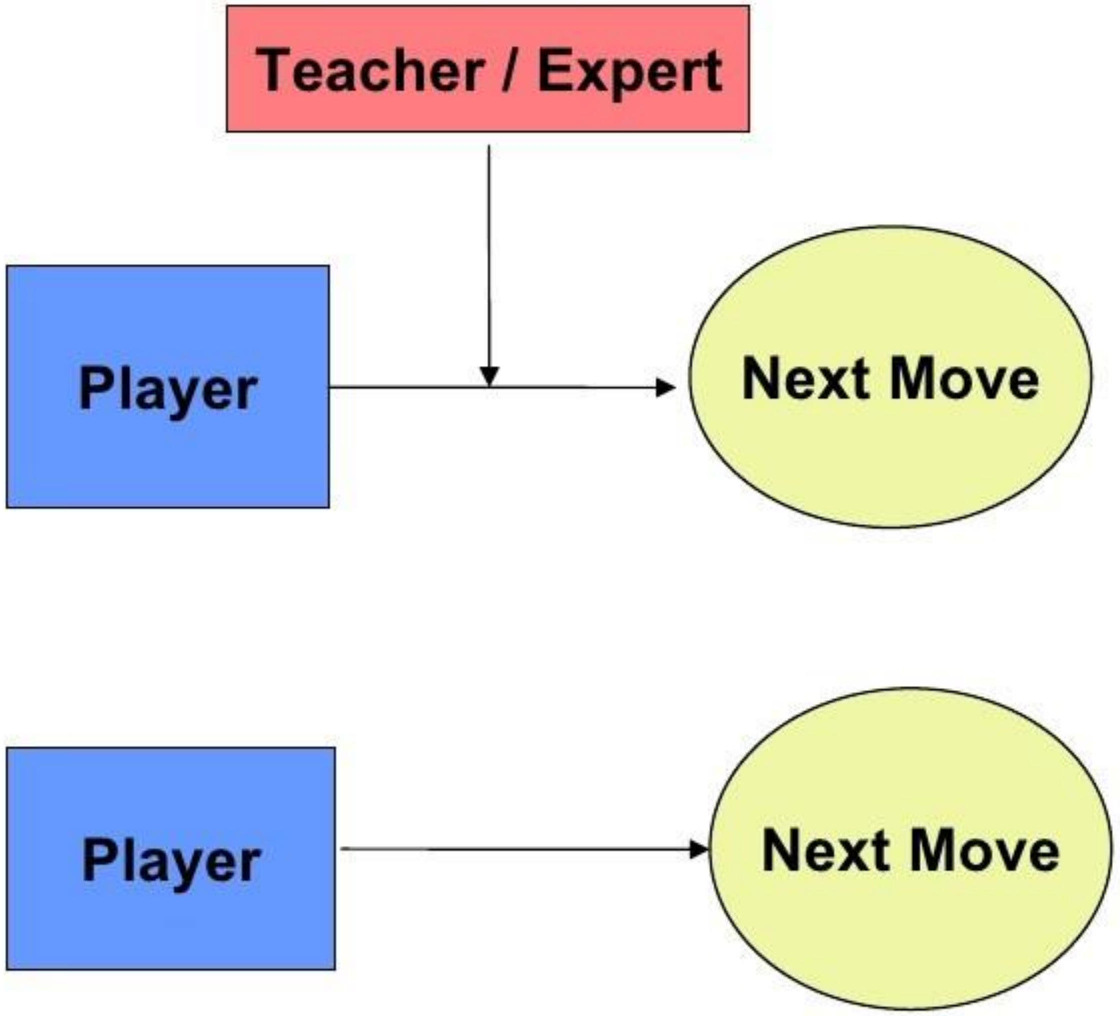
Next Move

Win / Loss

Second attribute is the **degree to which the learner controls the sequence of training examples**.

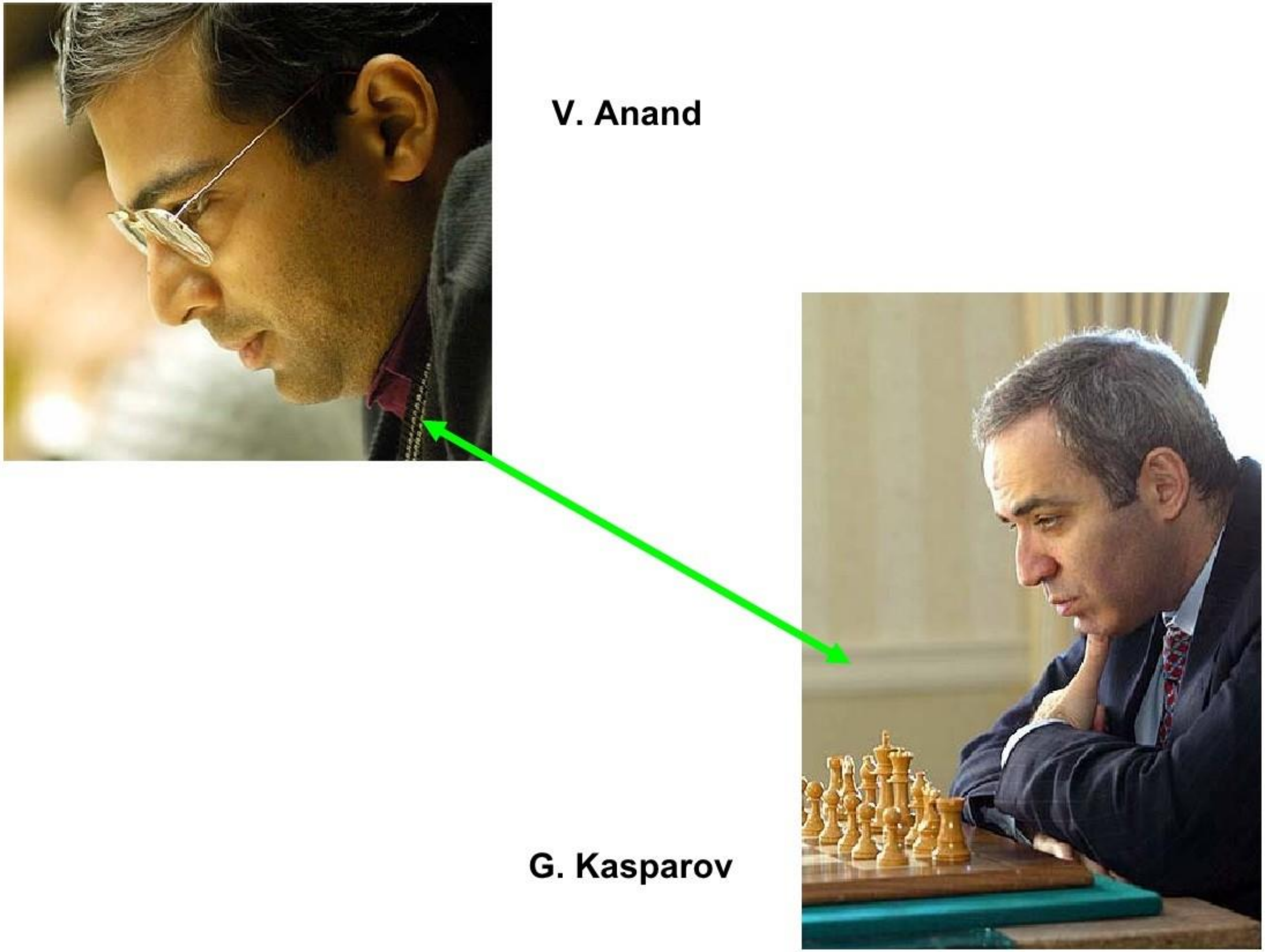

Next Move
Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley
Copyright © 1998 Timothy J. Rogers (Twelve@aol.com). Licensed to David Akerley

Another attribute is the **degree to which the learner controls the sequence of training examples**.



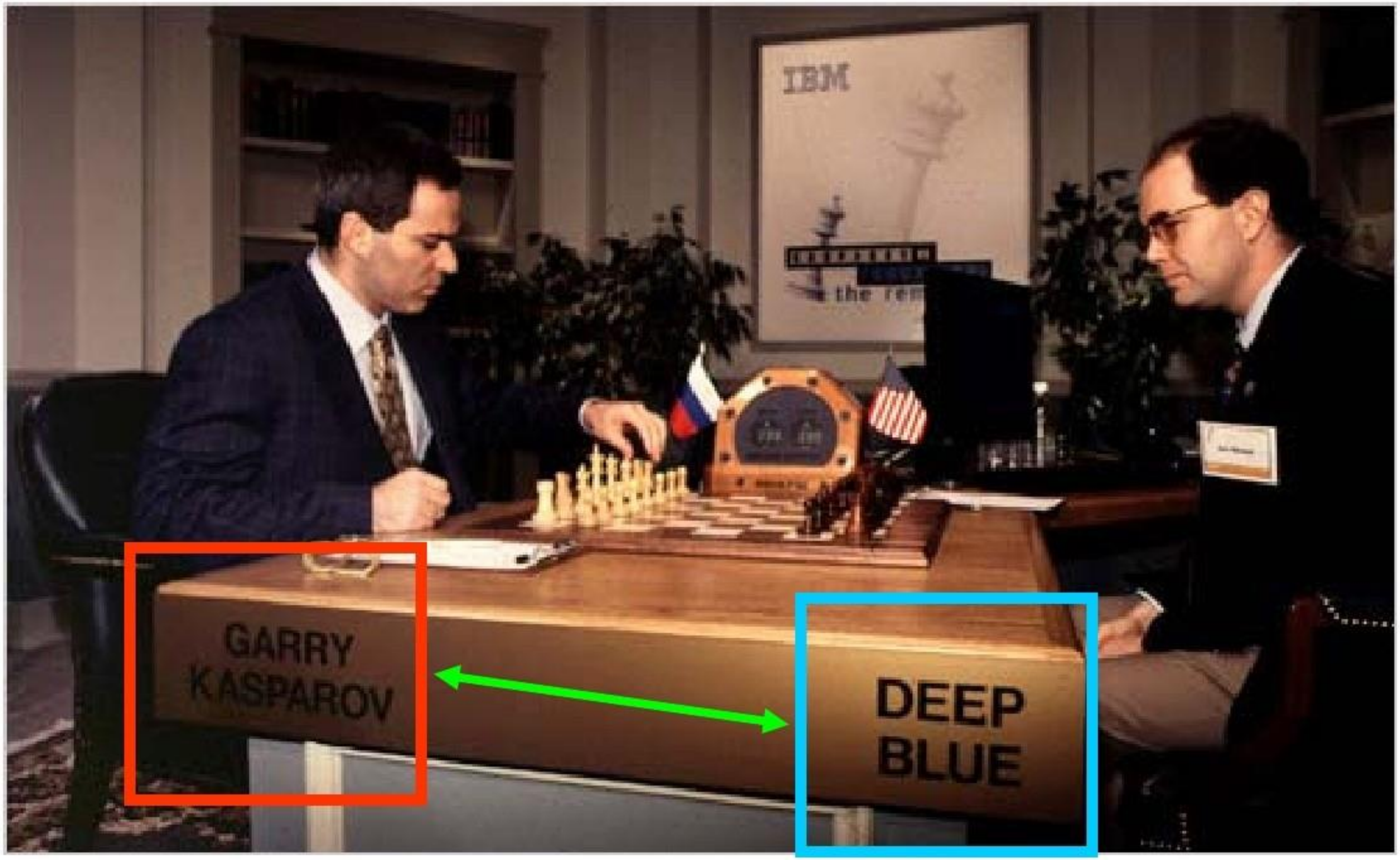
How well the training experience represents the distribution of examples over which the final system performance P must be measured.

(Learning will be most reliable when the training example follow a distribution similar to that of future test examples)



V. Anand

G. Kasparov



Garry Kasparov playing chess with IBM's Deep Blue.
Photo courtesy of IBM.

Kasparov

Monkey

PIXAR SHORTS

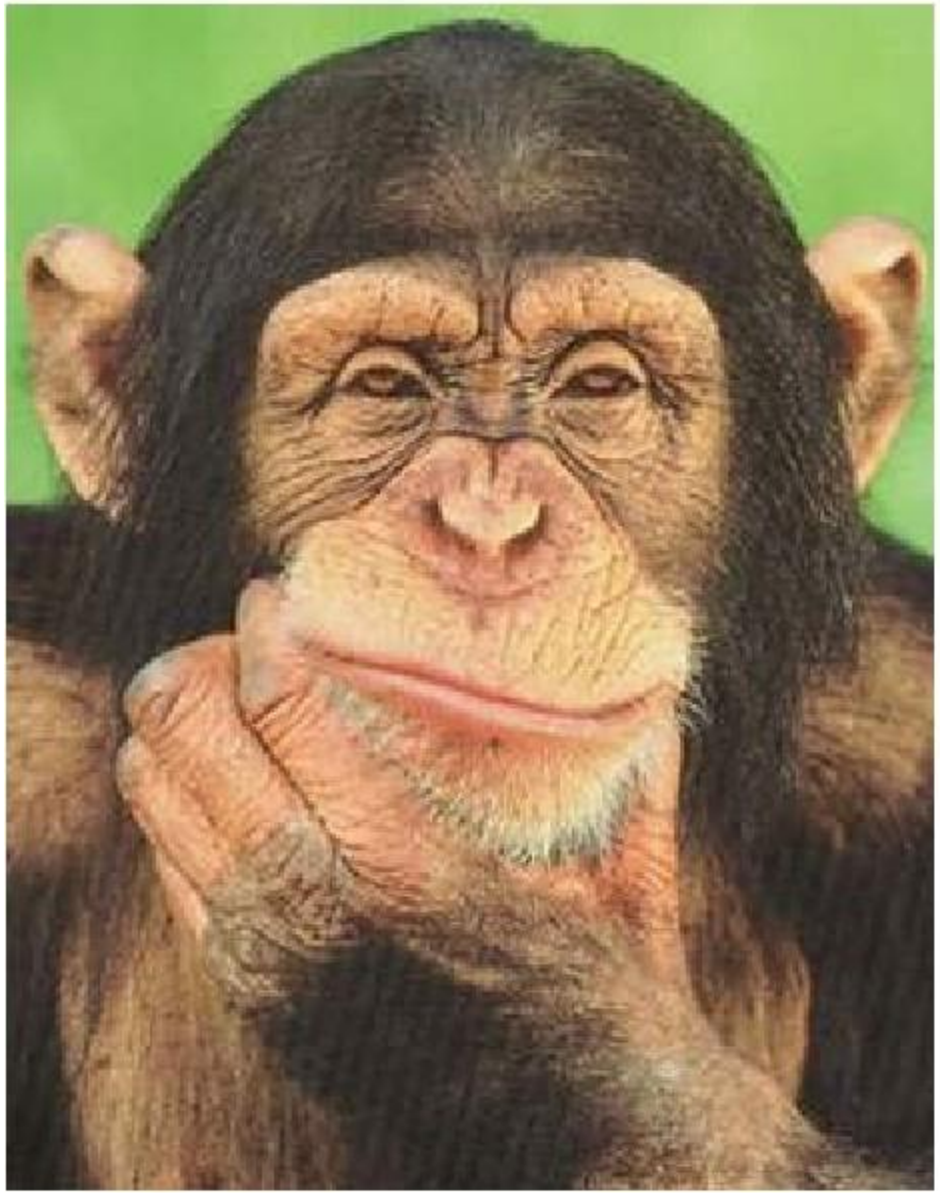
Geri's Game by Pixar in which an elderly man enjoys a game of chess with himself (*available at http://www.pixar.com/shorts/gg/theater/index.html*)

Won an Academy Award for best animated short film.

Assumptions

- Let us assume that our system will train by playing games against itself.
- And it is allowed to generate as much training data as time permits.

Issues Related to Experience

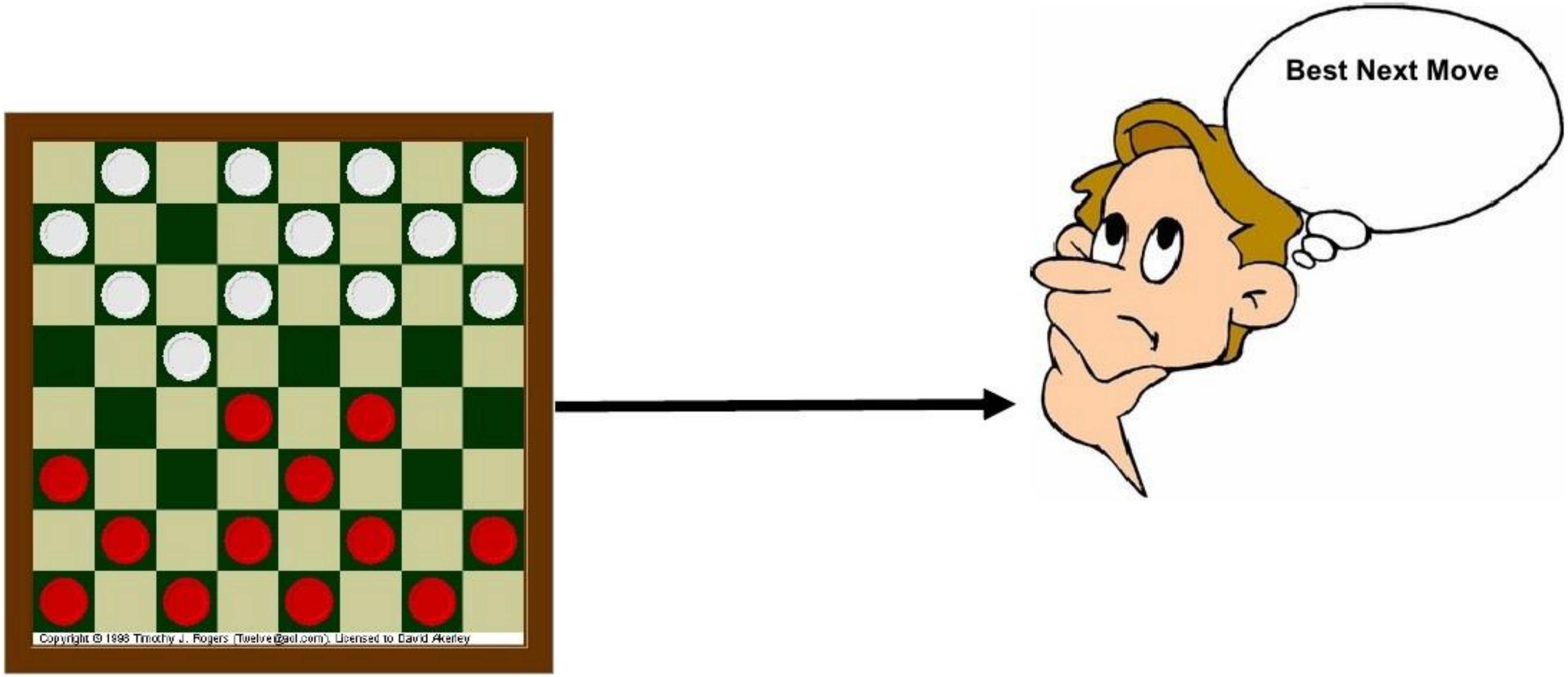


What type of knowledge/experience should one learn??

How to represent the experience ? (some mathematical representation for experience)

What should be the learning mechanism???

Main objective of the checker playing

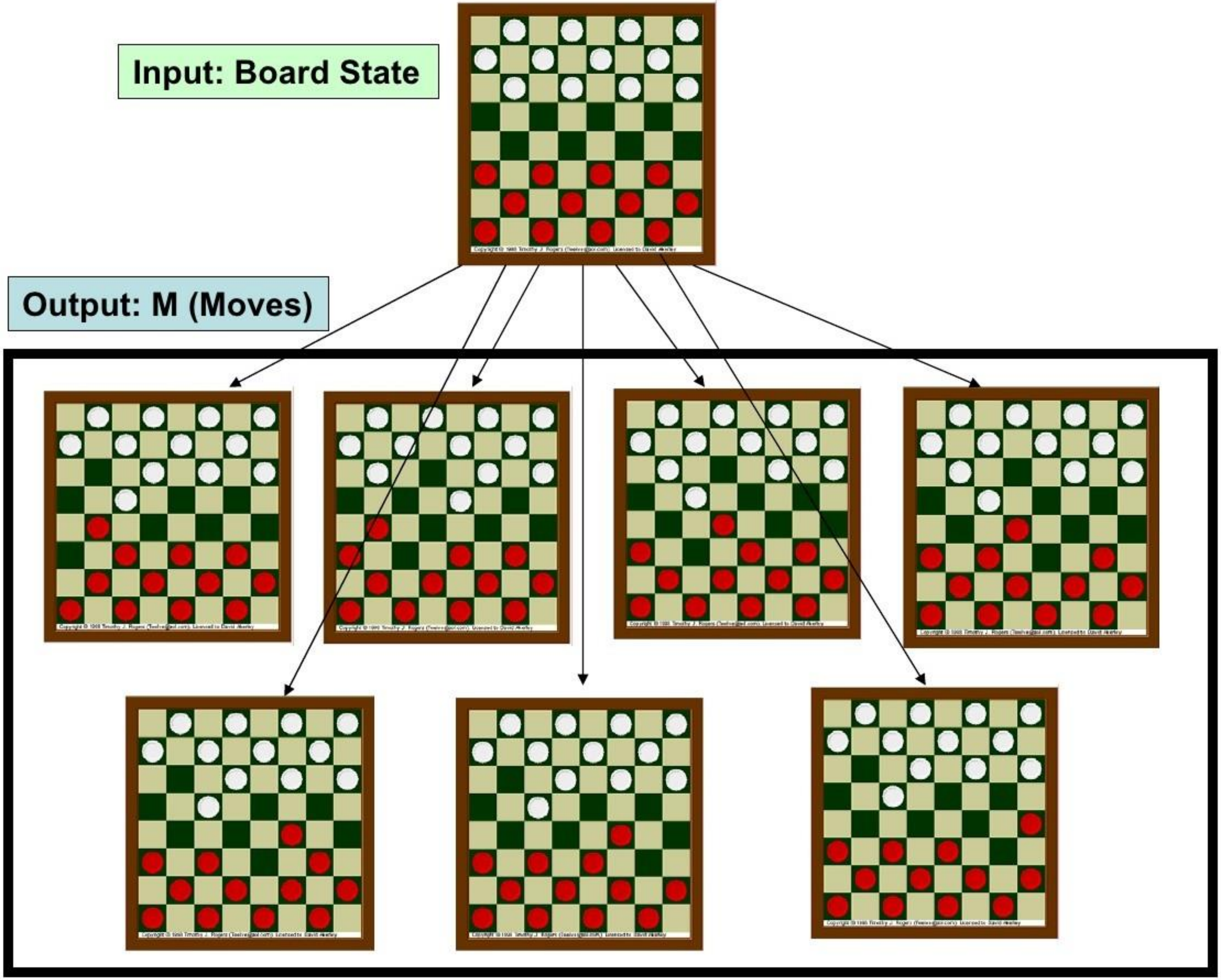

Copyright © 1998 Timothy J. Rogers (Twelve@aol.com); Licensed to David Akerley
Best Next Move

Target Function

Choose Move: B → M

Choose Move is a function

where input **B** is the set of legal board states and produces **M** which is the set of legal moves



Input: Board State

Output: M (Moves)

Target Function

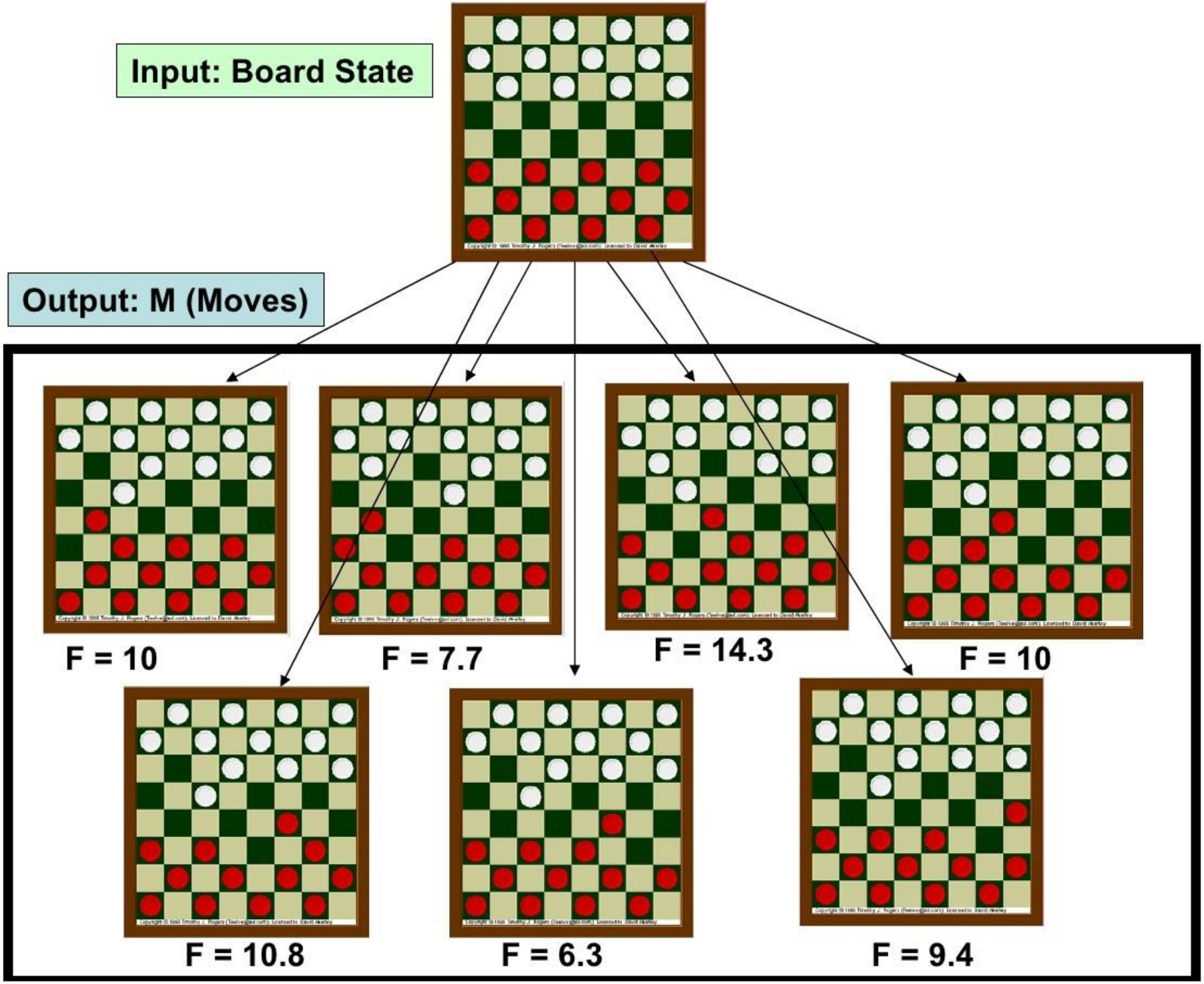
Choose Move: B → M

M = *Choose Move* (B)

Alternative Target Function

$$F : \mathrm{B} \rightarrow \mathrm{R}$$

F is the target function,
Input B is the board state and
Output R denotes a set of real number



Input: Board State

Output: M (Moves)

F = 10

F = 7.7

F = 14.3

F = 10

F = 10.8

F = 6.3

F = 9.4

Representation of Target Function

- ***xl:*** the number of white pieces on the board
- ***x2:*** the number of red pieces on the board
- ***x3:*** the number of white kings on the board
- **x4:** the number of red kings on the board
- ***x5:*** the number of white pieces threatened by red (i.e., which can be captured on red's next turn)
- ***X6:*** the number of red pieces threatened by white

$$F'(b) = w0 + w1x1+ w2x2 + w3x3 + w4x4 + w5x5 + w6x6$$

- Task T: playing checkers
- Performance measure ***P: %*** of games won in the world tournament
- Training experience E: games played against itself
- Target function: *F* : Board → R
- Target function representation

$$F'(b) = w0 + w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + w6x6$$

The problem of learning a checkers strategy reduces to the problem of learning values for the coefficients *w0* through *w6* in the target function representation

Adjustment of Weights

$$E(Error) \equiv \sum_{<b, F_{train(b)}> \in \, training \;\; examples} (\mathrm{F}_{\mathrm{train}}(b) - F'(b))^2$$

Generic issues of Machine Learning

- What algorithms exist for learning general target functions from specific training examples?
- In what settings will particular algorithms converge to the desired function, given sufficient training data?
- Which algorithms perform best for which types of problems and representations?
- How much training data is sufficient?
- What is the best way to reduce the learning task to one or more function approximation problems?

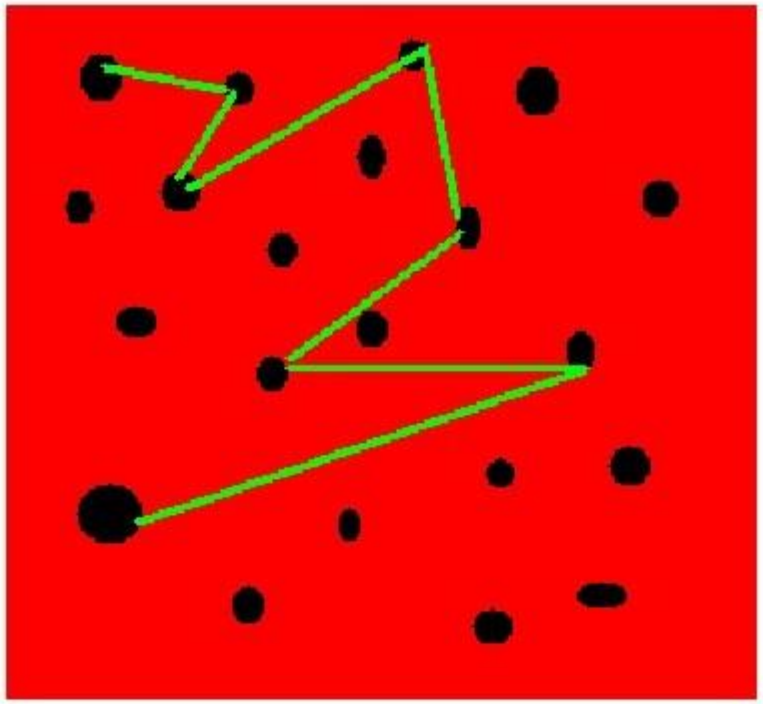
Generic issues of Machine Learning

- When and how can prior knowledge held by the learner guide the process of generalizing from examples?
- Can prior knowledge be helpful even when it is only approximately correct?
- What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
- How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

Concept Learning

Concept Learning

A task of acquiring a potential hypothesis (solution) that best fits the training examples

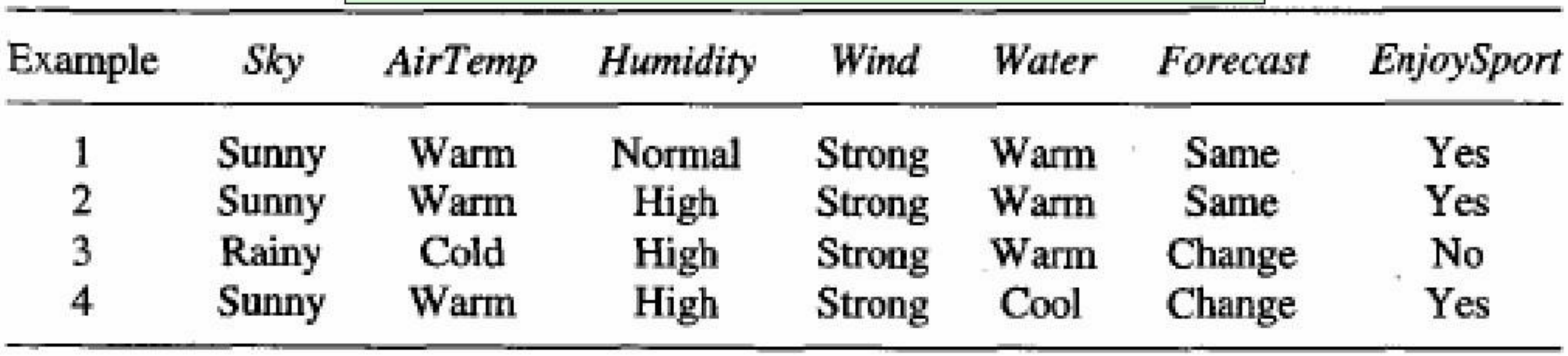


Concept Learning Task

Objective is to learn EnjoySport

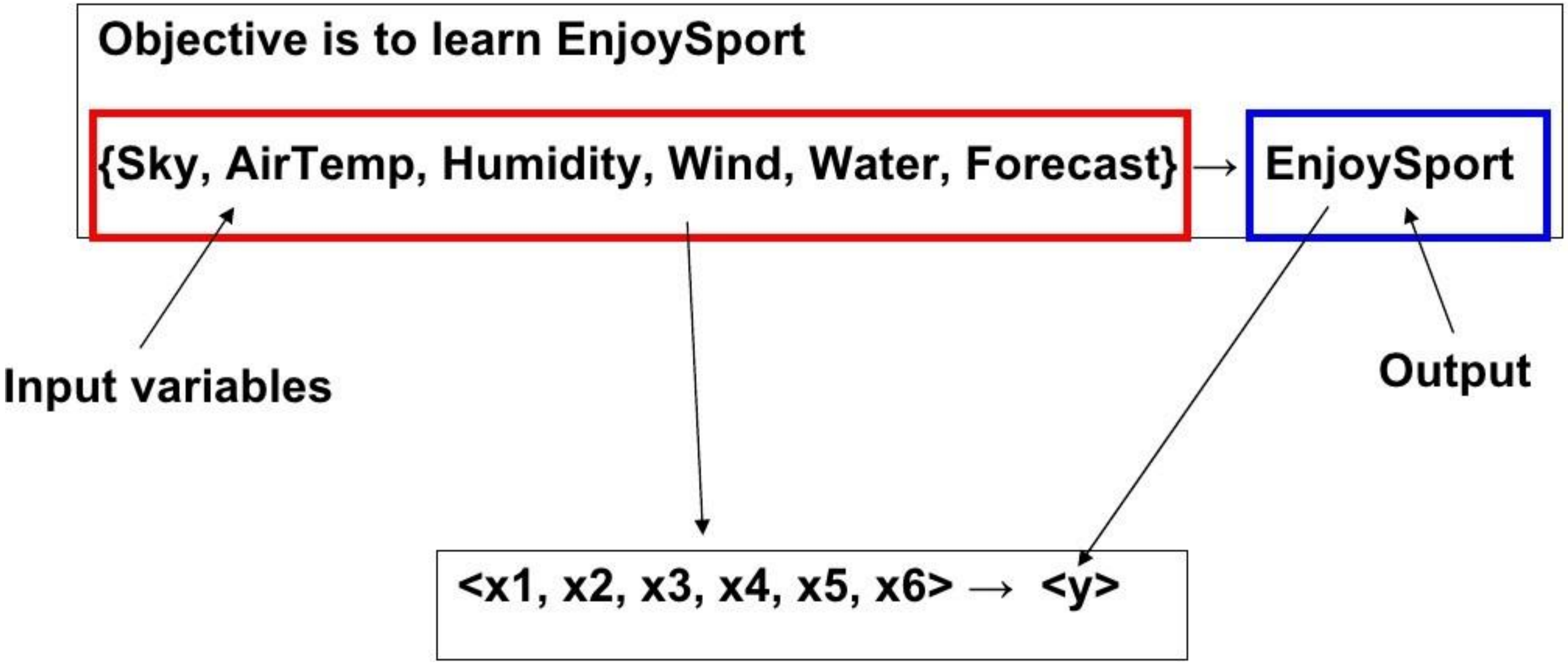
{Sky, AirTemp, Humidity, Wind, Water, Forecast} → EnjoySport

Tom enjoys his favorite water sports



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Concept Learning Task



Objective is to learn EnjoySport

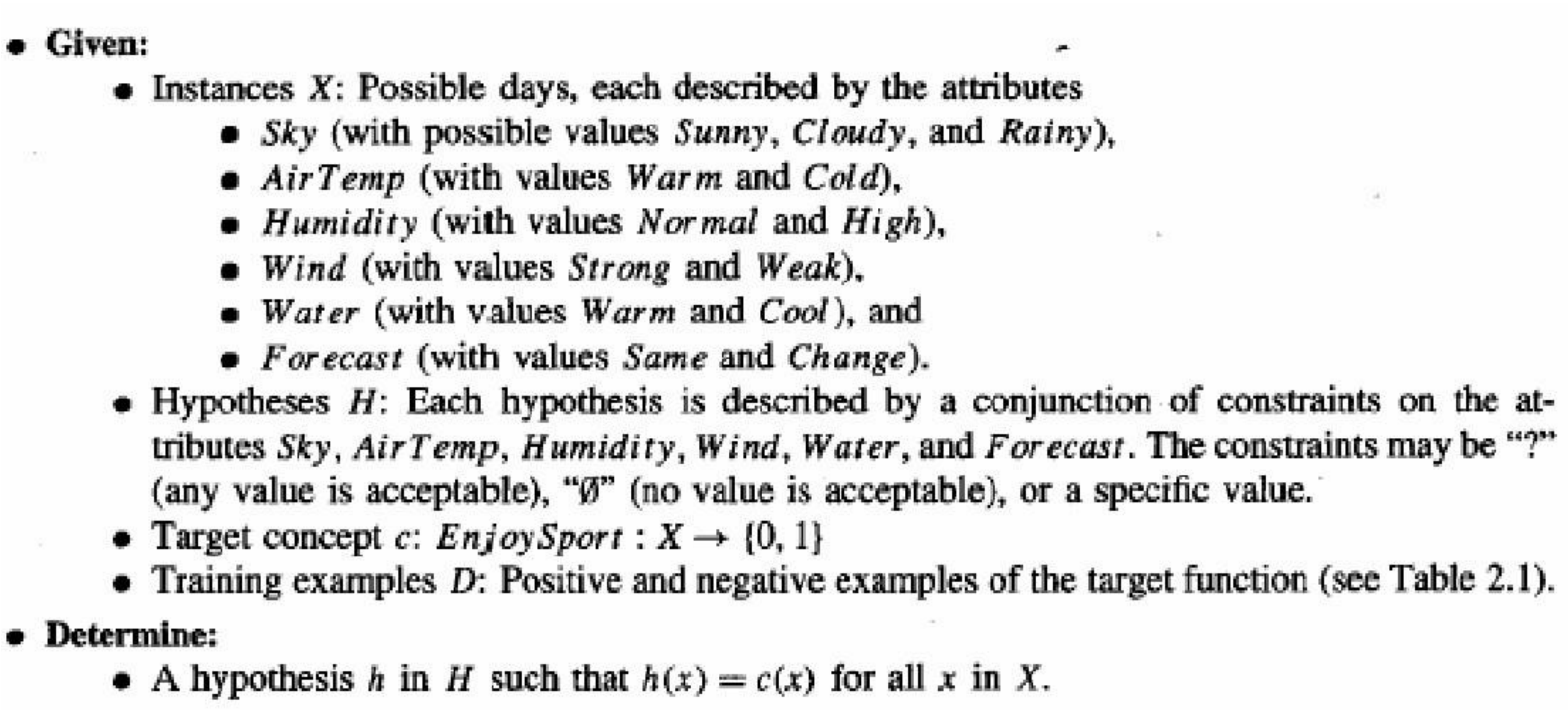
{Sky, AirTemp, Humidity, Wind, Water, Forecast} → EnjoySport

Input variables

Output

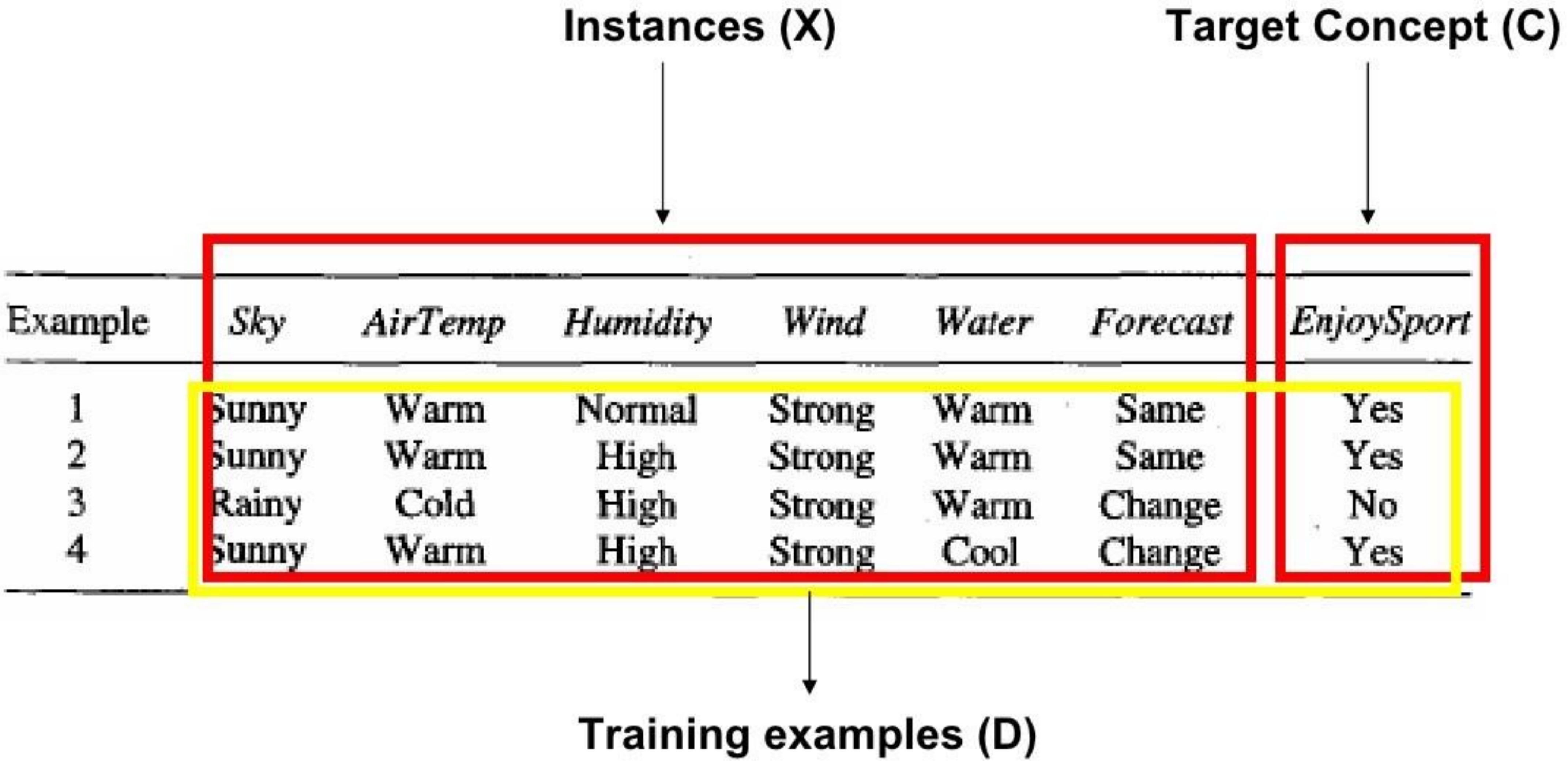
<x1, x2, x3, x4, x5, x6> → <y>

Notations



- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

Notations



Instances (X)

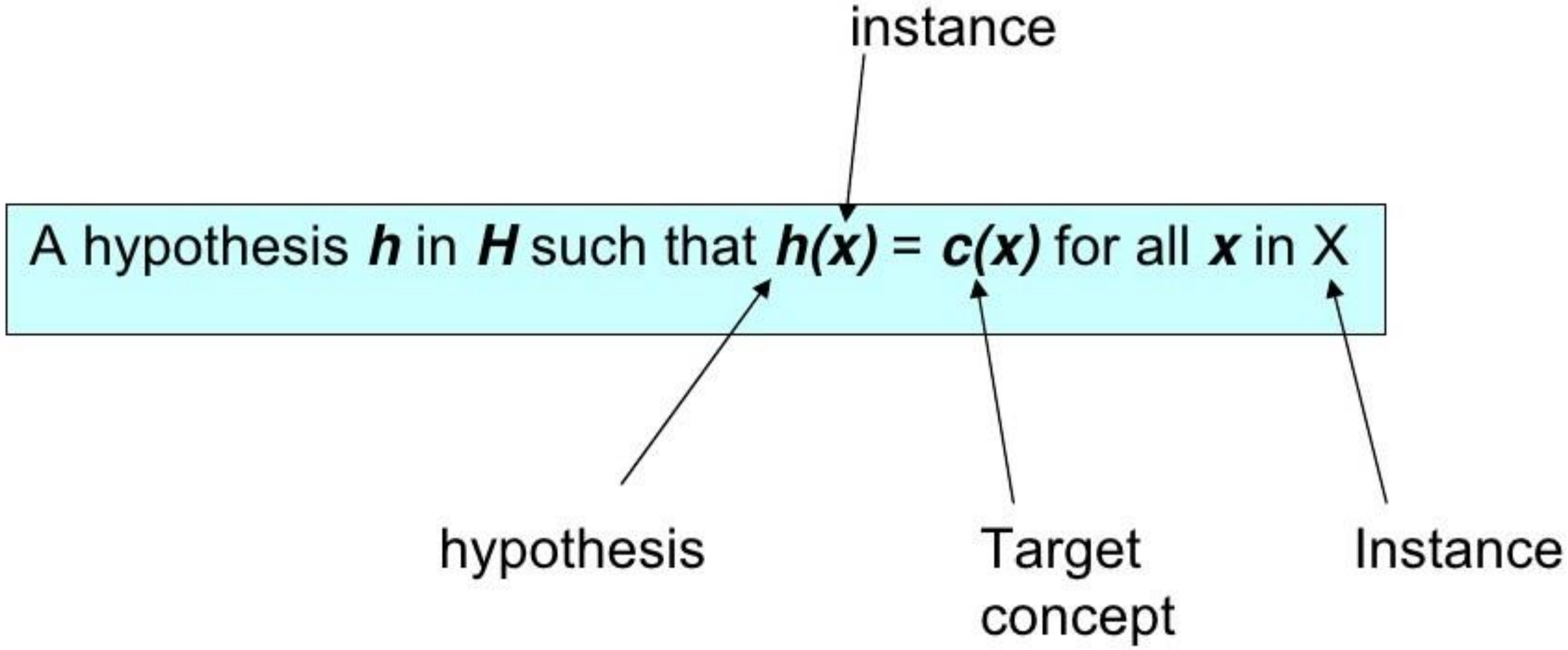
Target Concept (C)

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Training examples (D)

Concept Learning

Acquiring the definition of a general category from a given set of positive and negative training examples of the category.



instance

A hypothesis ***h*** in ***H*** such that ***h(x)*** = ***c(x)*** for all ***x*** in X

hypothesis

Target concept

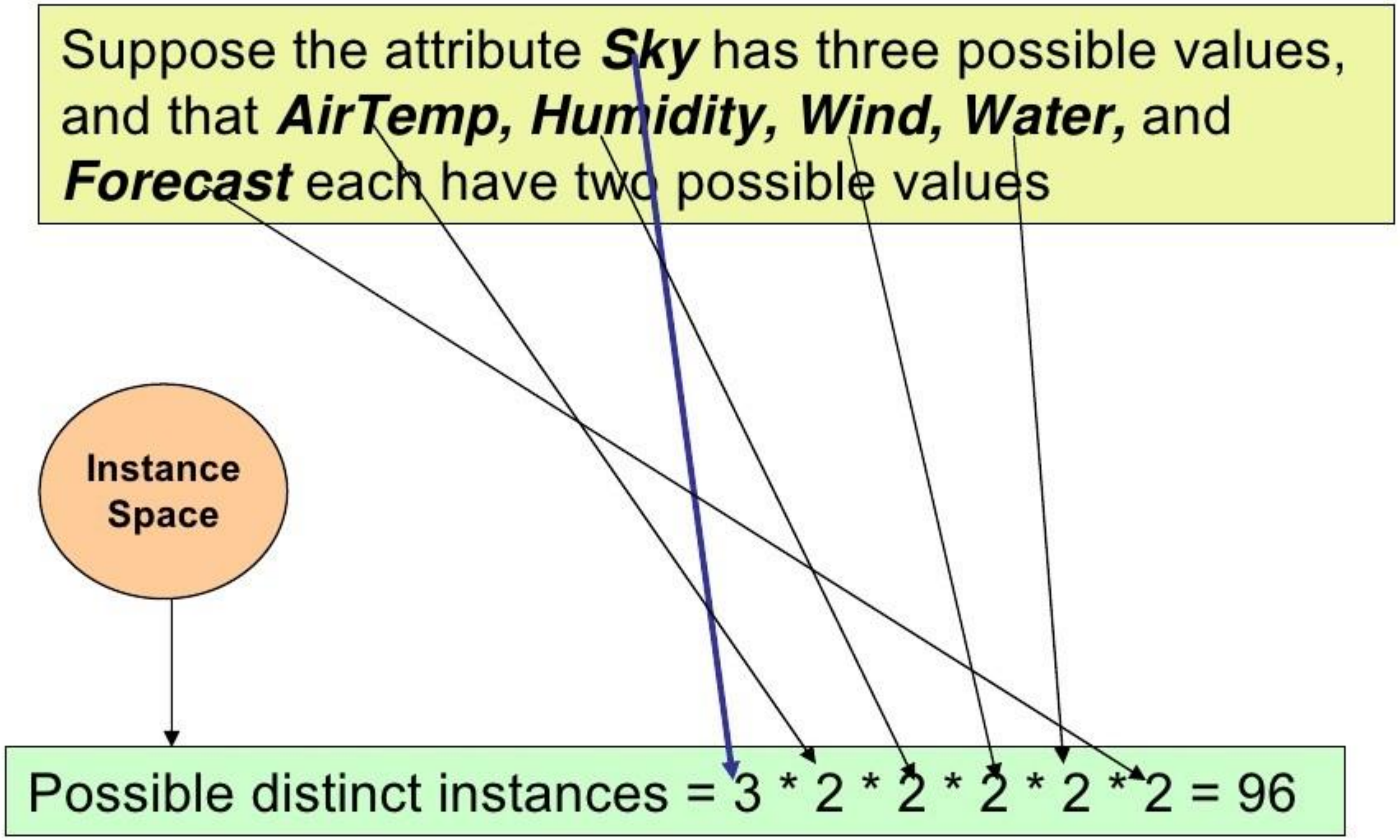
Instance

Instance Space

Suppose that the target hypothesis that we want to learn for the current problem is represented as a conjunction of all the attributes

Sky = ... AND AirTemp = …. AND Humidity= …. AND Wind = …. AND Water= …. AND Forecast= …. THEN EnjoySport= ….

Instance Space



Suppose the attribute ***Sky*** has three possible values, and that ***AirTemp, Humidity, Wind, Water,*** and ***Forecast*** each have two possible values

Instance Space

Possible distinct instances = 3 * 2 * 2 * 2 * 2 * 2 = 96

Hypothesis Space

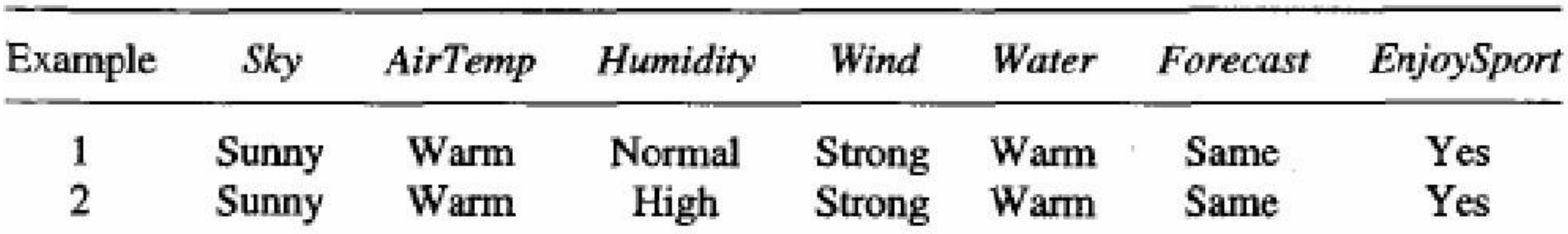
Hypothesis Space: A set of all possible hypotheses

Possible syntactically distinct Hypotheses for *EnjoySport*

= 5 * 4 * 4 * 4 * 4 * 4 = 5120

- **Sky has three possible values**
- **Fourth value don't care (?)**
- **Fifth value is empty set Ø**

Hypothesis Space

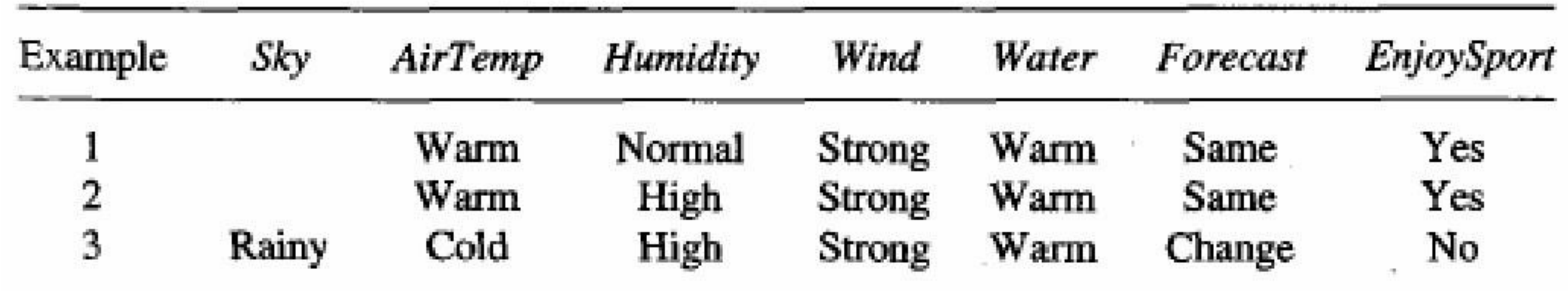


Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes

h = <Sunny, Warm, ?, Strong, Warm, Same>

Normal / High

Hypothesis Space



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1		Warm	Normal	Strong	Warm	Same	Yes
2		Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No

h = <Ø, Warm, ?, Strong, Warm, Same>

Concept Learning as Search

Concept learning can be viewed as the task of searching through a large space of hypothesis implicitly defined by the hypothesis representation.

The goal of the concept learning search is to find the hypothesis that best fits the training examples.

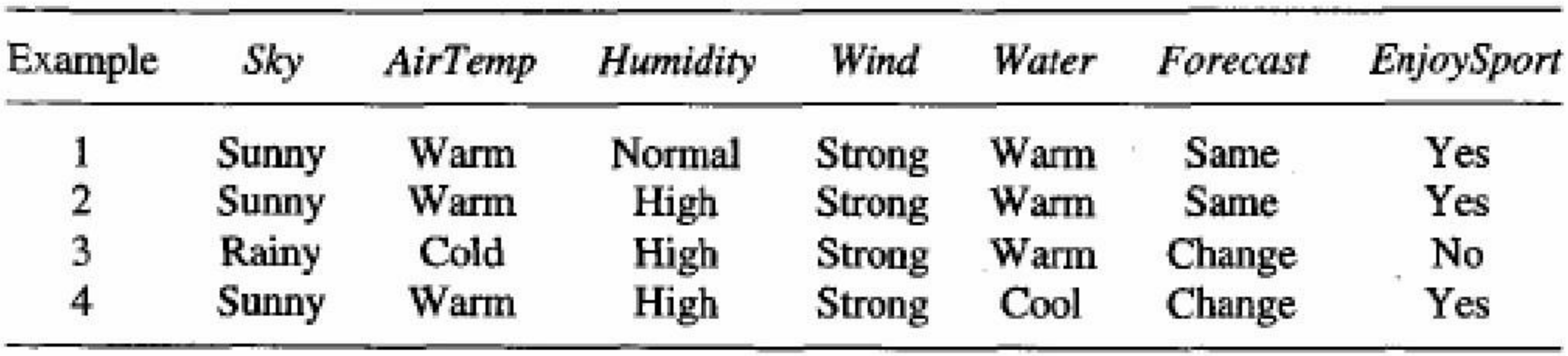
General-to-Specific Learning

Every day Tom his enjoy i.e., Only positive examples.

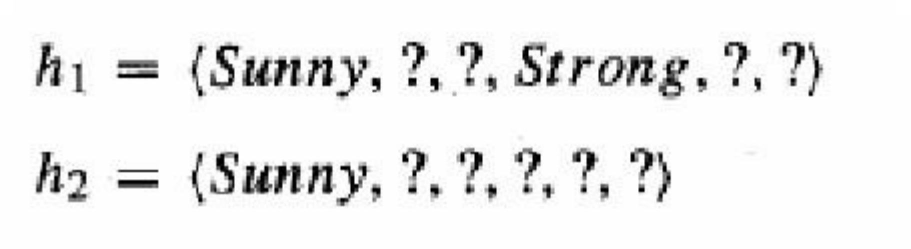
Most General Hypothesis: h = <?, ?, ?, ?, ?, ?>

Most Specific Hypothesis: h = < **ø, ø, ø, ø, ø, ø**>

General-to-Specific Learning



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



$$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$$

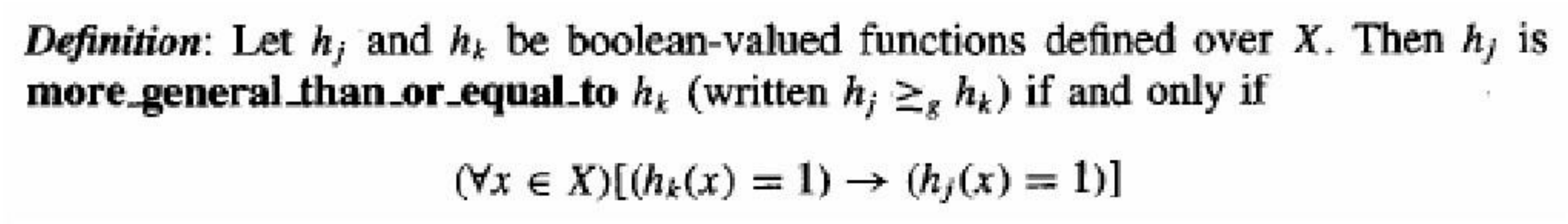
$$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$$

Consider the sets of instances that are classified positive by hl and by h2. **h2** imposes fewer constraints on the instance, thus it classifies more instances as positive.

Any instance classified positive by hl will also be classified positive by h2. Therefore, we say that h2 is more general than hl.

Definition

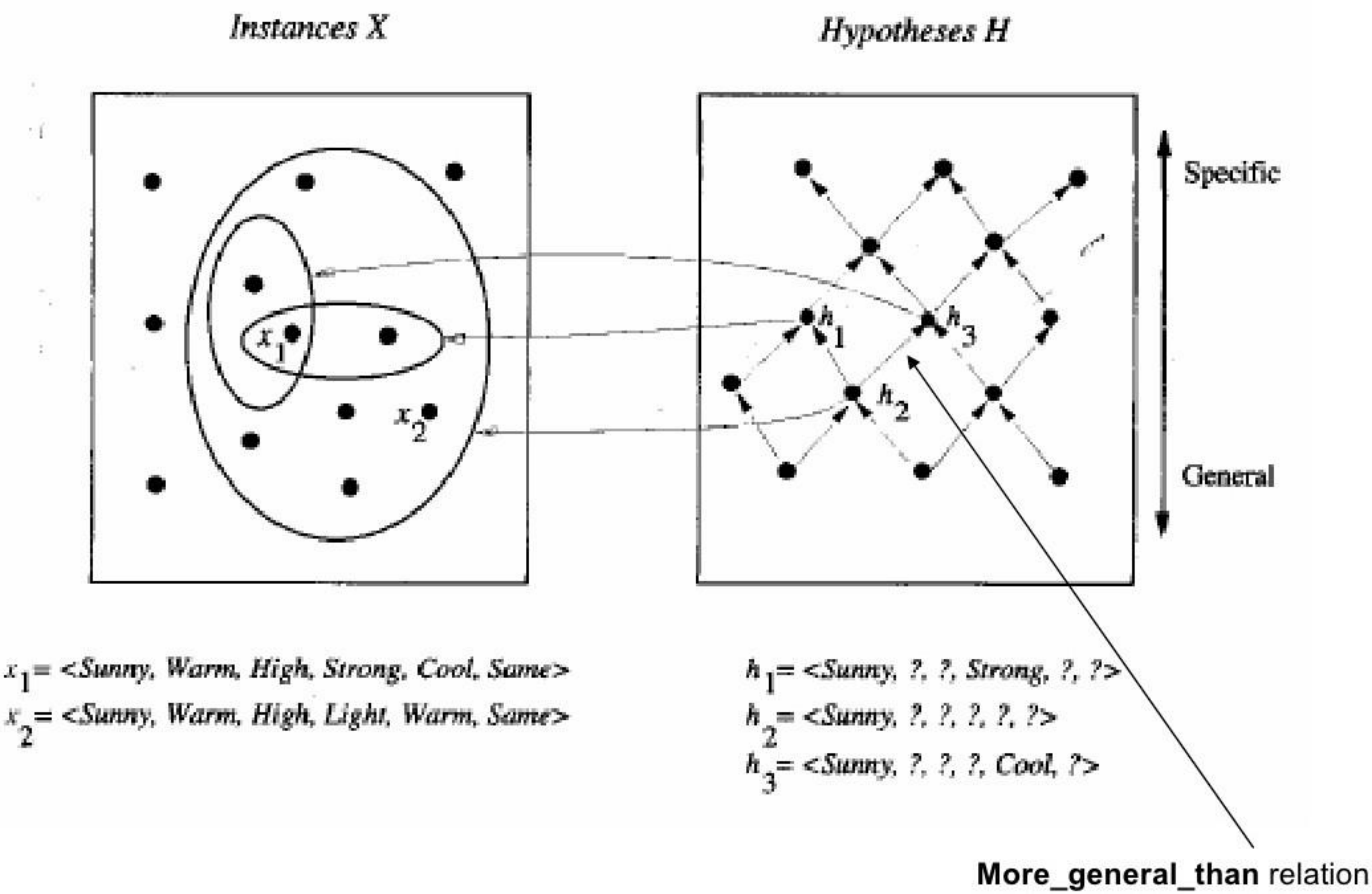
Given hypotheses **hj** and **hk**, **hj** is **more_general_than_or_equal_to hk** if and only if any instance that satisfies **hk** also satisfies ***hj.***



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 $h_j \geq_g h_k$) if and only if

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

We can also say that ***hj*** is ***more_specific_than hk*** when ***hk*** is ***more_general_than hj.***



x_1= <Sunny, Warm, High, Strong, Cool, Same>

x_2= <Sunny, Warm, High, Light, Warm, Same>

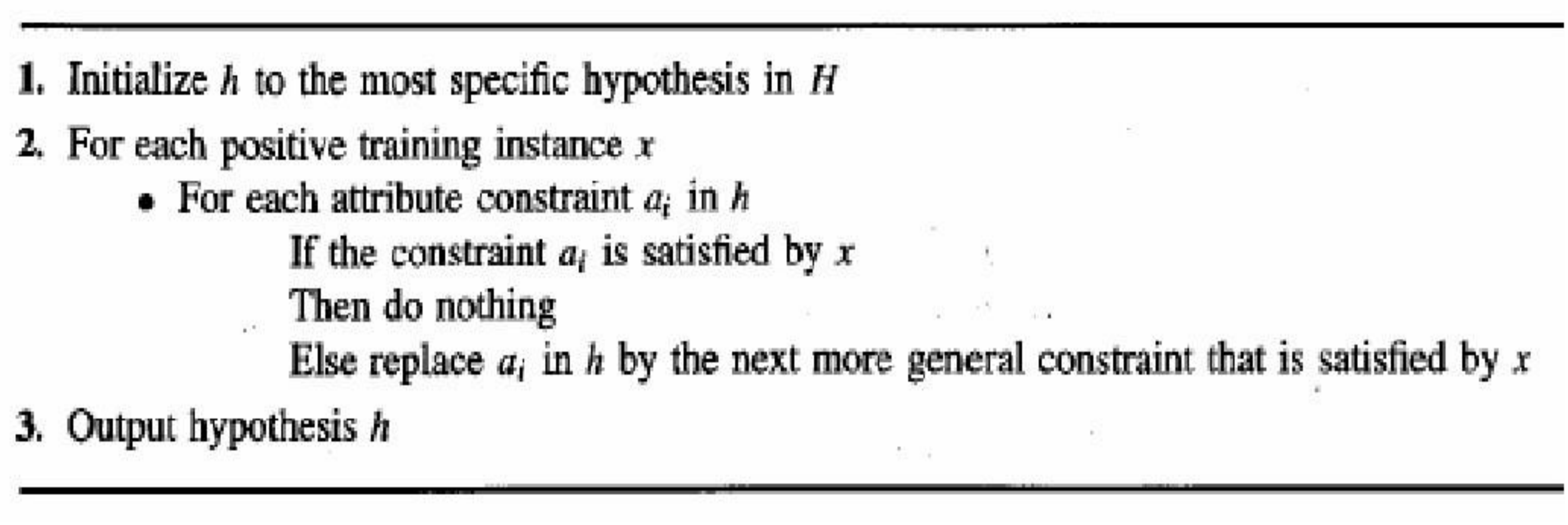
h_1= <Sunny, ?, ?, Strong, ?, ?>

h_2= <Sunny, ?, ?, ?, ?, ?>

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

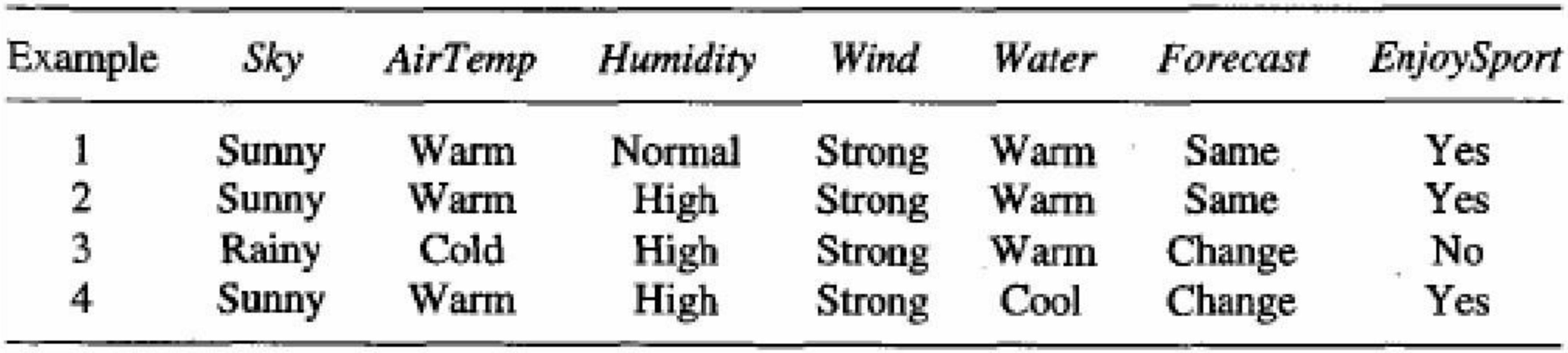
More_general_than relation

FIND-S: Finding a Maximally Specific Hypothesis

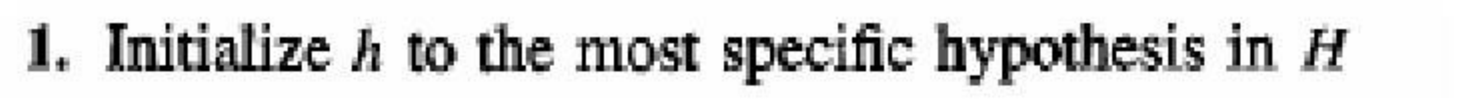


```
1. Initialize $h$ to the most specific hypothesis in $H$
2. For each positive training instance $x$
     • For each attribute constraint $a_i$ in $h$
           If the constraint $a_i$ is satisfied by $x$
           Then do nothing
           Else replace $a_i$ in $h$ by the next more general constraint that is satisfied by $x$
3. Output hypothesis $h$
```

Step 1: FIND-S



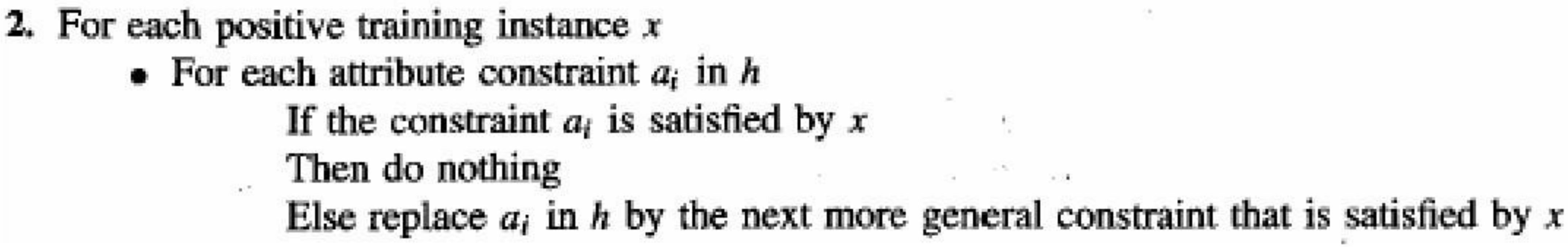
Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



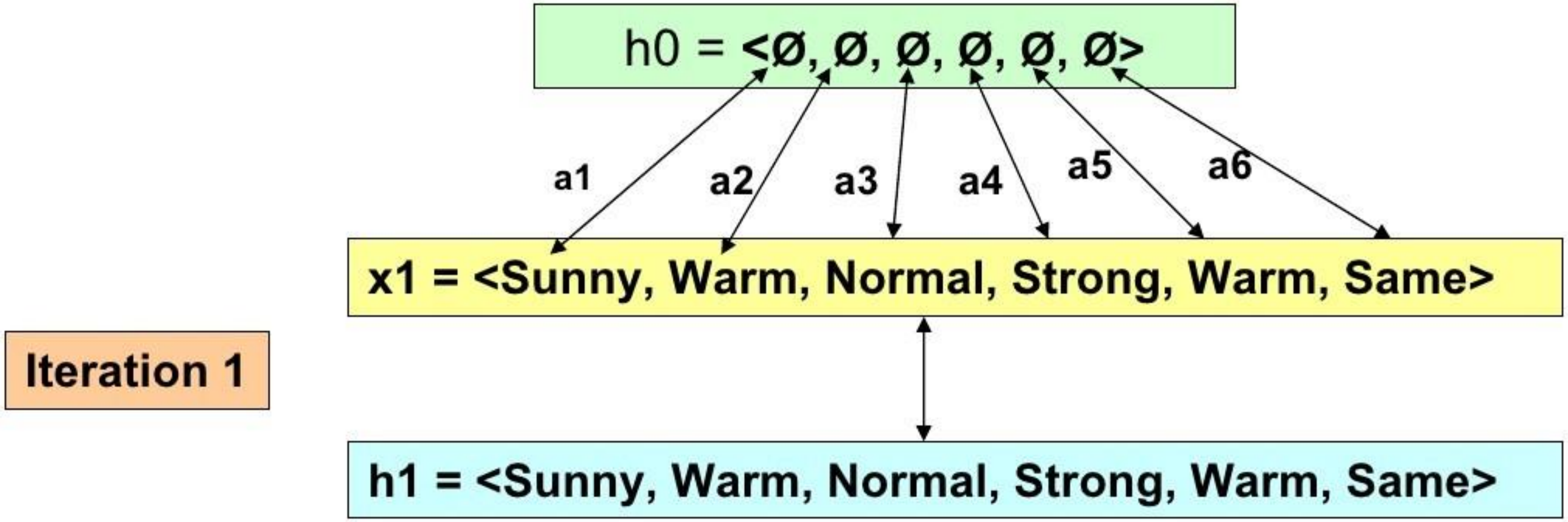
1. Initialize *h* to the most specific hypothesis in *H*

h0 = **<Ø, Ø, Ø, Ø, Ø, Ø>**

Step 2: FIND-S



```
2. For each positive training instance $x$
    • 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 satisfied by $x$
```



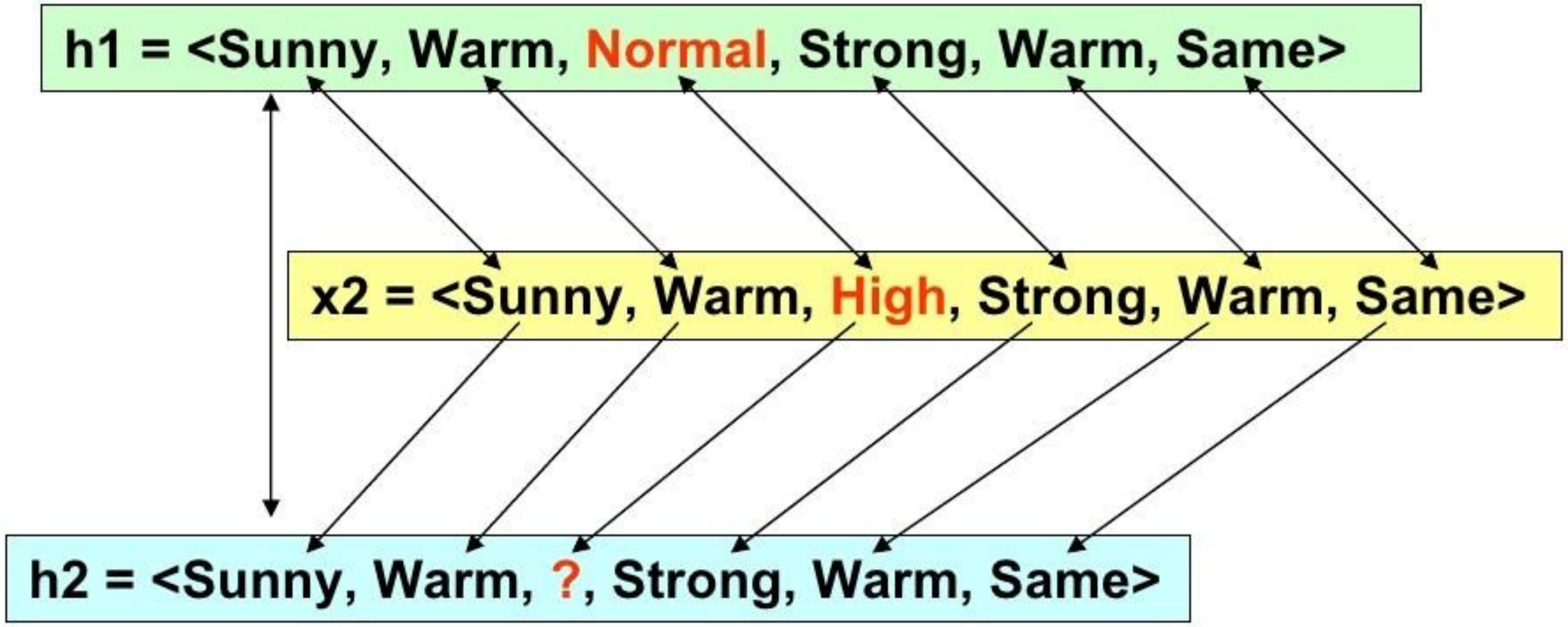
h0 = <Ø, Ø, Ø, Ø, Ø, Ø>

x1 = <Sunny, Warm, Normal, Strong, Warm, Same>

h1 = <Sunny, Warm, Normal, Strong, Warm, Same>

Iteration 1

Iteration 2



h1 = <Sunny, Warm, Normal, Strong, Warm, Same>

x2 = <Sunny, Warm, High, Strong, Warm, Same>

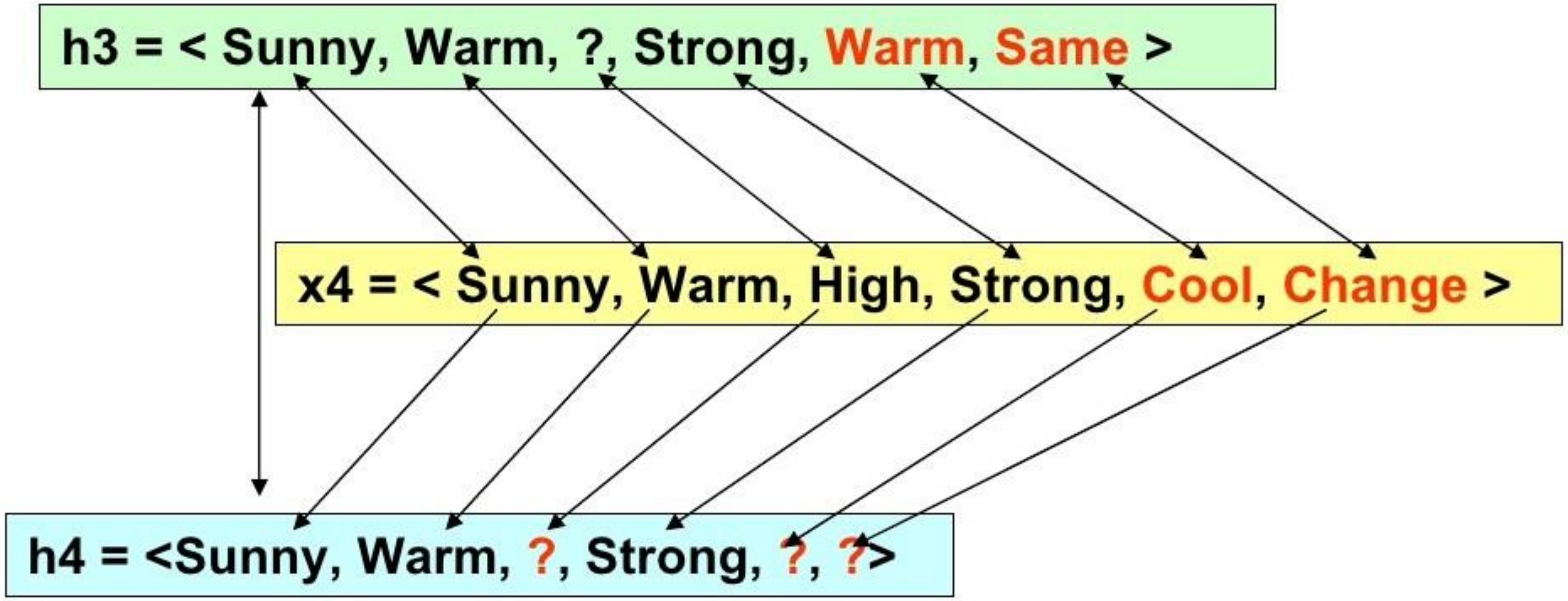
h2 = <Sunny, Warm, ?, Strong, Warm, Same>

Iteration 3

Ignore

h3 = <Sunny, Warm, ?, Strong, Warm, Same>

Iteration 4



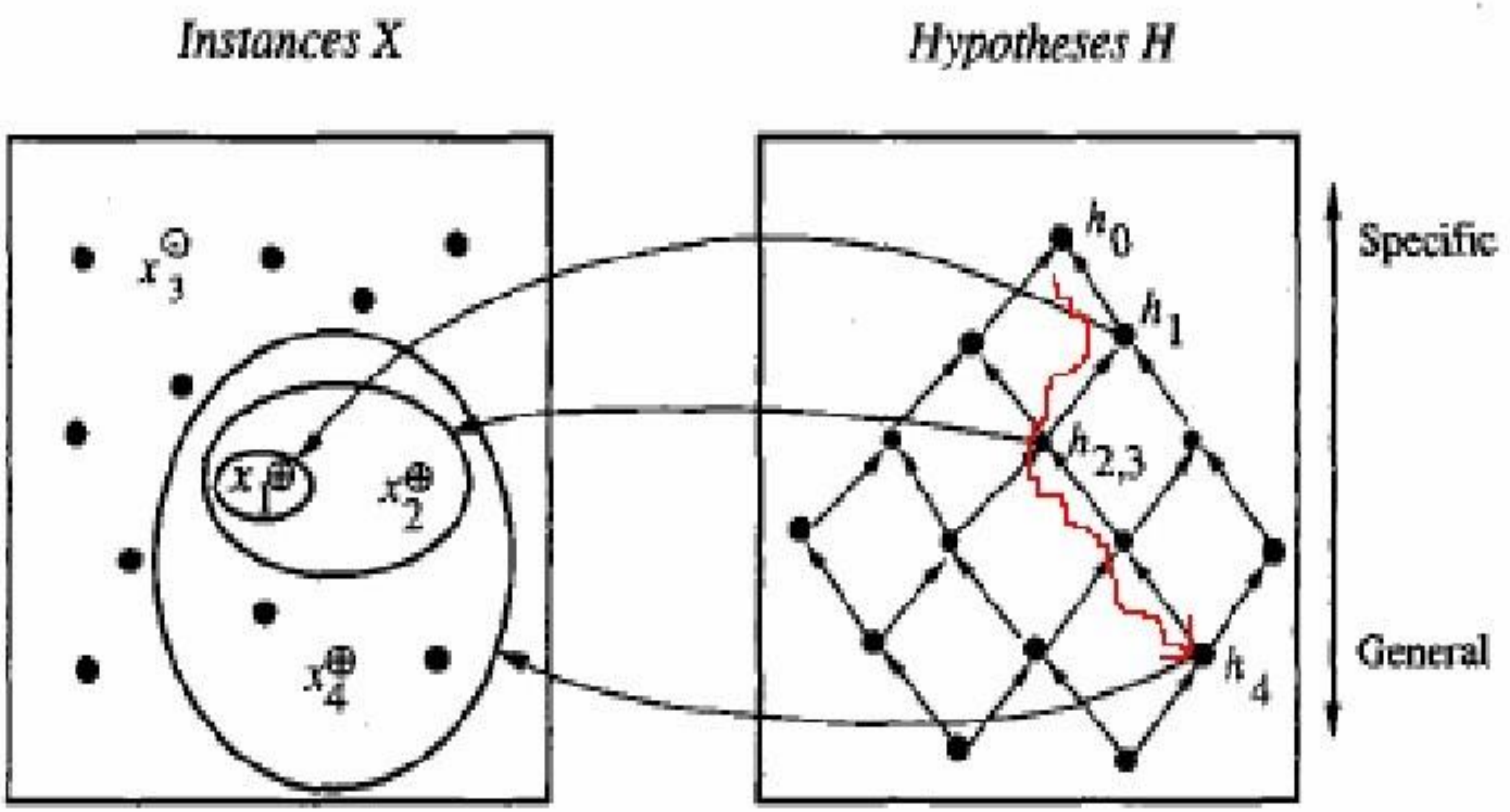
h3 = < Sunny, Warm, ?, Strong, Warm, Same >

x4 = < Sunny, Warm, High, Strong, Cool, Change >

Step 3

Output

h4 = <Sunny, Warm, ?, Strong, ?, ?>



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>, +

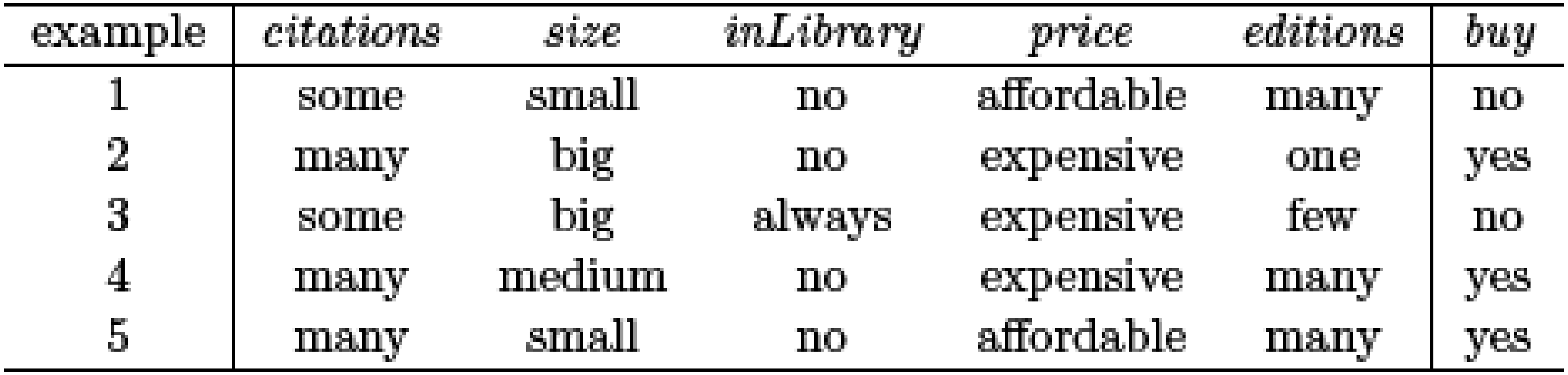
h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

h_1 = <Sunny Warm Normal Strong Warm Same>

h_2 = <Sunny Warm ? Strong Warm Same>

h_3 = <Sunny Warm ? Strong Warm Same>

h_4 = <Sunny Warm ? Strong ? ? >



example	*citations*	*size*	*inLibrary*	*price*	*editions*	*buy*
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

1. How many concepts are possible for this instance space?
2. How many hypotheses can be expressed by the hypothesis language presented in the lecture for this instance space?
3. Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.

Key Properties of FIND-S Algorithm

- Guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.

- Final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H, and provided the training examples are correct.

Unanswered Questions by FIND-S

- **Has the learner converged to the correct target concept?**

  Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the only hypothesis in ***H*** consistent with the data (i.e., the correct target concept

- **Why prefer the most specific hypothesis?**

  In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.

Unanswered Questions by FIND-S

- **Are the training examples consistent?**

In most practical learning problems there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples. We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.

Unanswered Questions by FIND-S

- **What if there are several maximally specific consistent hypotheses?**

In the hypothesis language H for the ***EnjoySport*** task, there is always a unique, most specific hypothesis consistent with any set of positive examples. However, for other hypothesis spaces (which will be discussed later) there can be several maximally specific hypotheses consistent with the data.

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)

Consistent

- An hypothesis h is consistent with a set of training examples D iff $h(x) = c(x)$ for each example in D

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

Version space

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

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

The *List-Then-Eliminate* algorithm

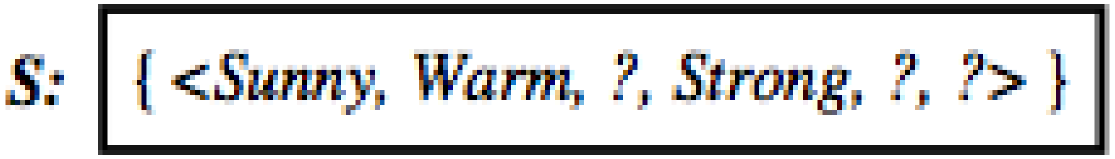
Version space as list of hypotheses

1. V$ersionSpace \leftarrow$ 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 $VersionSpace$

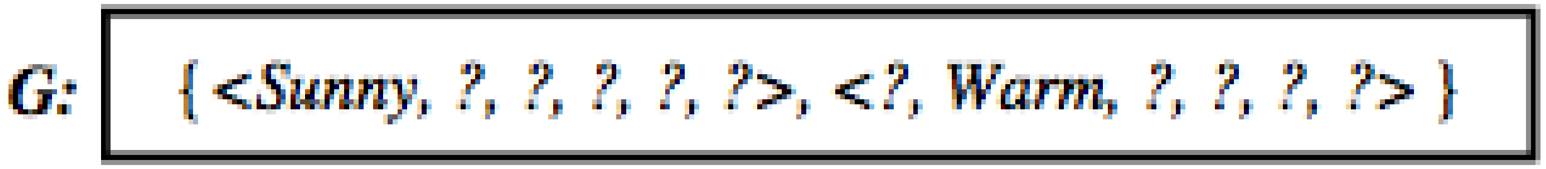
•Problems

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

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

  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 (h consistent with D) 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:

$\Leftarrow$ 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 $VS_{H,D}$

$\Rightarrow$ It can be proved by assuming a consistent h that does not satisfy the right-hand side and by showing that this would lead to a contradiction

Candidate elimination algorithm

$S \leftarrow$ minimally general hypotheses in H,

$G \leftarrow$ maximally general hypotheses in H

Initially any hypothesis is still possible

$$S_0 = \langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$$

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

Candidate elimination algorithm

For each training example d, do:

If d is *positive* example

Remove from G any hypothesis h inconsistent with d

For each hypothesis s in S not consistent with d:

- Remove s from S
- Add to S all minimal generalizations 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

If d is *negative* example

Remove from S any hypothesis h inconsistent with d

For each hypothesis g in G 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

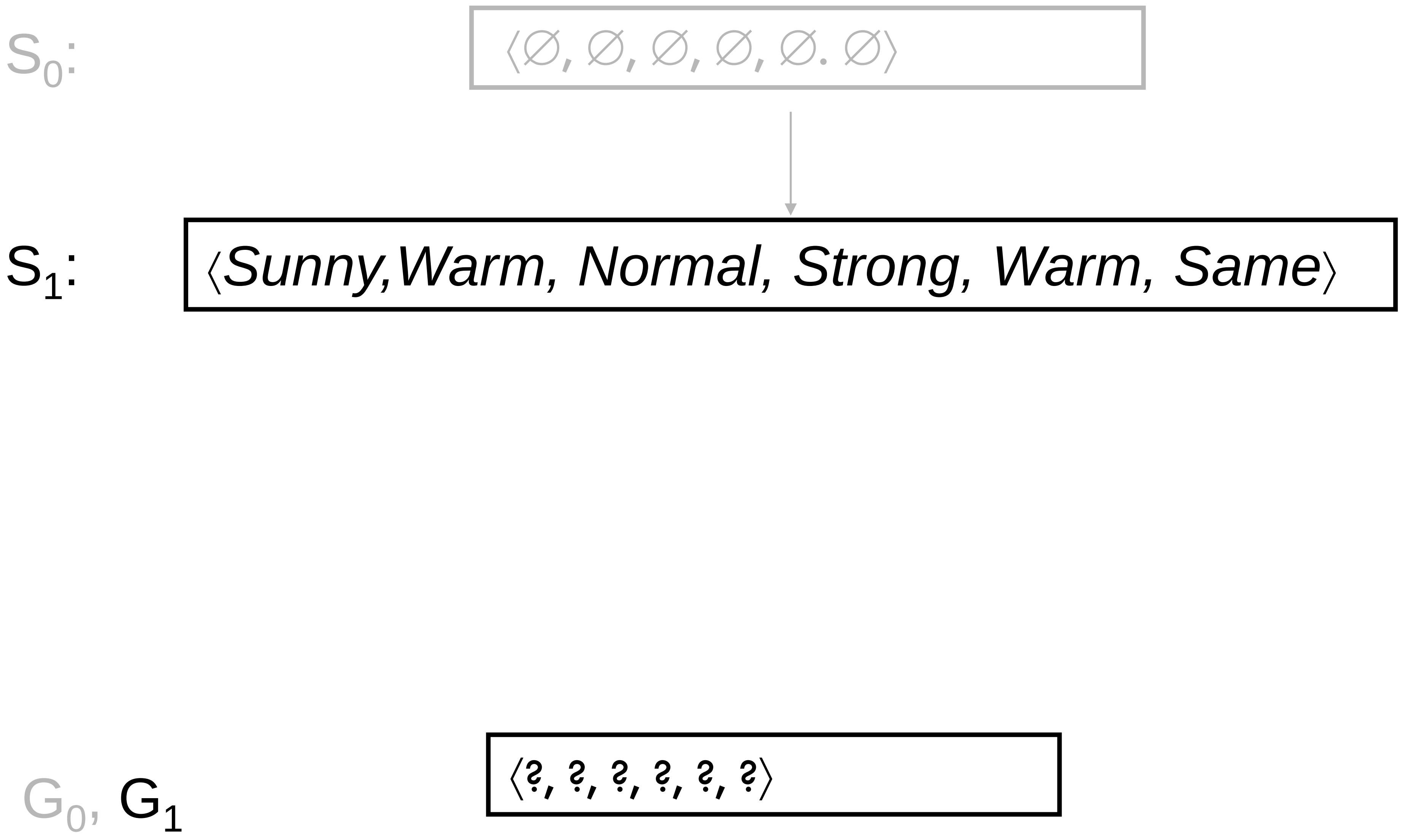
Example: initially

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:

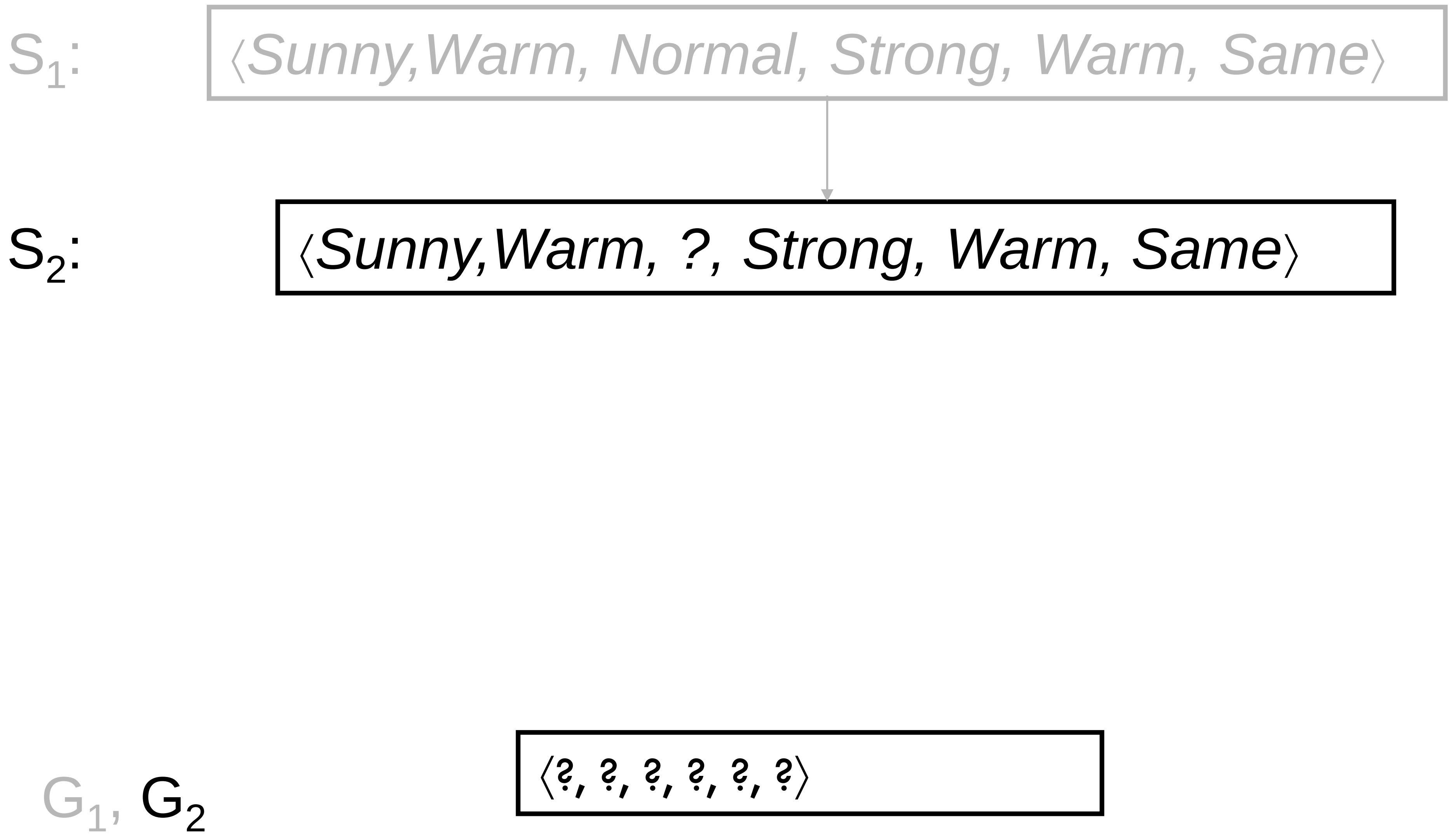
⟨*Sunny,Warm, Normal, Strong, Warm, Same*⟩

G_0, G_1

⟨?, ?, ?, ?, ?, ?⟩

Example:

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



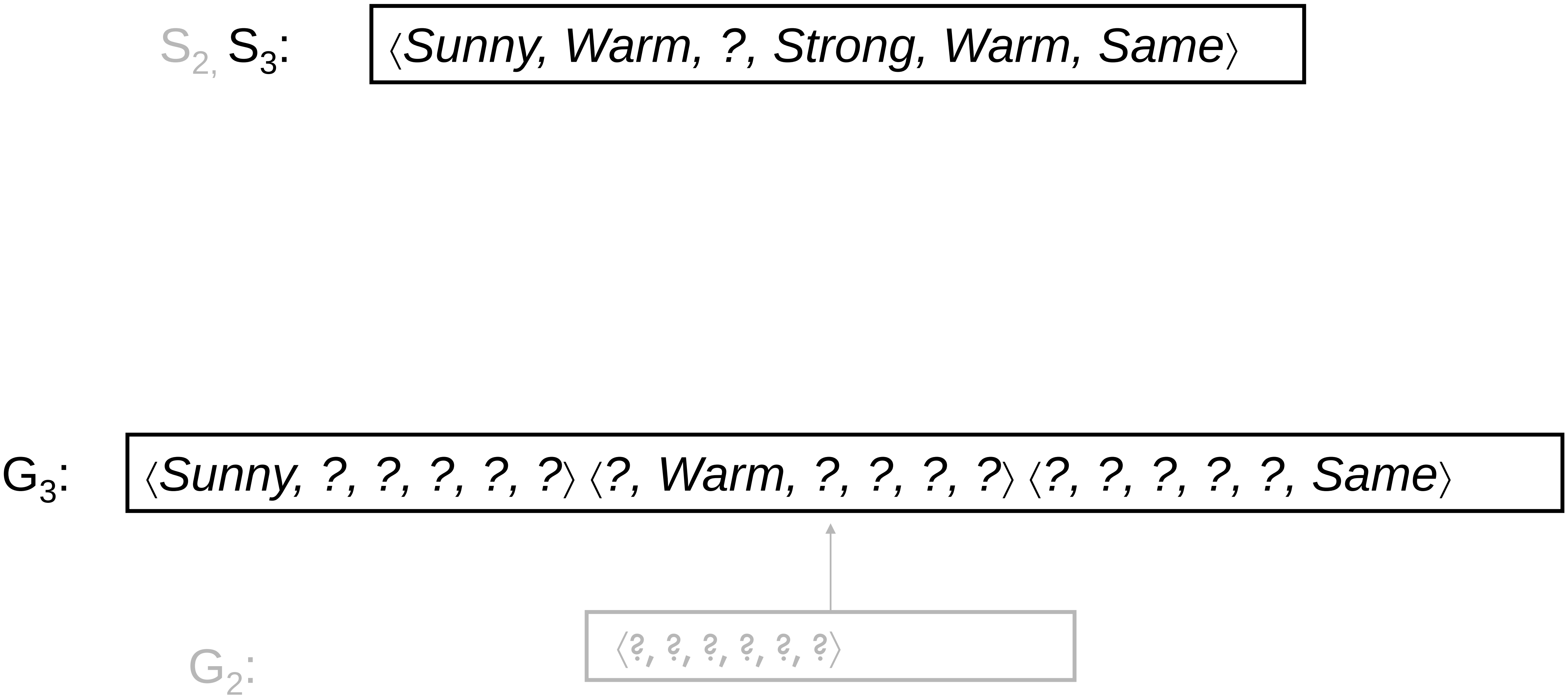
S$_1$: ⟨*Sunny,Warm, Normal, Strong, Warm, Same*⟩

S$_2$: ⟨*Sunny,Warm, ?, Strong, Warm, Same*⟩

G$_1$, G$_2$ ⟨?, ?, ?, ?, ?, ?⟩

Example:

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



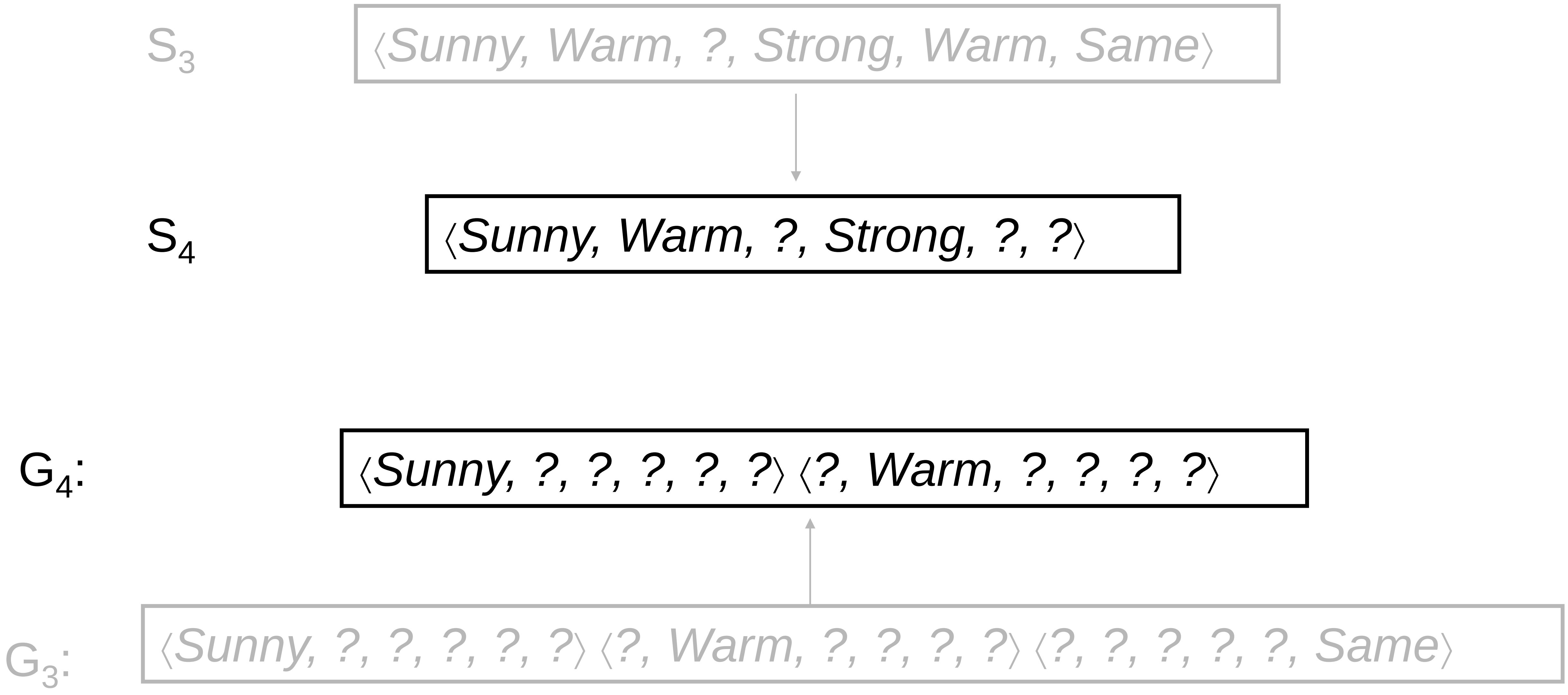
$S_{2,}$ S_3: ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

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

G_2: ⟨?, ?, ?, ?, ?, ?⟩

Example:

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



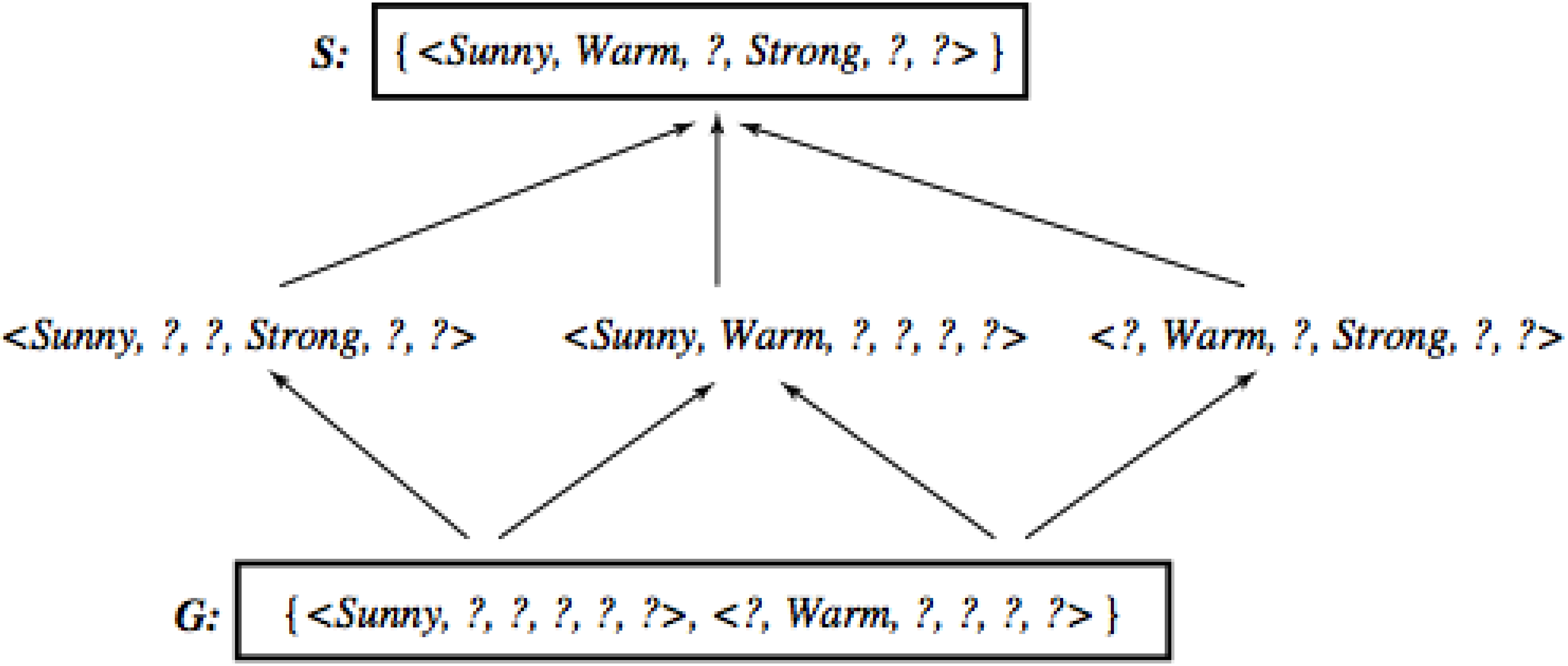
S_3 ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

S_4 ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

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

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

Learned Version Space



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

<*Sunny, ?, ?, Strong, ?, ?*>

<*Sunny, Warm, ?, ?, ?, ?*>

<*?, Warm, ?, Strong, ?, ?*>

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

Observations

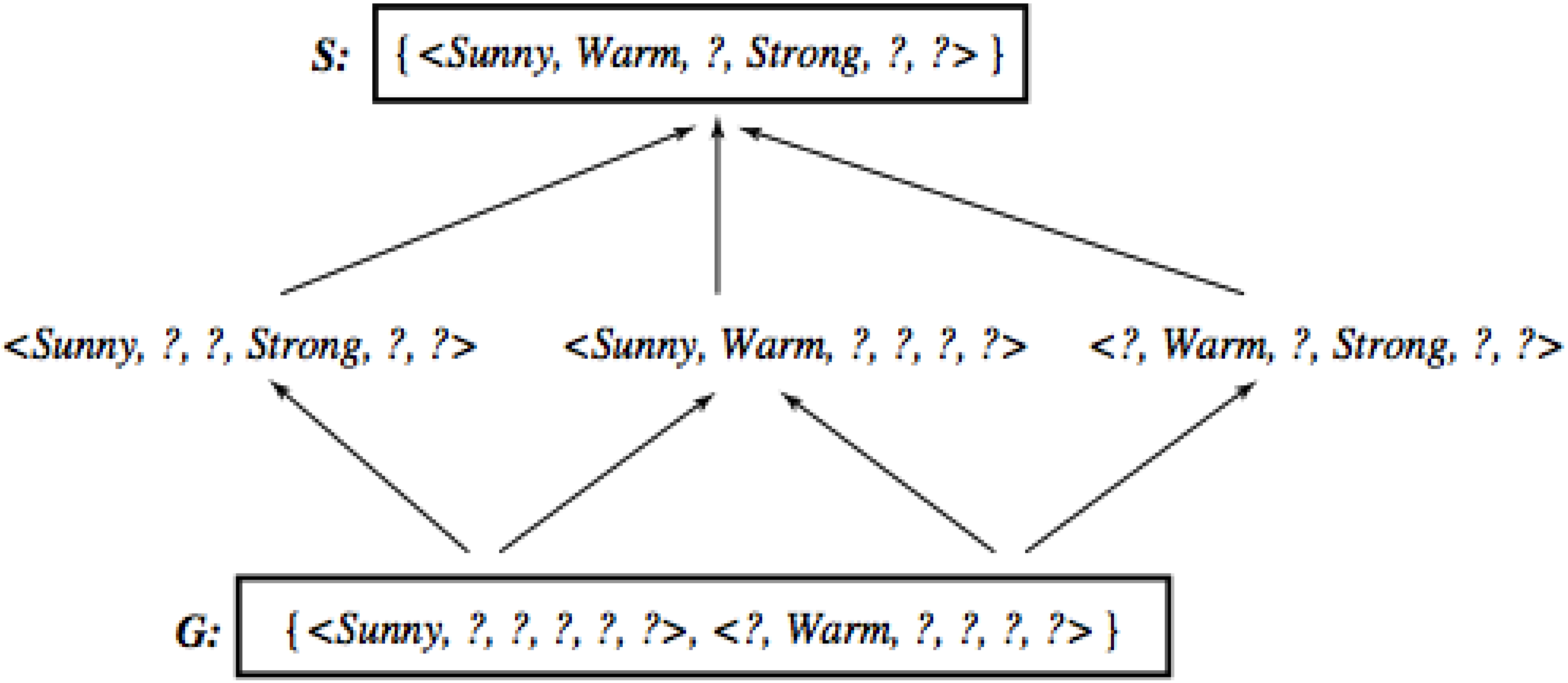
- The learned Version Space correctly describes the target concept, provided:
  1. There are no errors in the training examples
  2. There is some hypothesis that correctly describes the target concept
- If S and G converge to a single hypothesis the concept is exactly learned
- In case of errors in 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.
  - Ideally 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



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

<*Sunny, ?, ?, Strong, ?, ?*>

<*Sunny, Warm, ?, ?, ?, ?*>

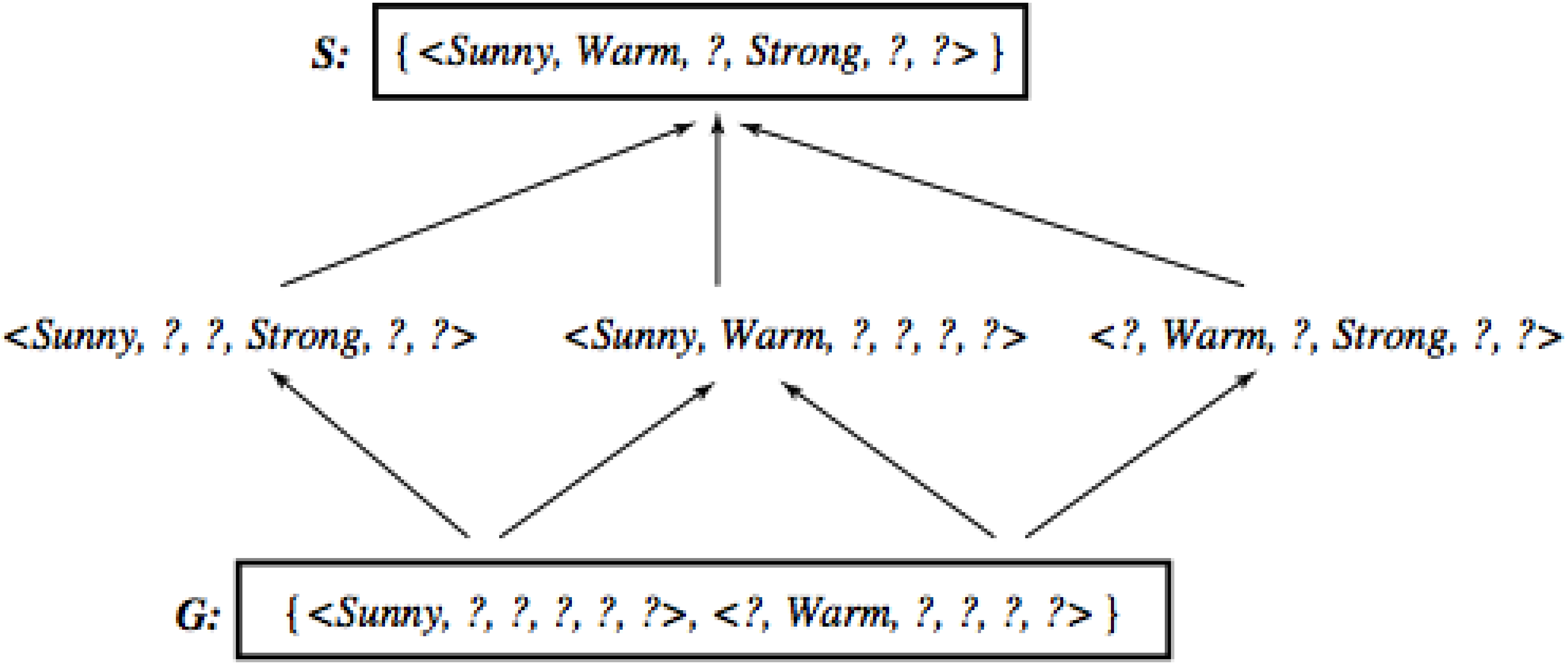
<*?, Warm, ?, Strong, ?, ?*>

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

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

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

Classifying new examples



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

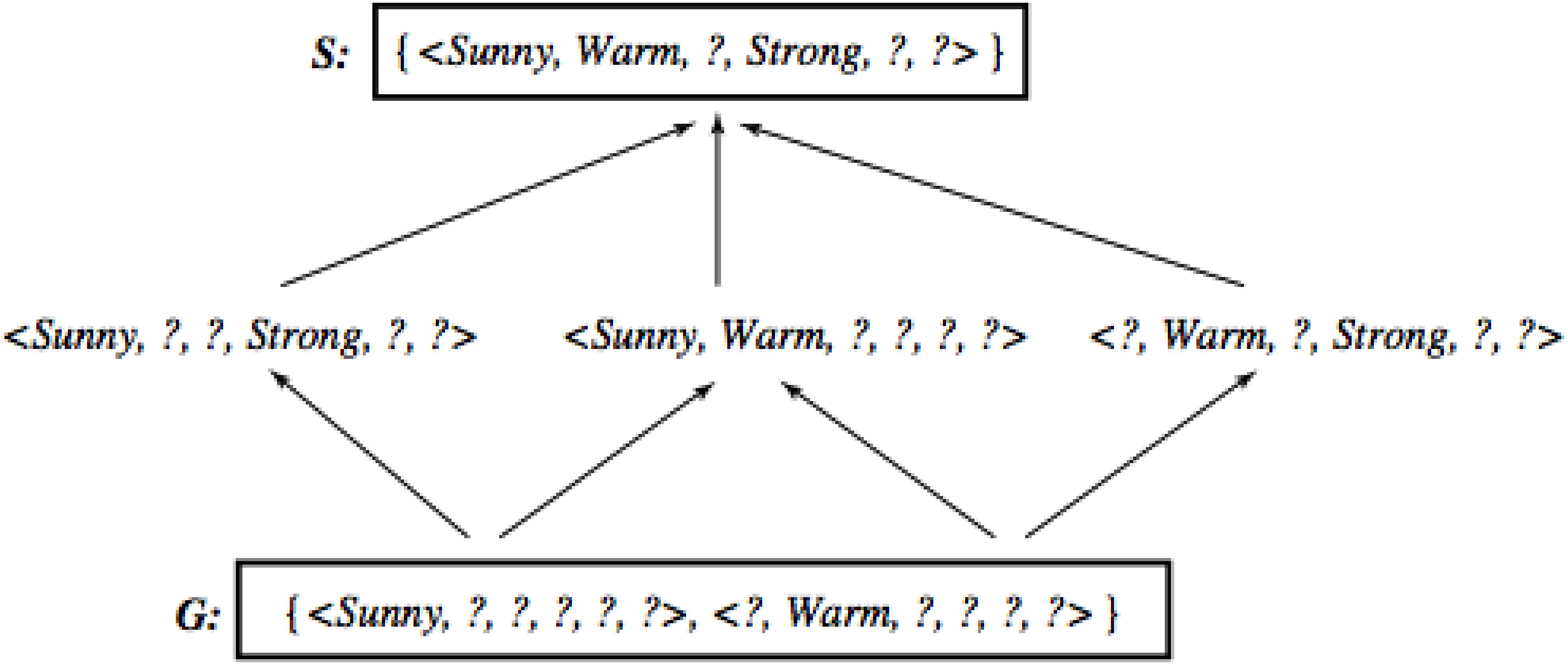
<Sunny, ?, ?, Strong, ?, ?>   <Sunny, Warm, ?, ?, ?, ?>   <?, Warm, ?, Strong, ?, ?>

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

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

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

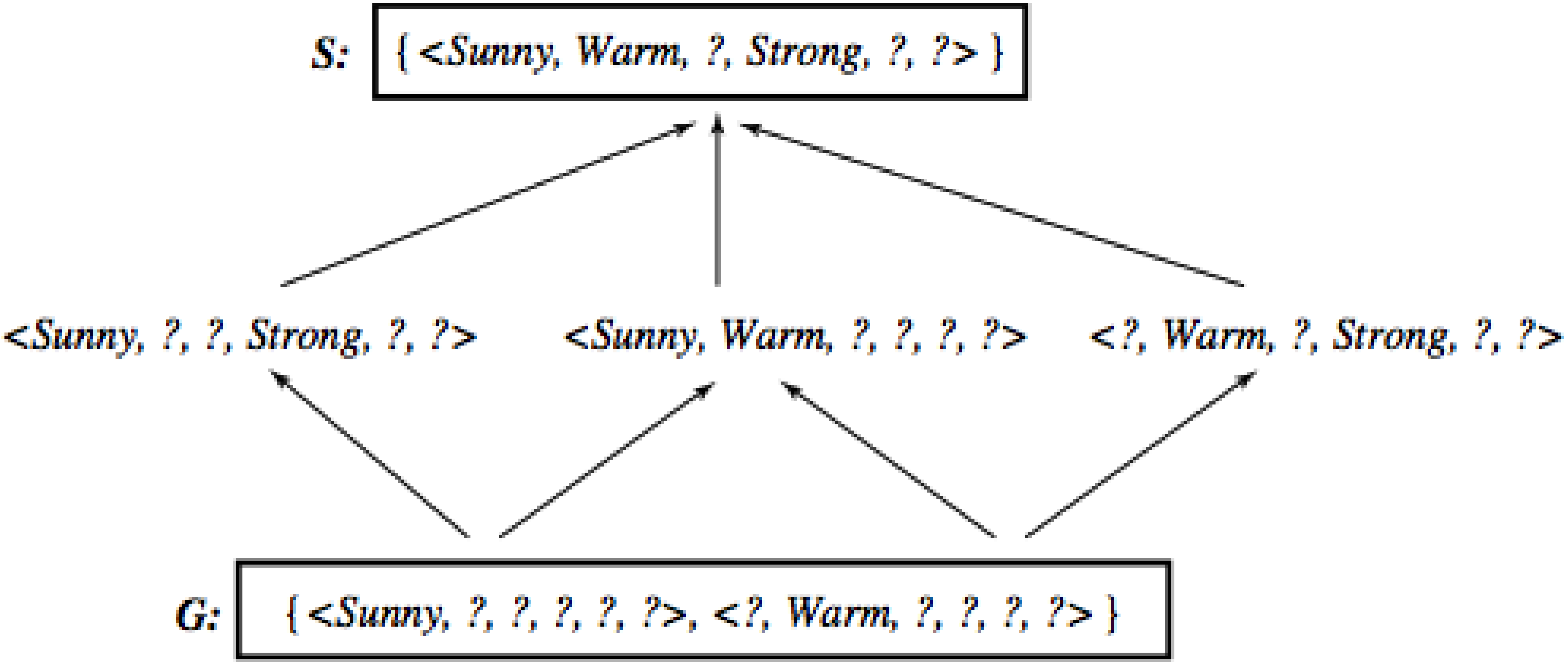
Classifying new examples



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

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

Classifying new examples



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

<*Sunny, ?, ?, Strong, ?, ?*>

<*Sunny, Warm, ?, ?, ?, ?*>

<*?, Warm, ?, Strong, ?, ?*>

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

⟨*Sunny, Cold, Normal, Strong, Warm, Same*⟩

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

  $\langle ?, \mathit{Warm}, \mathit{Normal}, \mathit{Strong}, \mathit{Cool}, \mathit{Change}\rangle$ is too general

- Target concept exists in a different space H', including disjunction and in particular the hypothesis

  $\mathit{Sky}=\mathit{Sunny}$ or $\mathit{Sky}=\mathit{Cloudy}$

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
  $\langle Sunny, ?, ?, ?, ?, ?\rangle$ V $<Cloudy, ?, ?, ?, ?, ?\rangle$
  Sunny(Sky) V Cloudy(Sky)
- 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!
- Take a majority vote?
  - 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

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: $D_c = \{\langle x, c(x)\rangle\}$
  - $L(\mathrm{x}_i, D_c)$ outcome of classification of x_i after learning
- Inductive inference ($\succ$):

  $$D_c \wedge \mathrm{x}_i \succ L(x_i, D_c)$$

- The *inductive bias* is defined as a minimal set of assumptions ***B***, such that ($\vdash$ for deduction)

  $$\forall\, (\mathrm{x}_i \in X)\ [\ (B \wedge D_c \wedge \mathrm{x}_i) \vdash L(x_i, D_c)\ ]$$

Inductive bias of Candidate-Elimination

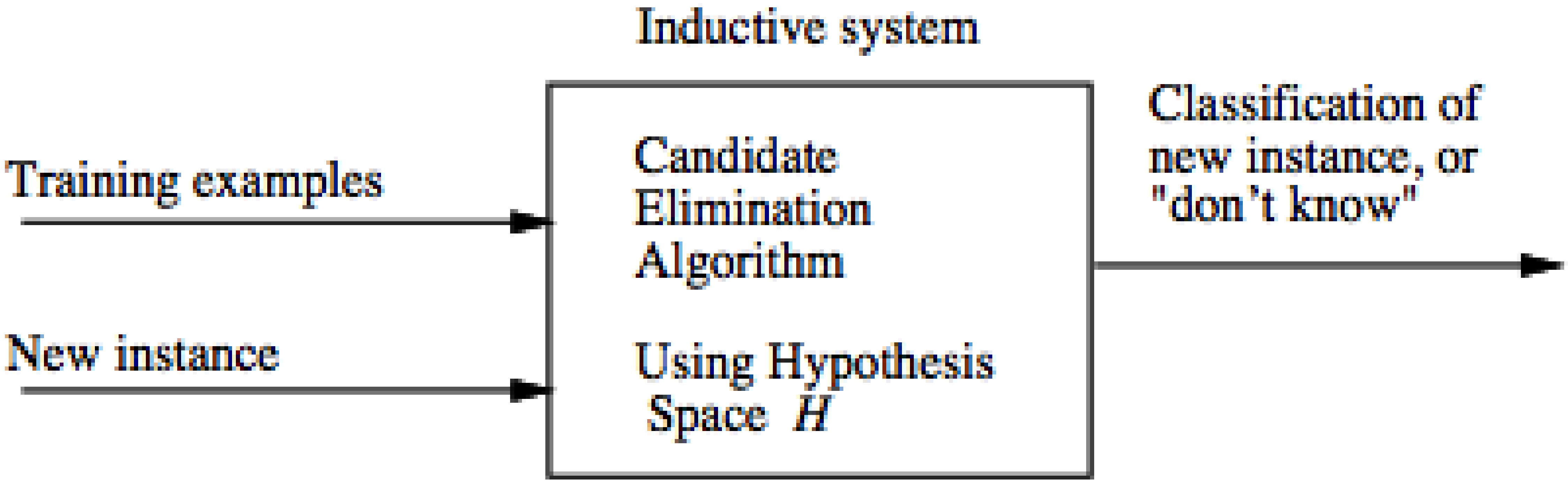
- Assume *L* is defined as follows:
  - compute $VS_{\mathrm{H,D}}$
  - classify new instance by complete agreement of all the hypotheses in $VS_{\mathrm{H,D}}$
- Then the inductive bias of Candidate-Elimination is simply

  $B \equiv (c \in \mathrm{H})$

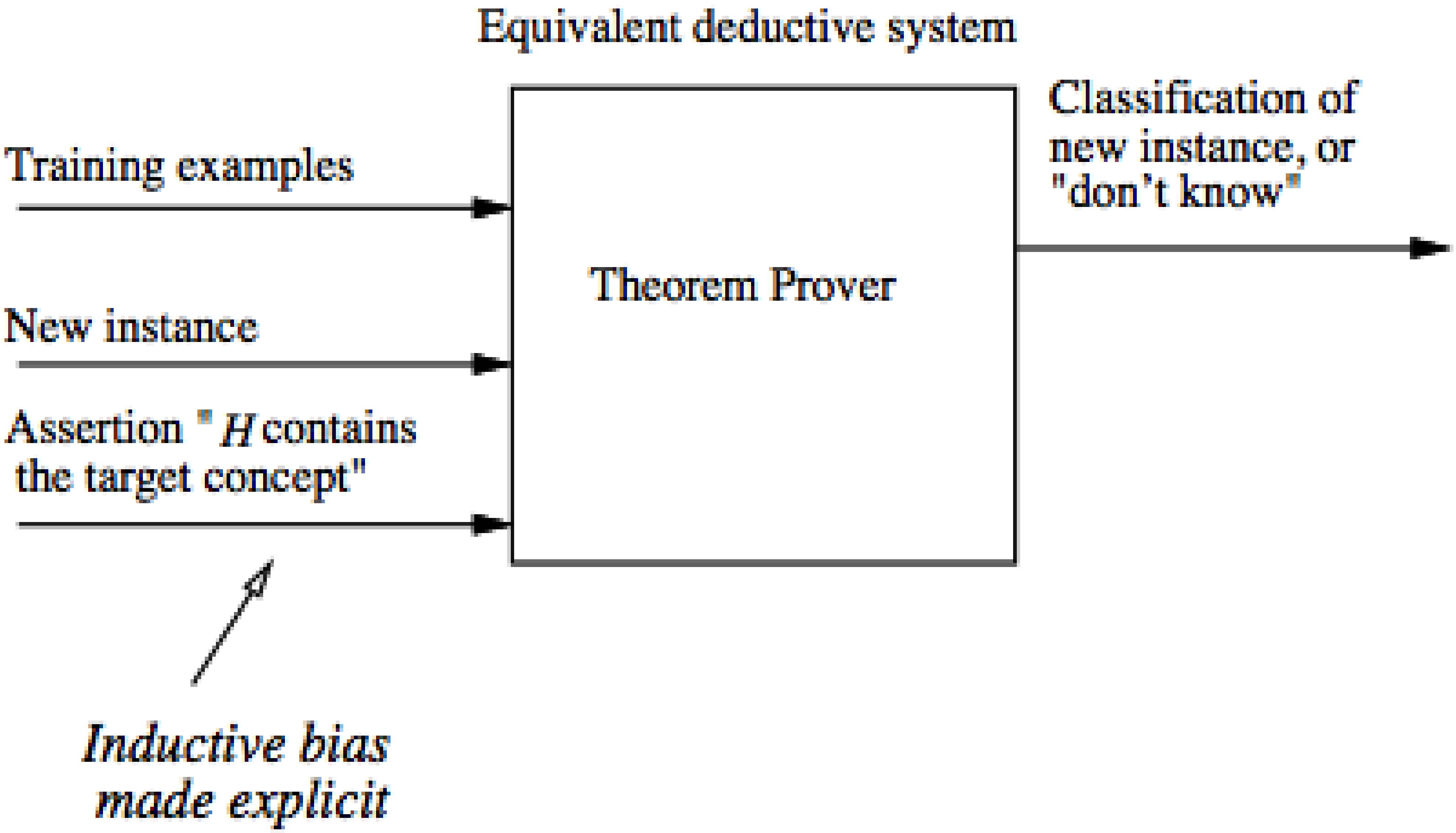
- In fact by assuming $c \in H$:

1. $c \in VS_{\mathrm{H,D}}$, in fact $VS_{\mathrm{H,D}}$ 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


Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space H
Classification of new instance, or "don't know"

Equivalent deductive system


Equivalent deductive system
Training examples
New instance
Assertion " H contains the target concept"
Theorem Prover
Classification of new instance, or "don't know"
Inductive bias made explicit

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” (stronger bias)

  – The stronger the bias, greater the ability to generalize and classify new instances (greater inductive leaps).