CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

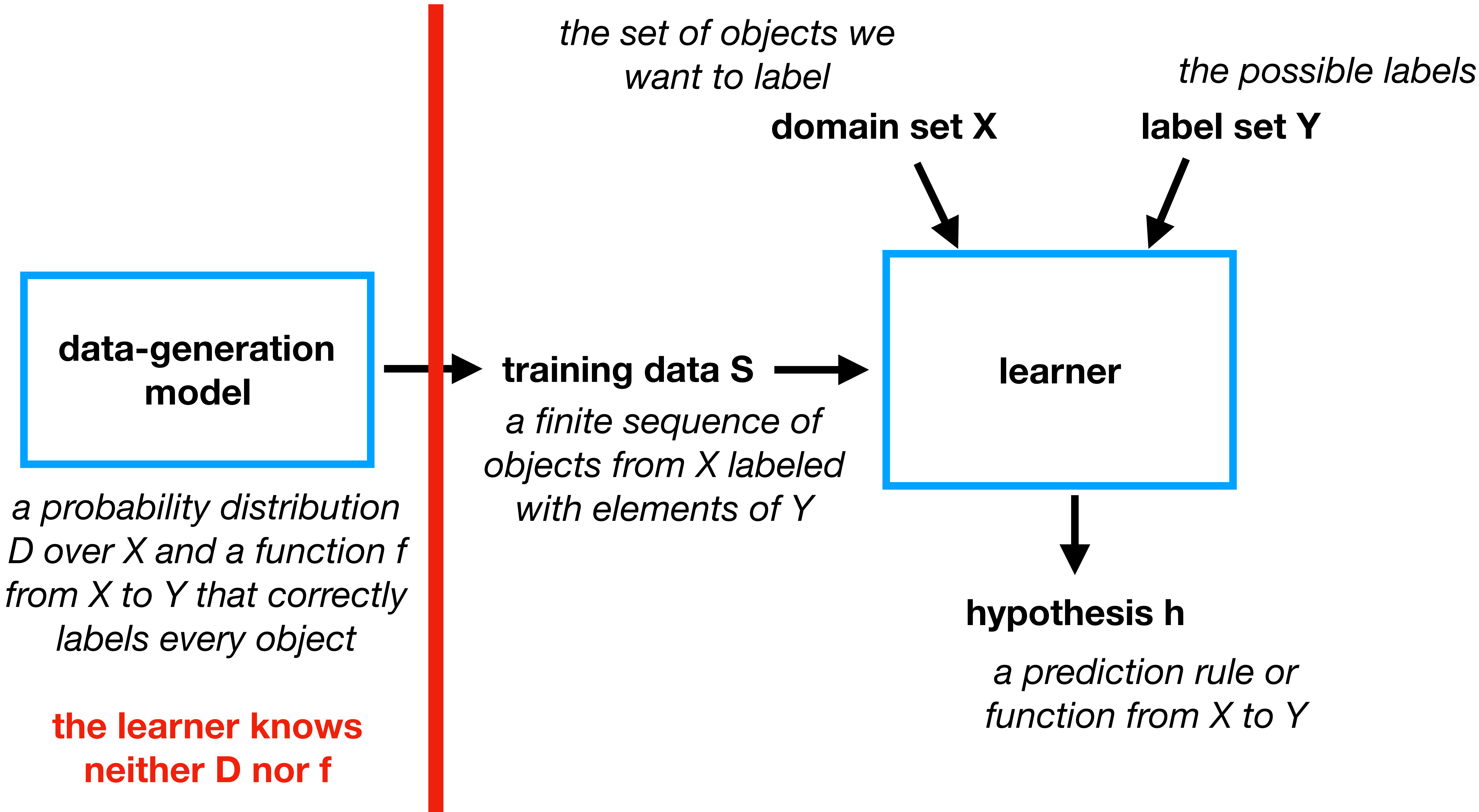
CMT311 Principles of Machine Learning

Concept Learning, ERM & PAC

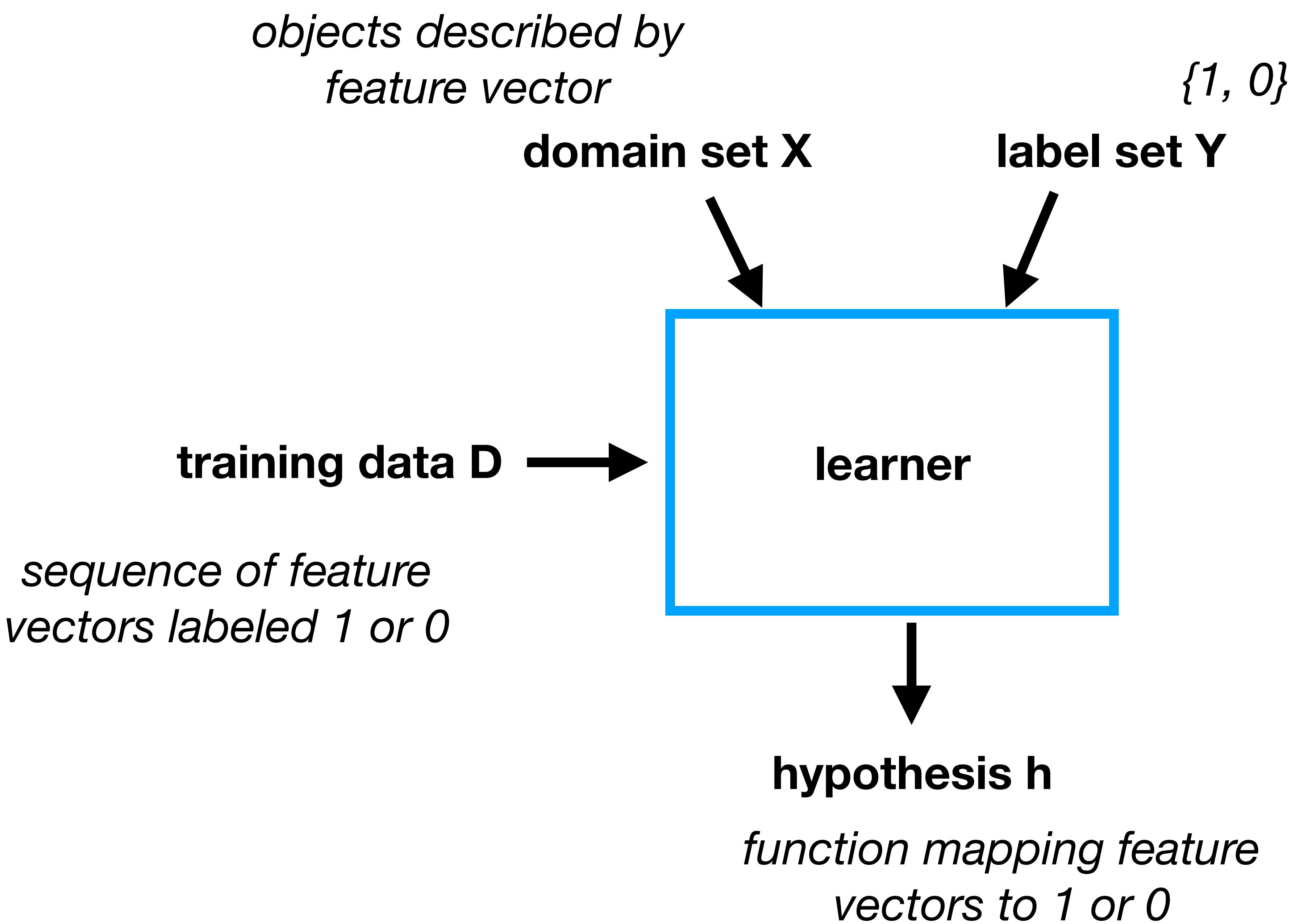
Angelika Kimmig
KimmigA@cardiff.ac.uk

11.10.2019

The Statistical Learning Framework



Boolean Concept Learning



Example

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy
Warm / Cold
Normal / High
Strong / Weak
Warm / Cool
Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day

domain set X

label set Y

training data D

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

learner

hypothesis h
conjunction of attribute constraints using ?, -, specific values

Example

points on a grid, described by coordinates (x,y) with x∈ [0,10], y∈[0,10]

domain set X

labels 1 / 0

label set Y

training data D

(2,4) 1
(7,4) 1
(5,1) 0
(5,3) 1
(2,6) 0
(6,5) 1

learner

hypothesis h

rectangle (a≤x≤b ∧ c≤y≤d) with a,b,c,d integers in [0,10]

More-general-than

- Let h_j and h_k be two Boolean-valued functions defined over X.
- Then h_j is **more general than or equal to** h_k, $h_j \geq_g h_k$, if and only if $\forall x \in X : h_k(x) = 1 \rightarrow h_j(x) = 1$
- h_j is **strictly more general than** h_k, $h_j >_g h_k$, if and only if $h_j \geq_g h_k$ and $h_k \ngeq_g h_j$
- h_j is **more specific than** h_k if and only if h_k is more general than h_j
- note: these notions are **independent** of the target concept

General-to-specific ordering

$$h_j \geq_g h_k \text{ if and only if } \forall x \in X : h_k(x) = 1 \rightarrow h_j(x) = 1$$

< Sky, AirTemp, Humidity, Wind, Water, Forecast >

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

h_1	h_2
<?,Cold,?,?,?,?>	<?,Cold,High,?,?,?>
<?,Cold,?,Strong,Cool,?>	<?,?,?,?,?,?>
<?,Cold,?,Strong,Cool,?>	<?,Cold,High,?,?,?>
<?,Cold,?,?,?,?>	<-,-,-,-,-,->
<-,-,-,-,-,->	<?,Cold,High,-,?,?>
<Sunny,Cold,High,Weak,Warm,Same>	<Sunny,Cold,High,Weak,Cool,Same>

General-to-specific ordering

$$h_j \geq_g h_k \text{ if and only if } \forall x \in X : h_k(x) = 1 \rightarrow h_j(x) = 1$$

rectangle (a≤x≤b ∧ c≤y≤d)
with a,b,c,d integers in [0,10]

h_1	h_2
(0≤x≤10 ∧ 0≤y≤10)	(0≤x≤10 ∧ 1≤y≤5)
(0≤x≤10 ∧ 1≤y≤5)	(0≤x≤9 ∧ 1≤y≤5)
(10≤x≤10 ∧ 1≤y≤1)	(0≤x≤10 ∧ 1≤y≤5)
(0≤x≤10 ∧ 1≤y≤5)	(10≤x≤0 ∧ 1≤y≤5)
(10≤x≤0 ∧ 1≤y≤5)	(3≤x≤1 ∧ 10≤y≤5)
(2≤x≤4 ∧ 3≤y≤7)	(1≤x≤4 ∧ 3≤y≤8)

A basic learner: FIND-S

- set *h* to the most specific hypothesis in *H*
- for each positive *x* in *D*
    - for each constraint *a* in *h*
        - if *x* does not satisfy *a* then replace *a* in *h* by the next more general constraint *a'* that is satisfied by *x*
- return *h*

Example

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy

Warm / Cold

Normal / High

Strong / Weak

Warm / Cool

Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day

domain set X

label set Y

training data D

learner

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

hypothesis h

conjunction of attribute constraints using ?, -, specific values

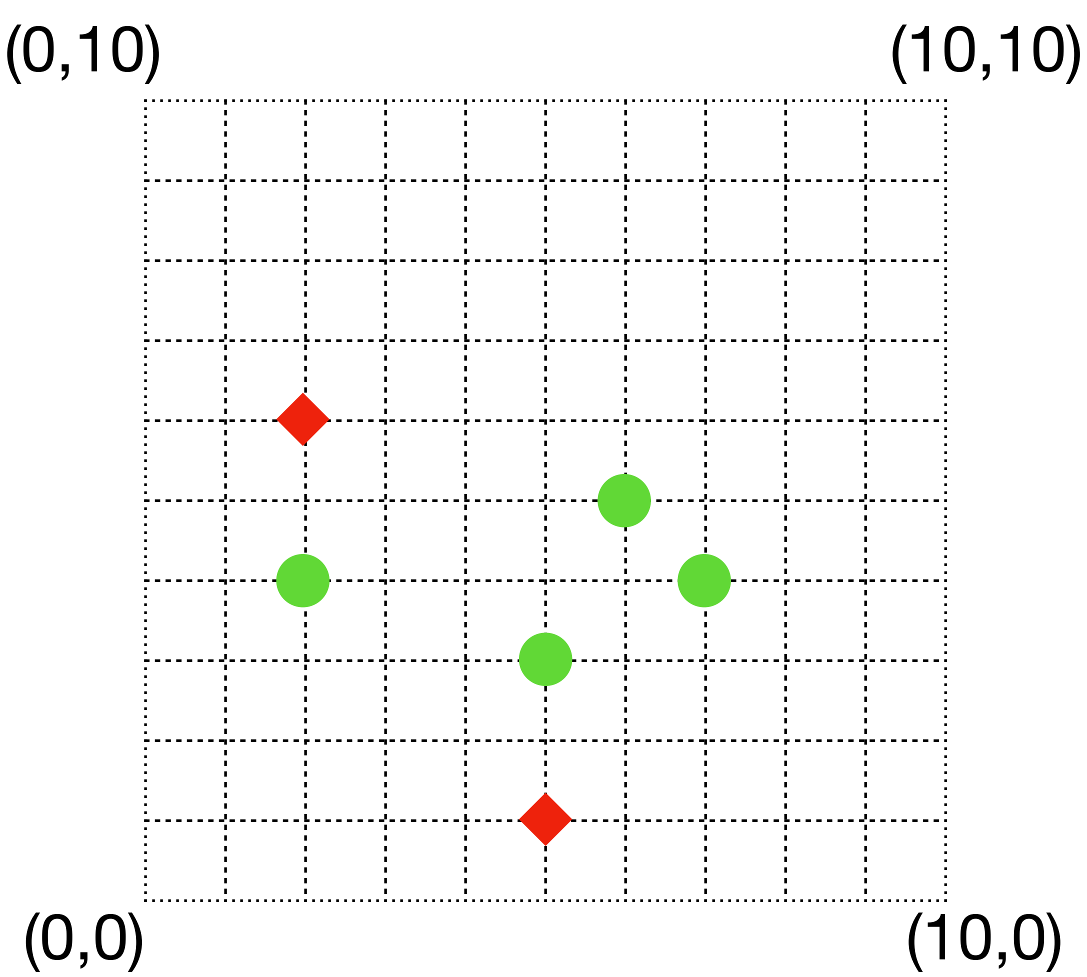
training example	current hypothesis h
-	<-,-,-,-,-,->
<Sunny, Warm, Normal, Strong, Warm, Same> 1	<Sunny, Warm, Normal, Strong, Warm, Same>
<Sunny, Warm, High, Strong, Warm, Same> 1	<Sunny, Warm, ?, Strong, Warm, Same>
<Rainy, Cold, High, Strong, Warm, Change> 0	<Sunny, Warm, ?, Strong, Warm, Same>
<Sunny, Warm, High, Strong, Cool, Change> 1	<Sunny, Warm, ?, Strong, ?, ?>

hypothesis returned by FIND-S

Exercise

- Consider again the space of rectangles (a≤x≤b ∧ c≤y≤d) on the [0,10]x[0,10] grid.

- Trace the FIND-S algorithm for the following sequence of examples:
  (2,4) 1
  (7,4) 1
  (5,1) 0
  (5,3) 1
  (2,6) 0
  (6,5) 1



FIND-S: Discussion

- the hypothesis returned by FIND-S is
  - the most specific one in H that correctly labels all positive training examples
  - correctly labels all negative training examples, provided that the correct target concept is in H and the training data is correct
- open questions:
  - has the learner converged to the correct answer?
  - why prefer the most specific h?
  - what if the training data is not labeled correctly?
  - what if there are several maximally specific hypotheses for the training data?

Using version spaces

- A hypothesis h is **consistent** with training data D if and only if for all examples (x,y) in D, h(x)=y

- Goal: a learner that finds all hypotheses in H that are consistent with D, using the “more general than” order

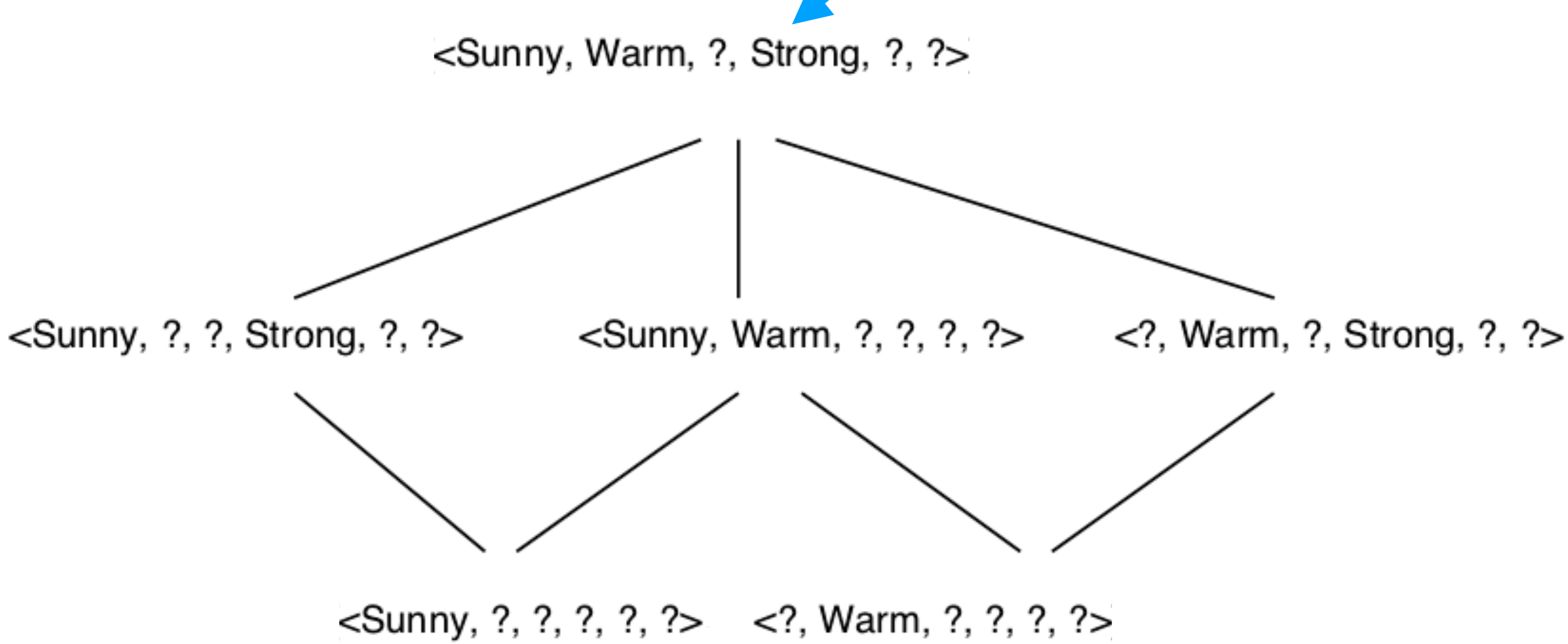
- The **version space** $VS_{H,D}$ with respect to hypothesis space H and training data D is the set of all hypotheses in H consistent with D

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

Example

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

the hypothesis returned by FIND-S on this data



[Figure: Mitchell]

another learner: LIST-THEN-ELIMINATE

- VS = list of all hypotheses in H
- for each example (x,y) in D
    - remove from VS all h with h(x)≠y
- return VS

Version space boundaries

- The **general boundary G** with respect to hypothesis space H and training data D is the set of maximally general members of H consistent with D.

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

- The **specific boundary S** with respect to hypothesis space H and training data D is the set of minimally general members of H consistent with D.

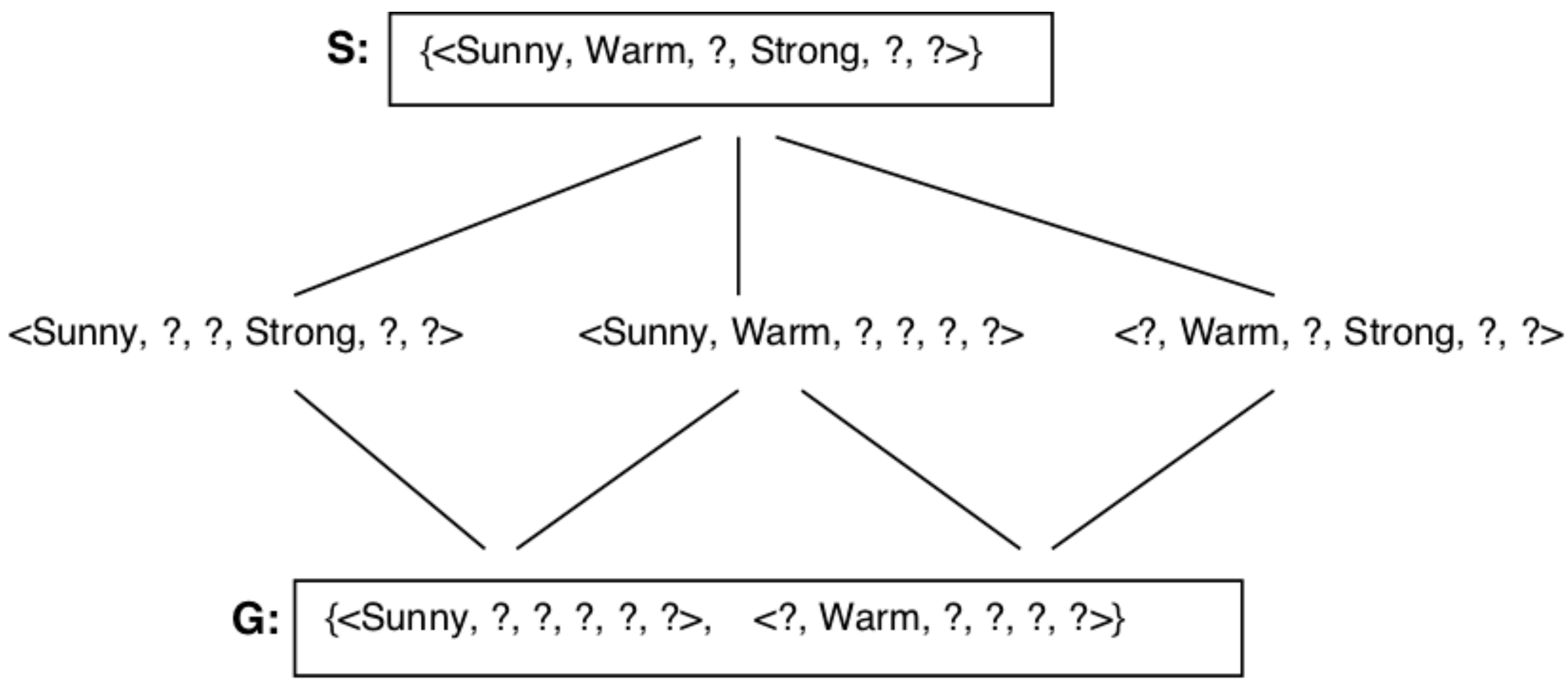
$$S \equiv \{s \in H \mid consistent(s, D) \wedge \neg \exists s' \in H : s >_g s' \wedge consistent(s', D)\}$$

- Every member of the version space lies between G and S:

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

Example

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1



[Figure: Mitchell]

CANDIDATE-ELIMINATION

- G = set of maximally general hypotheses in H
- S = set of maximally specific hypotheses in H
- for each training example d
    - if d is positive
        - remove from G any h inconsistent with d
        - for each s in S that is not consistent with d
            - remove s from S
            - add to S all minimal generalisations h of s such that h is consistent with d and some member of G is more general than h
            - remove from S any h that is more general than some h’ in S
    - if d is negative
        - remove from S any h inconsistent with d
        - for each g in G that is not consistent with d
            - remove g from G
            - add to G all minimal specialisations h of g such that h is consistent with d and some member of S is more specific than h
            - remove from G any h that is less general than some h’ in G

Example

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy

Warm / Cold

Normal / High

Strong / Weak

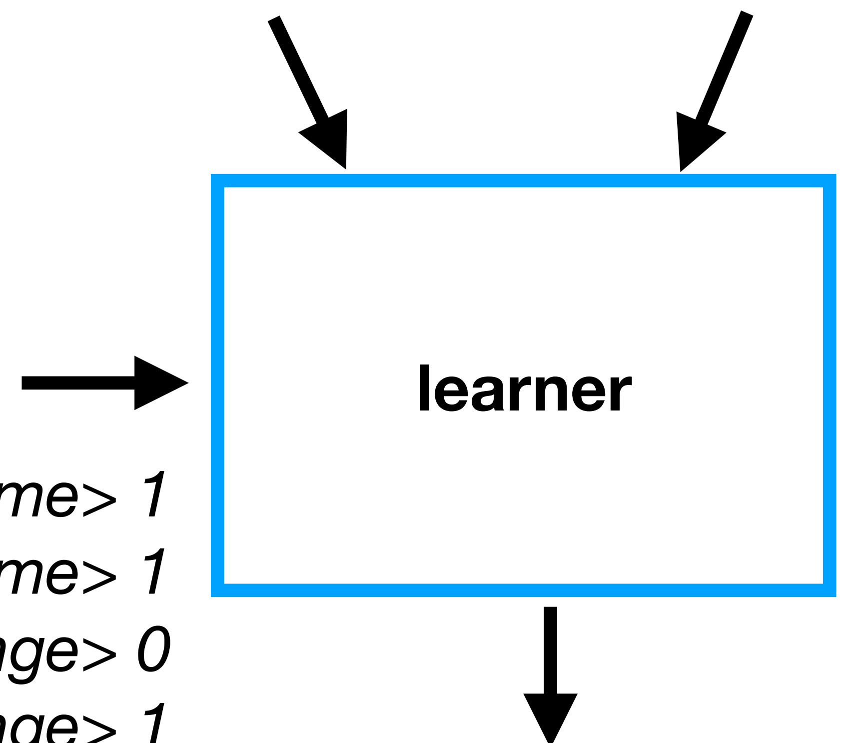
Warm / Cool

Same / Change

domain set X

label set Y

training data D



<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

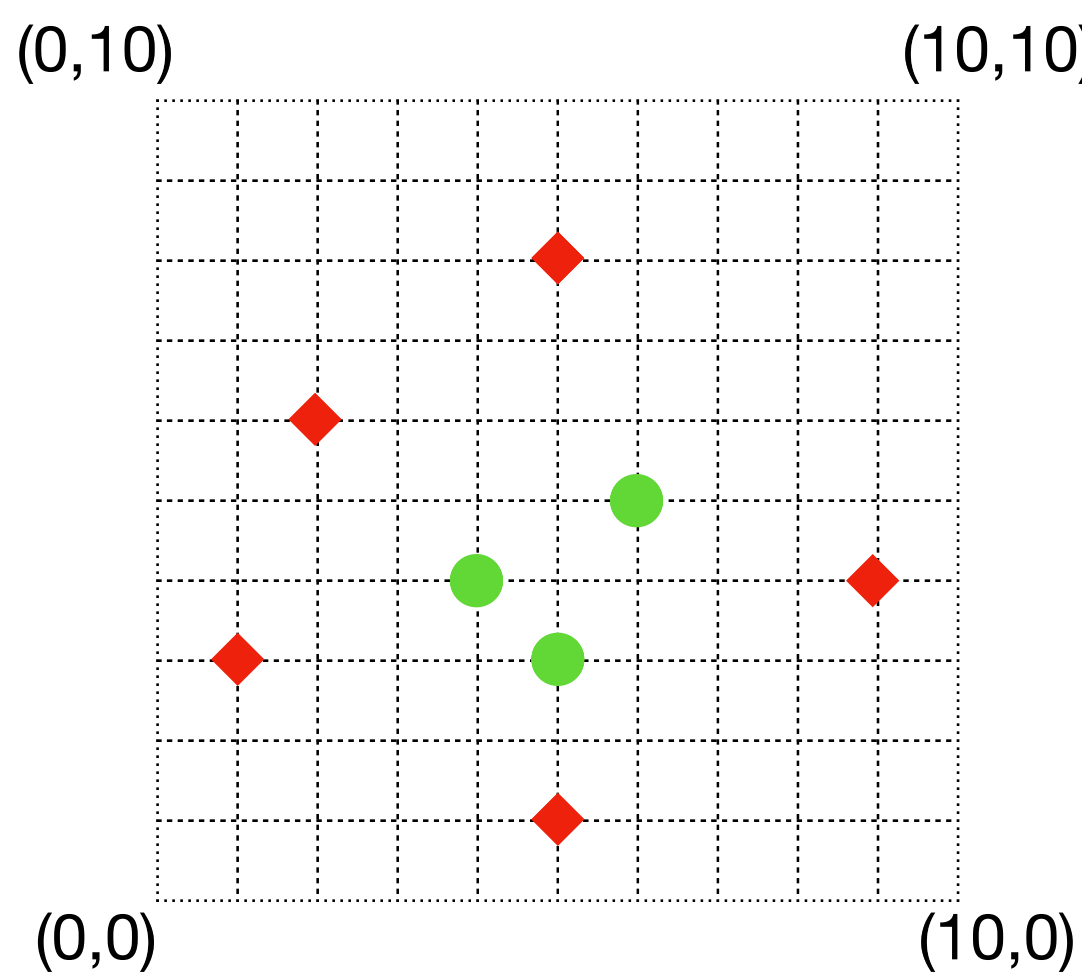
hypothesis h

conjunction of attribute constraints using ?, -, specific values

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

Exercise

- Consider again the space of rectangles ($a \leq x \leq b \wedge c \leq y \leq d$) on the $[0,10]x[0,10]$ grid, and the positive and negative training examples in the figure.
- What are the G and S boundaries of the version space? Write them down and draw them on the grid.
- Imagine the learner can ask the teacher to label a specific point as next training example. Suggest a point that would guarantee to shrink the version space independently of its label, and one that wouldn't.
- What is the smallest number of examples for which CANDIDATE-ELIMINATION can precisely learn any specific rectangle, say, ($2 \leq x \leq 8 \wedge 3 \leq y \leq 5$)?

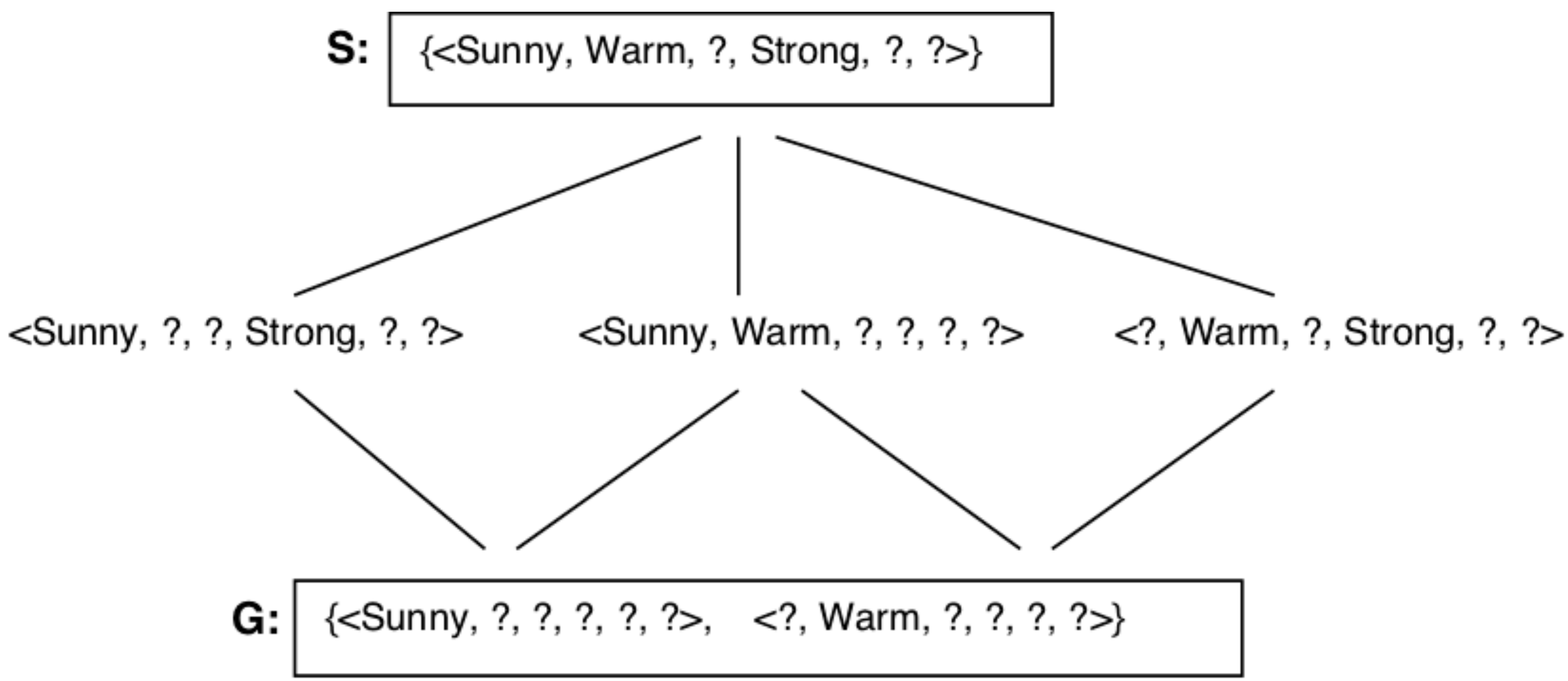


Discussion

- The version space learned by CANDIDATE-ELIMINATION converges towards the hypothesis correctly describing the target concept, provided that
  - there is such a hypothesis in H, and
  - the training data is labeled correctly
- The size of the version space tells us how close we are
- What if we don’t have enough data to converge?
- What if there is no correct h in H?

Using version spaces as classifiers

<Sunny, Warm, Normal, Strong, Cool, Change>
<Rainy, Cold, Normal, Light, Warm, Same>
<Sunny, Warm, Normal, Light, Warm, Same>
<Sunny, Cold, Normal, Strong, Warm, Same>



[Figure: Mitchell]

No correct h in H

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy

Warm / Cold

Normal / High

Strong / Weak

Warm / Cool

Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day

domain set X

label set Y

training data D

learner

<Sunny, Warm, Normal, Strong, Cool, Change> 1
<Cloudy, Warm, Normal, Strong, Cool, Change> 1
<Rainy, Warm, Normal, Strong, Cool, Change> 0

hypothesis h

conjunction of attribute constraints

No correct h in H

- Problem: there are many more Boolean functions over X than hypotheses in H, so the assumption that there is a good h in H is too strong
- What about including all these functions in H?
- Syntactically, this is easy: just allow any disjunctions, conjunctions and negations of our earlier hypotheses, e.g., <Sunny,?,?,?,?,?> v <Cloudy,?,?,?,?,?>

but…

- CANDIDATE-ELIMINATION now boils down to **memorisation**:
    - S = disjunction of all positive training examples
    - G = negated disjunction of all negative training examples
- only **converges** after **seeing all** instances
- every **unseen** instance is classified **positive by half** of the version space and **negative by the other half**

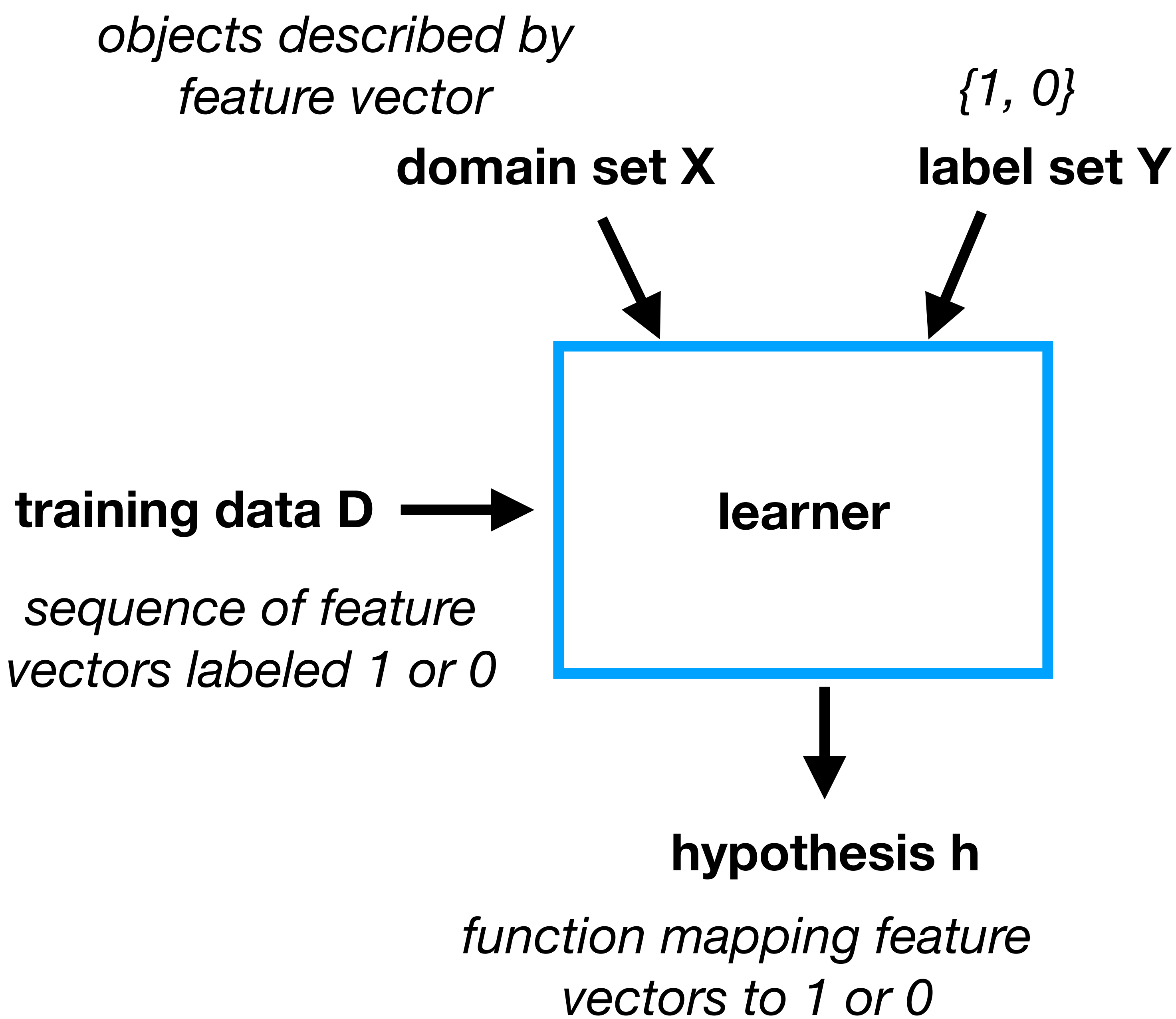
Inductive bias

- This tension is central to machine learning: we cannot learn **successfully** unless we **restrict** the hypothesis space
- Different learners make different assumptions to achieve learning; these assumptions are also called **inductive bias**
- Learners with stronger bias make more inductive leaps, classifying larger parts of the instance space

Inductive bias: example

	learning	classification	inductive bias
learner 1	store training data in memory	stored label if available, “unknown” otherwise	none
learner 2	CANDIDATE-ELIMINATION	agreed label if all members of the version space agree, “unknown” otherwise	target concept in hypothesis space
learner 3	FIND-S	label given by learned hypothesis	target concept in H & all examples negative unless there is reason to consider them positive

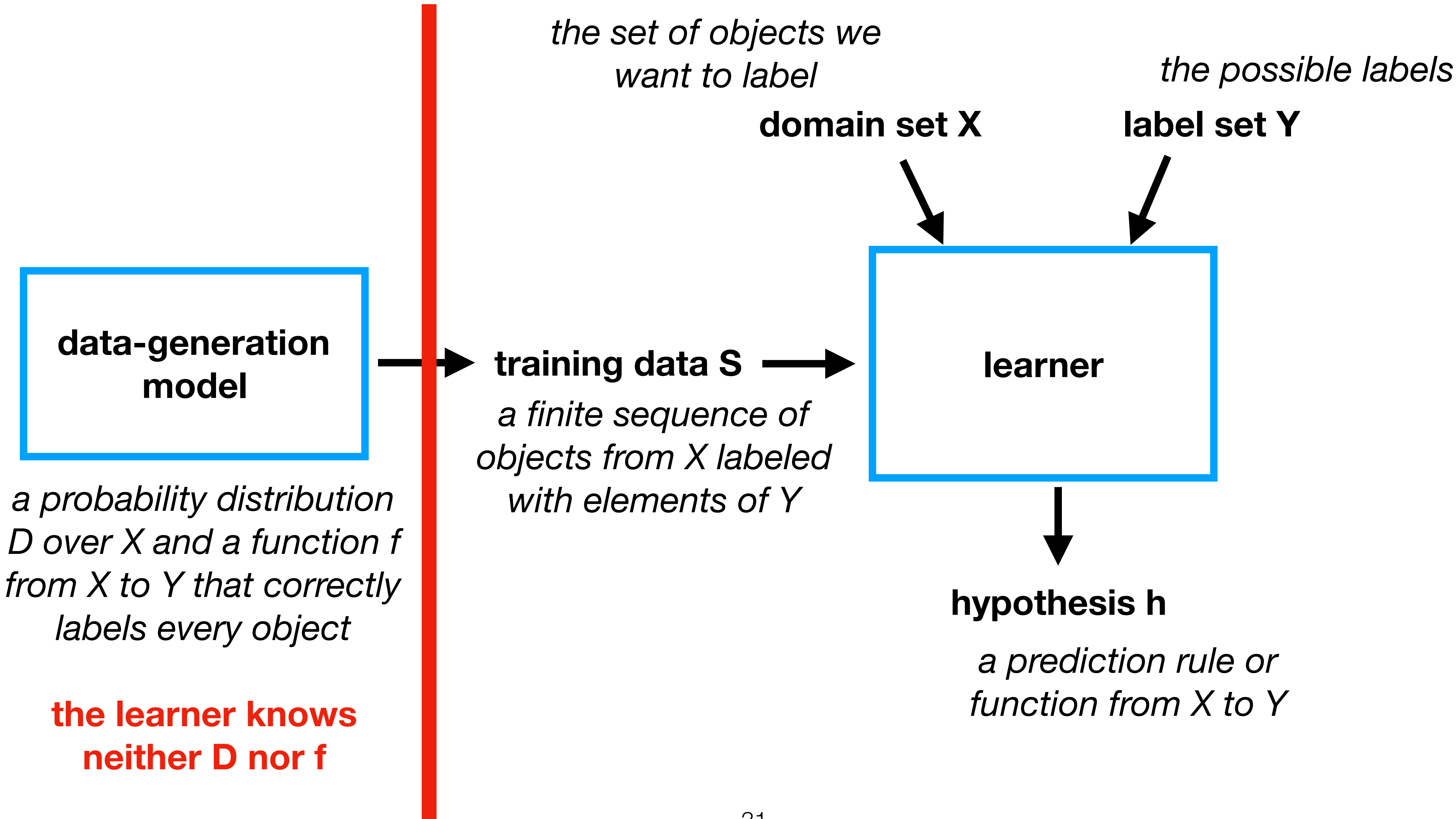
Boolean Concept Learning



Lots of choices when building a learner for a given problem:

- different feature vector representations
- different hypothesis spaces
- different learning algorithms with different inductive bias

The Statistical Learning Framework



Measure of success

- **error** of a hypothesis h = probability of h assigning a wrong label to a random object x drawn from D

- formally:

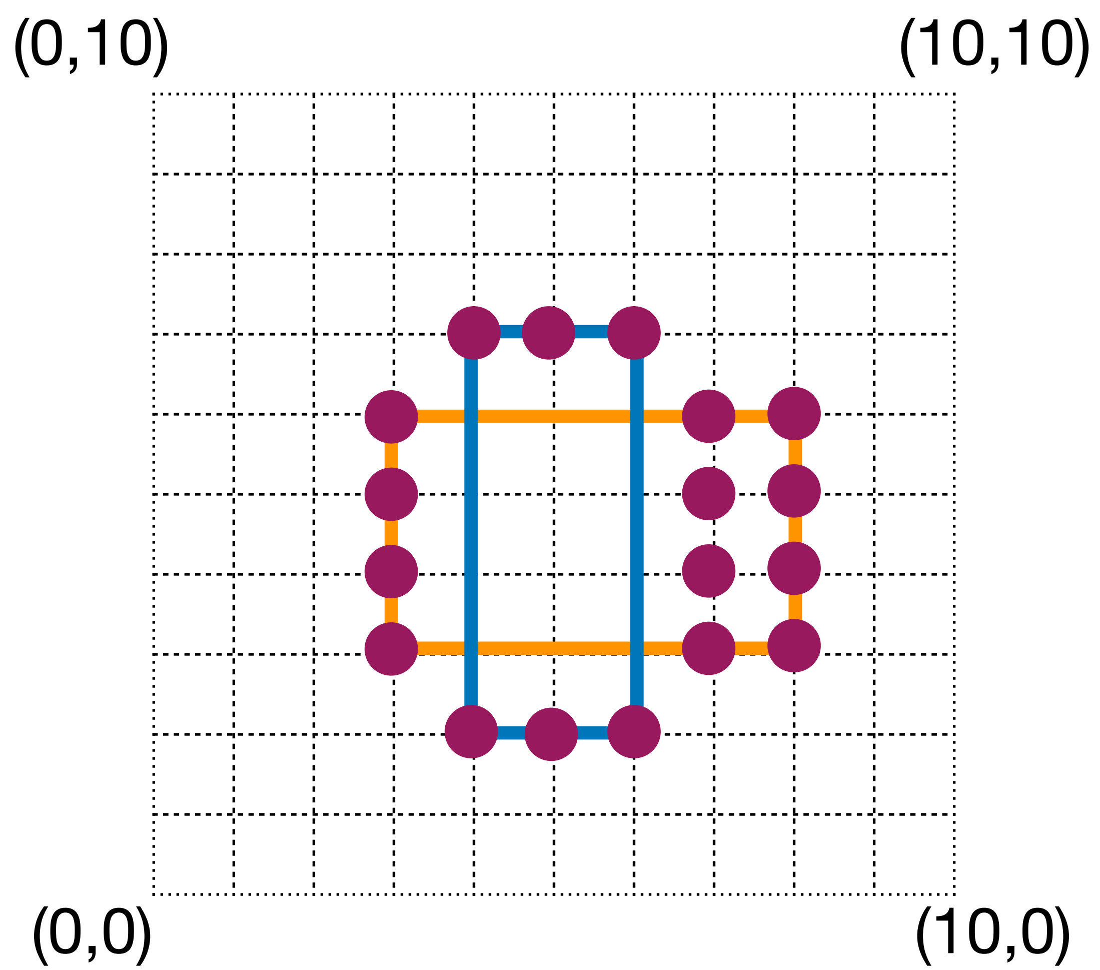
$$L_{D,f}(h) = D(\ \{x \in X \mid h(x) \neq f(x)\}\ )$$

error (or loss) of hypothesis h with respect to distribution D and correct labeling function f

probability according to distribution D of the subset of X where hypothesis h and correct function f disagree

- If the learner would know D and f, it could simply search for the h with minimal $L_{D,f}(h)$

Example



assume D is uniform, i.e., each point on the grid has probability $\frac{1}{121}$

correct function f: $3 \leq x \leq 8 \wedge 3 \leq y \leq 6$

hypothesis h: $4 \leq x \leq 6 \wedge 2 \leq y \leq 7$

$$L_{D,f}(h) = D(\{x \in X \mid h(x) \neq f(x)\}) = \frac{18}{121} = 0.149$$

Empirical Risk Minimisation (ERM)

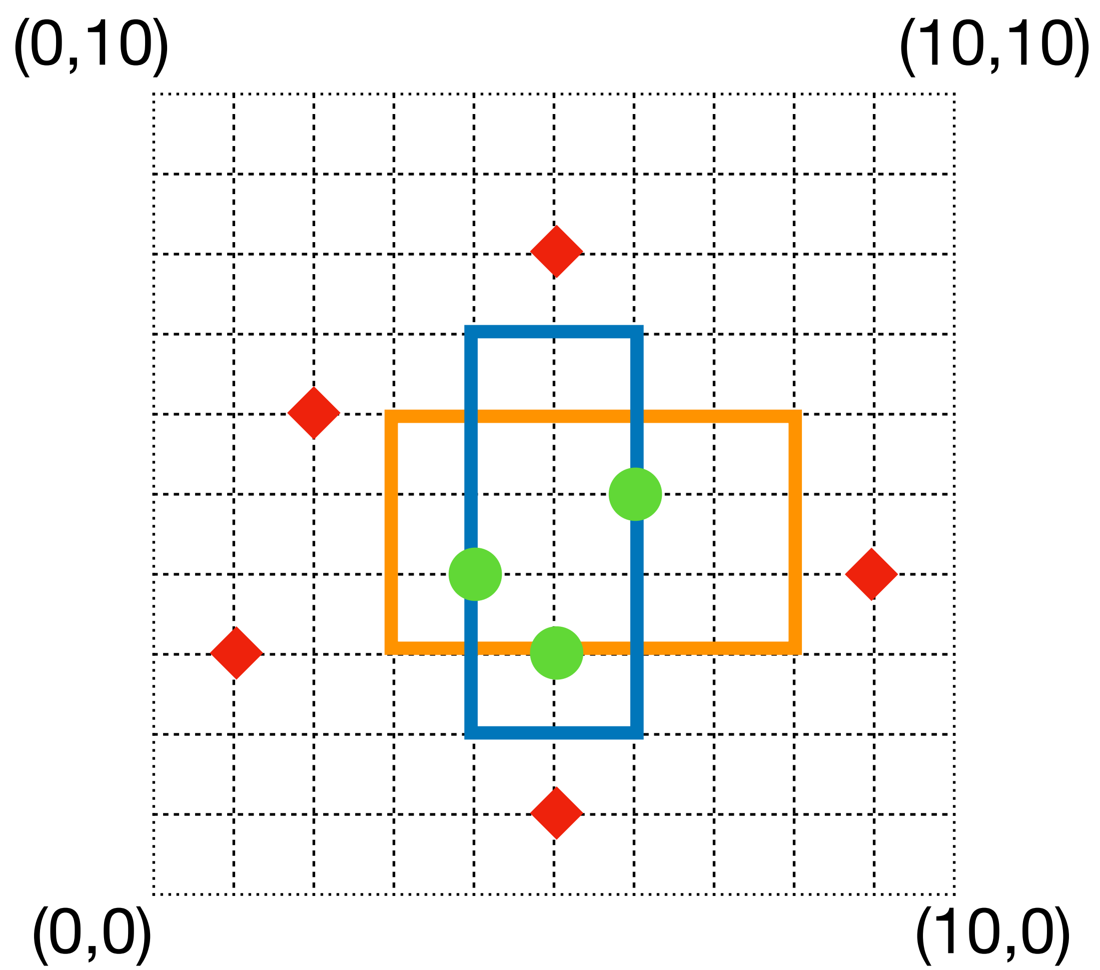
- The **training error** (also called **empirical error** or **empirical risk**) of hypothesis h with respect to training sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is the **fraction** of the training sample h is **not consistent** with, i.e.,

$$L_S(h) = \frac{\left| \{ i \in \{1, \ldots, m\} \mid h(x_i) \neq y_i \} \right|}{m}$$

- The learner can compute this for any given hypothesis!

- An **ERM (empirical risk minimisation) learner** returns a hypothesis h that minimises $L_S(h)$ given S

Example



positive training example

negative training example

assume D is uniform, i.e., each point on the grid has probability $\frac{1}{121}$

correct function f: $3 \leq x \leq 8 \wedge 3 \leq y \leq 6$

hypothesis h: $4 \leq x \leq 6 \wedge 2 \leq y \leq 7$

$L_{D,f}(h) = D(\{x \in X \mid h(x) \neq f(x)\})$

$$= \frac{18}{121} = 0.149$$

$$L_S(h) = \frac{\left| \{ i \in \{1, \ldots, m\} \mid h(x_i) \neq y_i \} \right|}{m}$$

$$= \frac{0}{8} = 0$$

Example ERM learners

	learning	classification
learner 1	store training data in memory	stored label if available, 0 otherwise
learner 2	CANDIDATE-ELIMINATION	agreed label if all members of the version space agree, 0 otherwise
learner 3	FIND-S	label given by learned hypothesis

all have empirical error $\mathbf{L_S(h)=0}$, but true error $\mathbf{L_{D,F}(h)}$ depends on the **unseen positive examples**

Example

days described by

< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy

Warm / Cold

Normal / High

Strong / Weak

Warm / Cool

Same / Change

assume

uniform distribution over days,

true function **f = <?,Warm,?,?,?,?>**,

training data S: <Sunny,Warm,High,Weak,Warm,Same> 1

<Sunny,Warm,High,Weak,Warm,Change> 1

learned hypothesis **h = <Sunny,Warm,High,Weak,Warm,?>**

what is the **empirical error** of h?

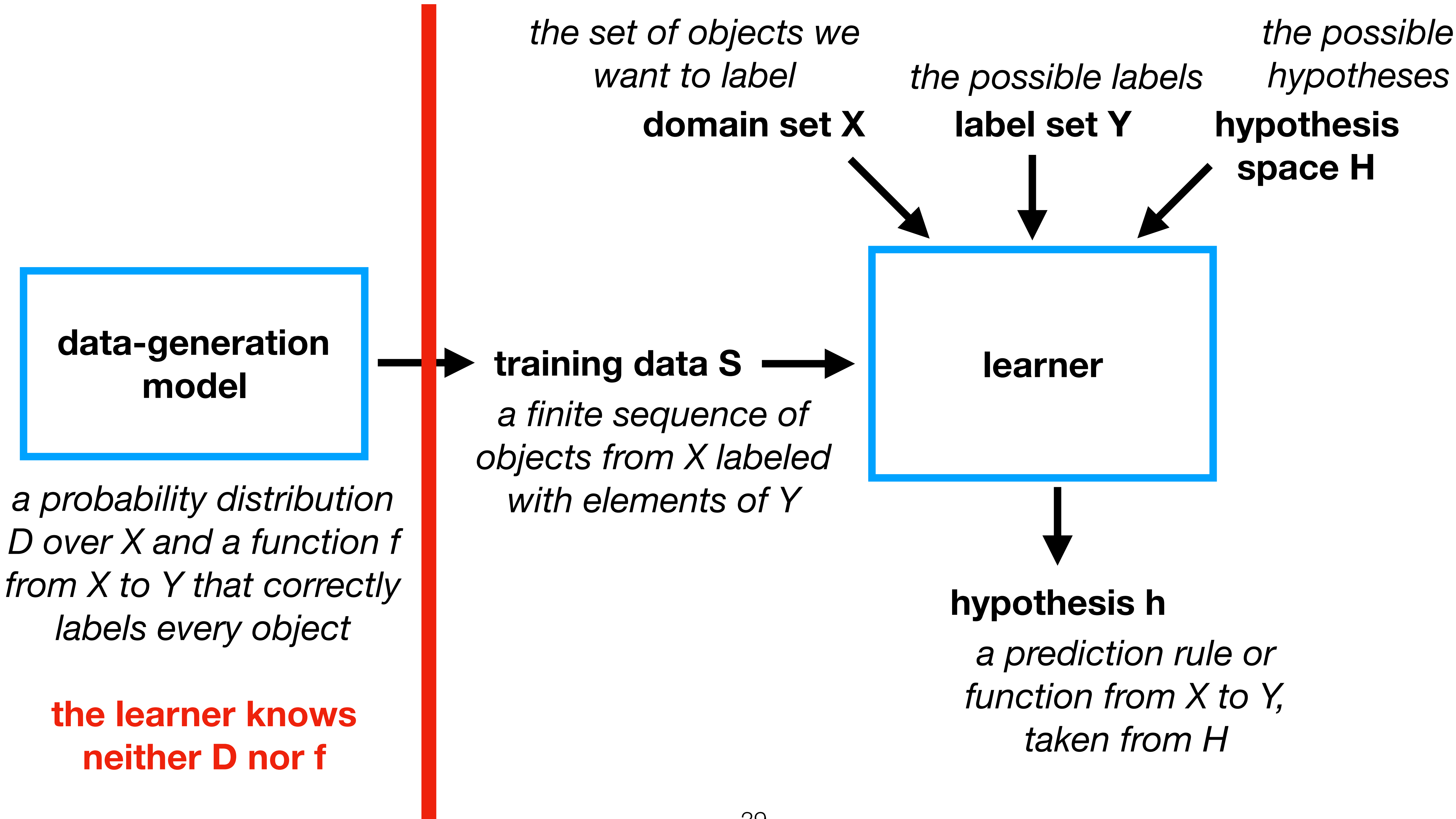
what is the **true error** of h?

this is called **overfitting**: h fits the training data very well, but generalises poorly to unseen examples

Overfitting

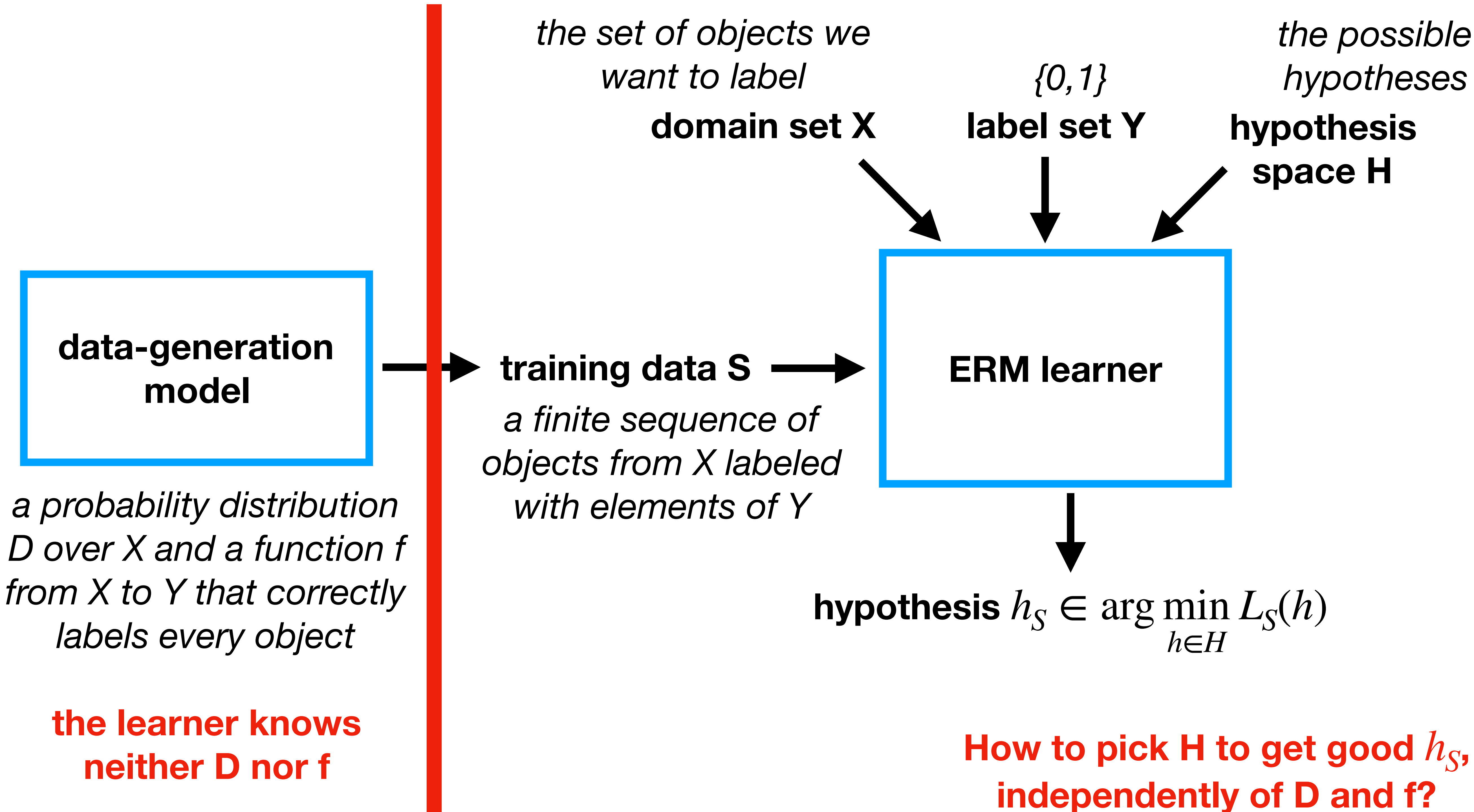
- We saw another example of overfitting earlier: CANDIDATE-ELIMINATION memoizes training examples if we allow it to learn arbitrary Boolean functions
- One way to avoid overfitting is to restrict the hypothesis space before seeing the data

The Statistical Learning Framework



ERM Learning

(Boolean functions)



ERM Learning

- Under the following conditions, ERM will not overfit:
    - H is finite
    - there is a $h \in H$ such that $L_{D,f}(h) = 0$
    - S is "large enough"

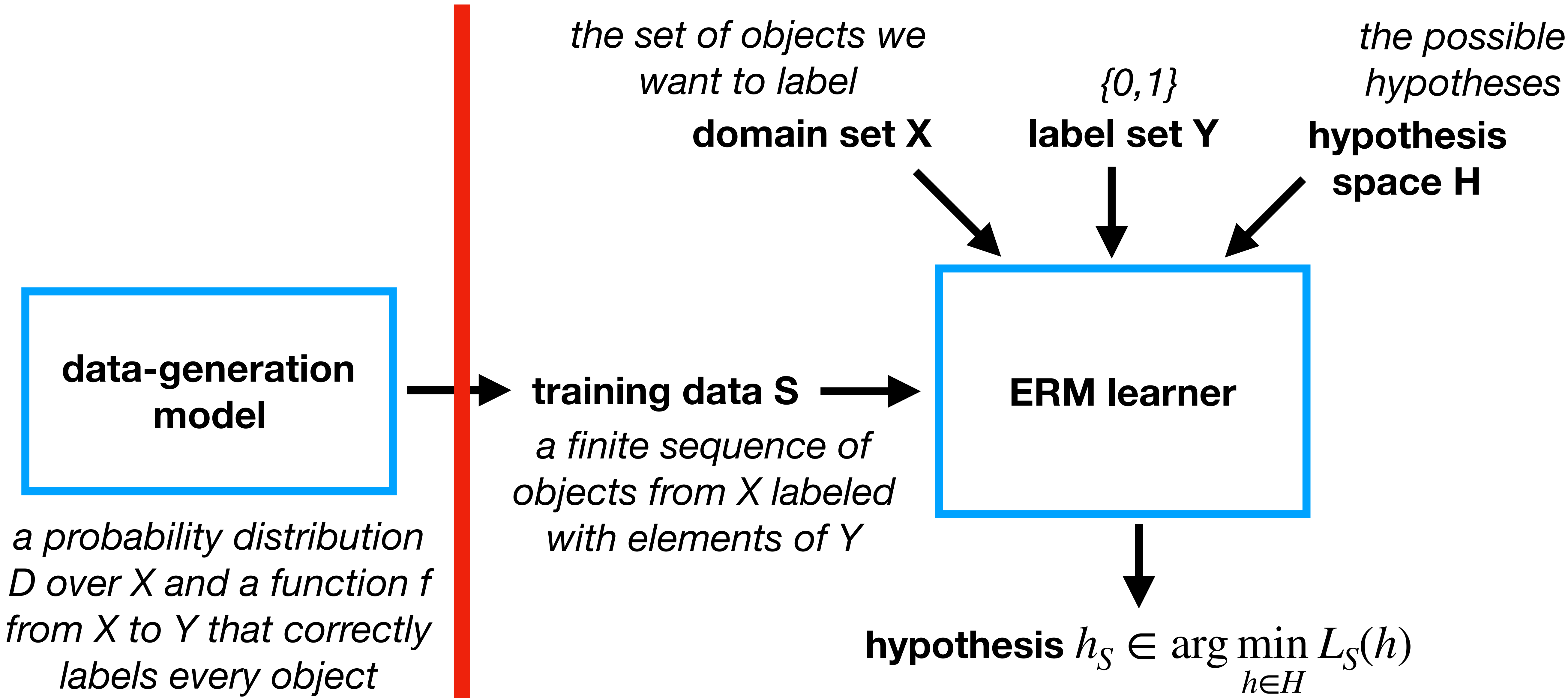
not a necessary condition (more later)

the **realisability** assumption

note: realisability implies $L_S(h_S) = 0$

we'll make this precise next

ERM Learning

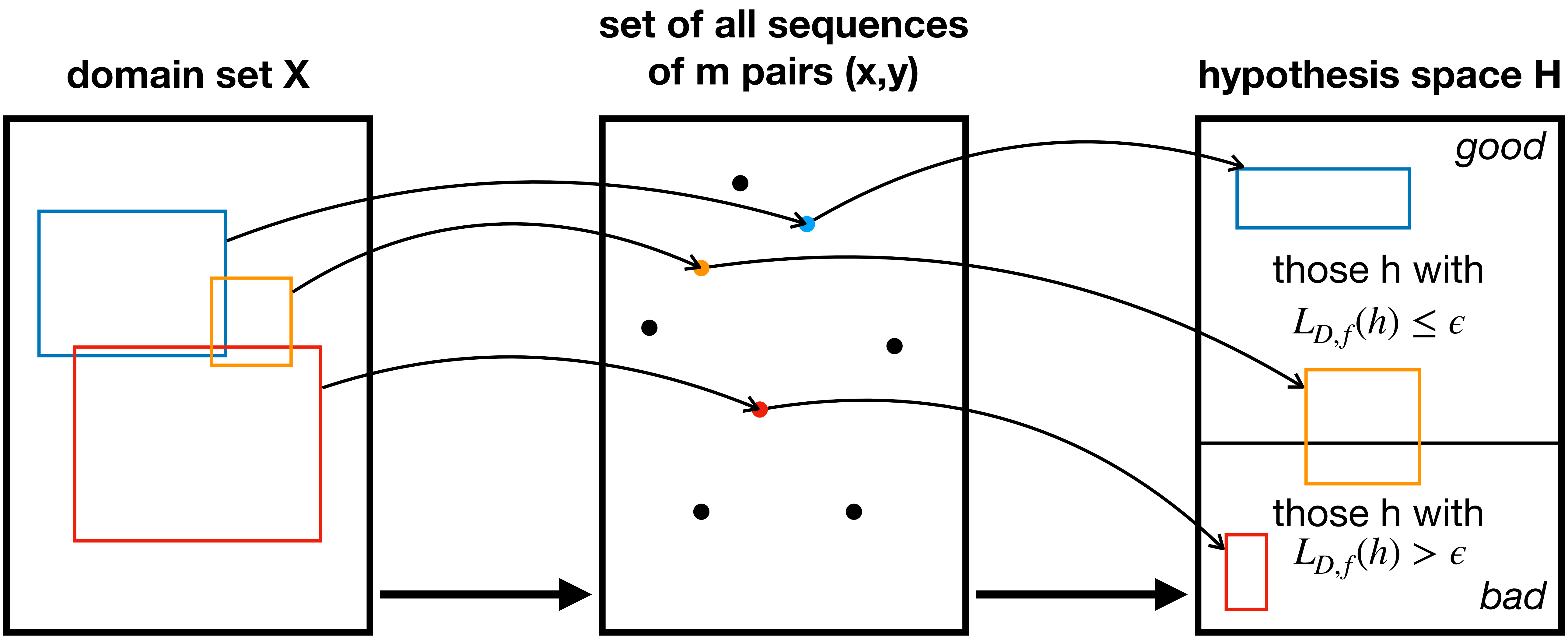


the learner knows neither D nor f

i.i.d. assumption, $S \sim D^m$: S contains m examples that are independently and identically distributed according to D and labeled using f

ERM Learning

- ideally, we'd want ERM to return h_S with $L_{D,f}(h_S) = 0$
- this is not realistic: the random process may give us a misleading S
- instead, we aim for h_S that is **probably approximately correct,** i.e., for which it is very likely that $L_{D,f}(h_S)$ is small for a randomly selected S



unknown probability distribution D generating samples

sample m elements x_i from X according to D and label each with $y_i=f(x_i)$ to generate sequence $((x_1,y_1),\ldots,(x_m,y_m))$

ERM_H selects

$$h_S \in \arg\min_{h \in H} L_S(h)$$

unknown tie-breaking mechanism selecting one of the hypotheses consistent with S

goal: upper-bound the probability that ERM_H selects a bad hypothesis

good news: *bad* is defined using $L_{D,f}(h)$, the probability of h making an error on x drawn from D

Formally

- Fix an **accuracy parameter** ϵ, and consider $L_{D,f}(h_S) > \epsilon$ a **failure** of the learner.

- **Goal**: ensure that the probability of failure (over samples S drawn from D and labeled by f) is at most δ, where we call $(1-\delta)$ the **confidence parameter.**

- That is, given parameters ϵ and δ, we want $P(L_{D,f}(h_S) > \epsilon) \leq \delta$ or equivalently $P(L_{D,f}(h_S) \leq \epsilon) > 1-\delta$

- Question: how large should S be for this to hold?

Basic process

- The learner knows the object set X and hypothesis space H.
- The learner chooses the parameters ϵ and δ.
- The learner does not know the distribution D and function f, but can request an arbitrary but fixed number m of training examples drawn i.i.d. from D and labeled using f.
- How many examples should the learner ask for to achieve $P(L_{D,f}(h_S) > \epsilon) \leq \delta$?

Which m to choose?

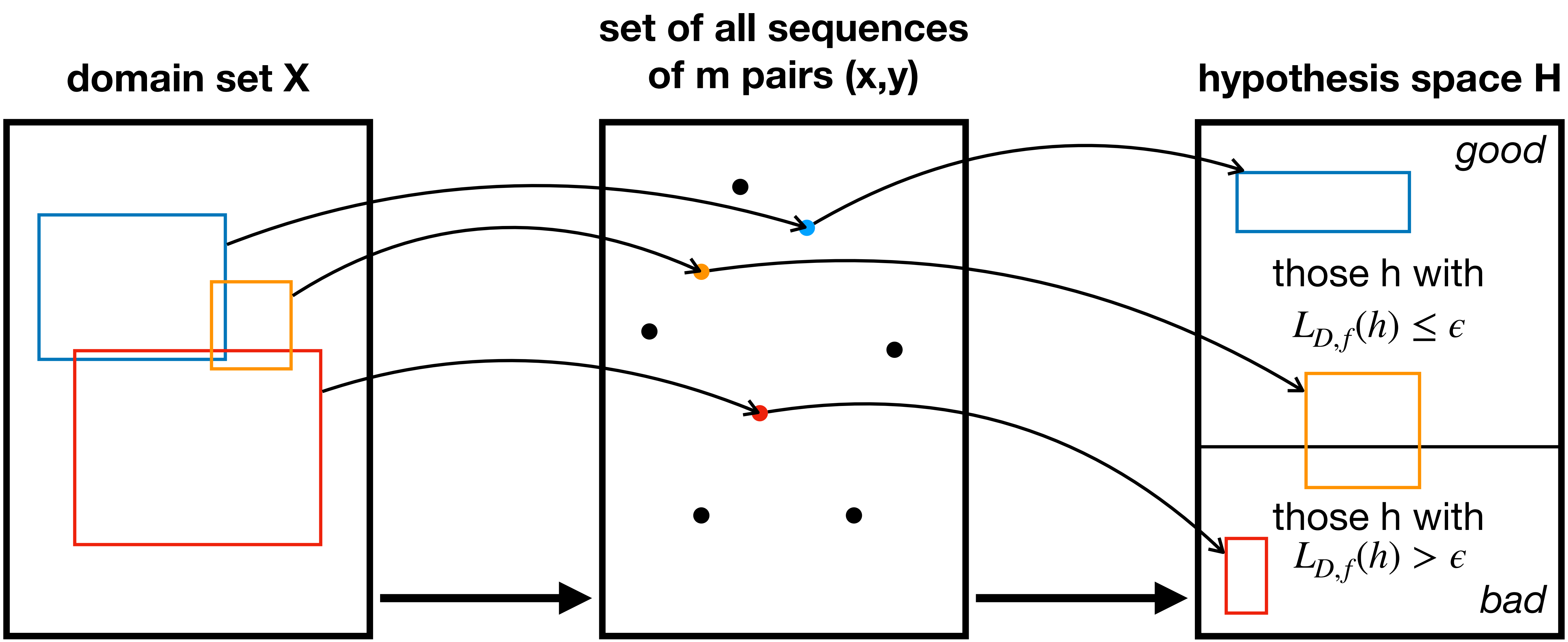
- How many examples should the learner ask for to achieve $P(L_{D,f}(h_S) > \epsilon) \leq \delta$?

- We'll answer this question by
  - providing a function $g(m)$ such that $P(L_{D,f}(h_S) > \epsilon) \leq g(m)$

    preview: $g(m) = |H| e^{-\epsilon m}$

  - rearranging $g(m) \leq \delta$ to obtain an inequality with just m on one side

    preview: $m \geq \dfrac{\log(|H|/\delta)}{\epsilon}$



unknown probability distribution D generating samples

sample m elements x_i from X according to D and label each with $y_i=f(x_i)$ to generate sequence $((x_1,y_1),\ldots,(x_m,y_m))$

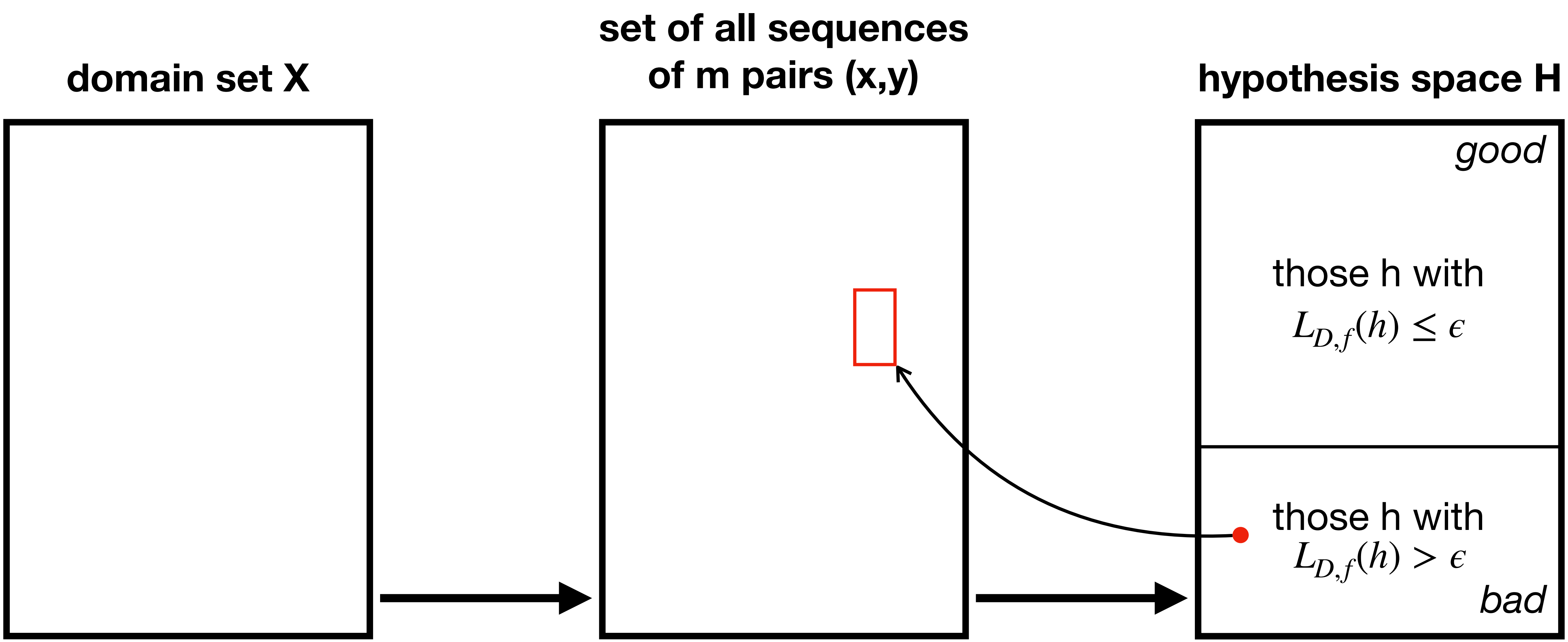
ERM_H selects

$$h_S \in \arg\min_{h\in H} L_S(h)$$

unknown tie-breaking mechanism selecting one of the hypotheses consistent with S

goal: upper-bound the probability that ERM_H selects a bad hypothesis

one such upper bound is the **probability** of getting a sequence $S=((x_1,y_1),\ldots,(x_m,y_m))$ such that **some bad h is consistent** with S



sample m elements x_i from X according to D and label each with y_i=f(x_i) to generate sequence ((x_1,y_1),…,(x_m,y_m))

ERM$_H$ selects

$$h_S \in \arg\min_{h \in H} L_S(h)$$

for a specific bad hypothesis h, what is the probability of getting a sequence S=((x_1,y_1),…,(x_m,y_m)) such that this h is consistent with S?

for a specific bad hypothesis h, what is the probability of getting a sequence $S=((x_1,y_1),\dots,(x_m,y_m))$ such that this <u>h is consistent with S</u>?

for each x_i, $h(x_i)=y_i$

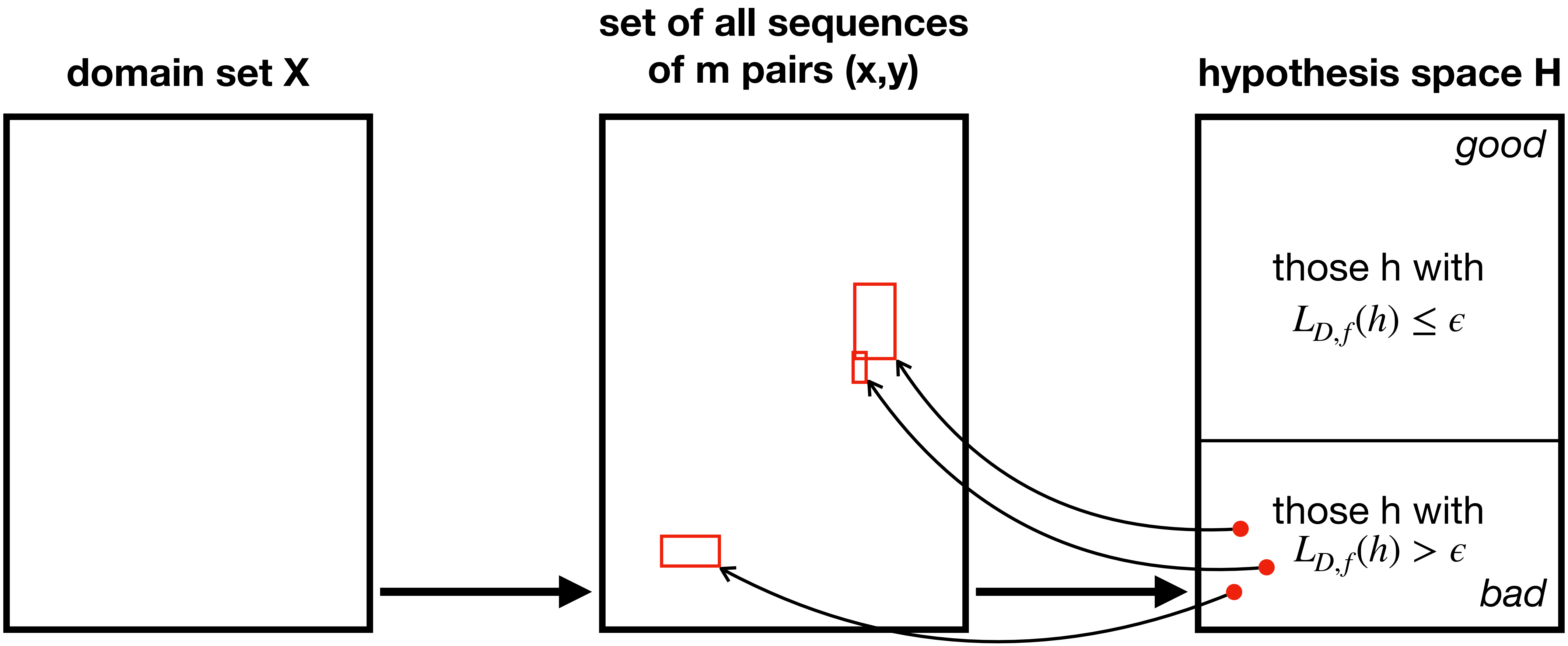
for each x_i, $h(x_i)=f(x_i)$

recall that $L_{D,f}(h)$ is the probability that for x drawn from D, $h(x) \neq f(x)$

thus, $1 - L_{D,f}(h)$ is the probability that for x drawn from D, $h(x) = f(x)$

as each x_i in S is drawn i.i.d. from D,
the probability of getting S consistent with h is $(1 - L_{D,f}(h))^m \leq (1 - \epsilon)^m$

h is bad



sample m elements x_i from X according to D and label each with $y_i = f(x_i)$ to generate sequence $((x_1,y_1),\ldots,(x_m,y_m))$

ERM_H selects

$$h_S \in \arg\min_{h \in H} L_S(h)$$

for a specific bad hypothesis h, what is the probability of getting a sequence $S=((x_1,y_1),\ldots,(x_m,y_m))$ such that this h is consistent with S? $\leq (1-\epsilon)^m$

the probability of getting S consistent with *some* bad h is $\leq |H_{bad}|(1-\epsilon)^m$

$$\leq |H| \underline{(1-\epsilon)^m \leq} |H| \underline{e^{-\epsilon m}}$$

holds for all $\epsilon \in [0,1]$

Which m to choose?

- How many examples should the learner ask for to achieve $P(L_{D,f}(h_S) > \epsilon) \leq \delta$?
- We'll answer this question by
  - providing a function $g(m)$ such that $P(L_{D,f}(h_S) > \epsilon) \leq g(m)$

    preview: $g(m) = |H|e^{-\epsilon m}$
  - rearranging $g(m) \leq \delta$ to obtain an inequality with just m on one side

    preview: $m \geq \dfrac{\log(|H|/\delta)}{\epsilon}$

ERM Learning

- Let H be a finite hypothesis space of Boolean functions on X, $\delta \in [0,1]$, $\epsilon \in [0,1]$, and m an integer satisfying $m \geq \dfrac{\log(|H|/\delta)}{\epsilon}$.

- Then, for any distribution D over X and any labeling function f for which the realisability assumption holds, with probability of at least $1 - \delta$ over the choice of an i.i.d. sample S of size m, we have that for every ERM hypothesis h_S it holds that $L_{D,f}(h_S) \leq \epsilon$.

*That is, for sufficiently large m, any ERM hypothesis is **probably** (with confidence $1 - \delta$) **approximately** (up to an error of ϵ) **correct**.*

Example

$$m \geq \frac{\log(|H|/\delta)}{\epsilon}$$

days described by

< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sky: *Sunny / Cloudy / Rainy*

AirTemp: *Warm / Cold*

Humidity: *Normal / High*

Wind: *Strong / Weak*

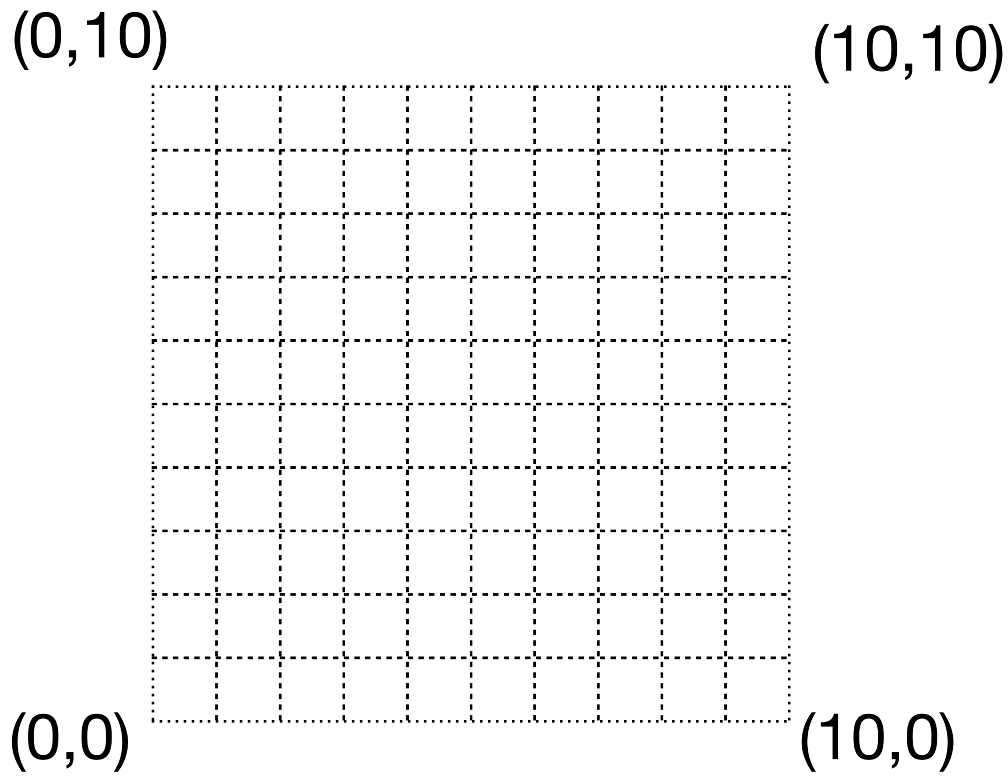
Water: *Warm / Cool*

Forecast: *Same / Change*

δ \ ϵ	**0.05**	**0.01**
0.05		
0.01		

Example

$$m \geq \frac{\log(|H|/\delta)}{\epsilon}$$



δ \ ϵ	**0.05**	**0.01**
0.05		
0.01		

PAC Learnability

- PAC = **P**robably **A**pproximately **C**orrect

- A hypothesis class H is **PAC learnable** if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution D over X, and for every function $f : X \to \{0,1\}$, if the realisability assumption holds w.r.t. H, D, f, then if given $m \geq m_H(\epsilon, \delta)$ i.i.d. examples generated by D and labeled by f, the algorithm returns a hypothesis h such that with probability at least $1 - \delta$ over the choice of the examples, the true error $L_{D,f}(h)$ is at most ϵ.

Sample Complexity

- The function $m_H : (0,1)^2 \to \mathbb{N}$ determines the **sample complexity** of learning H, i.e., the number of samples needed to guarantee a probably approximately correct solution.

- More precisely, $m_H(\epsilon, \delta)$ is the minimal integer the satisfies the requirements of PAC learning

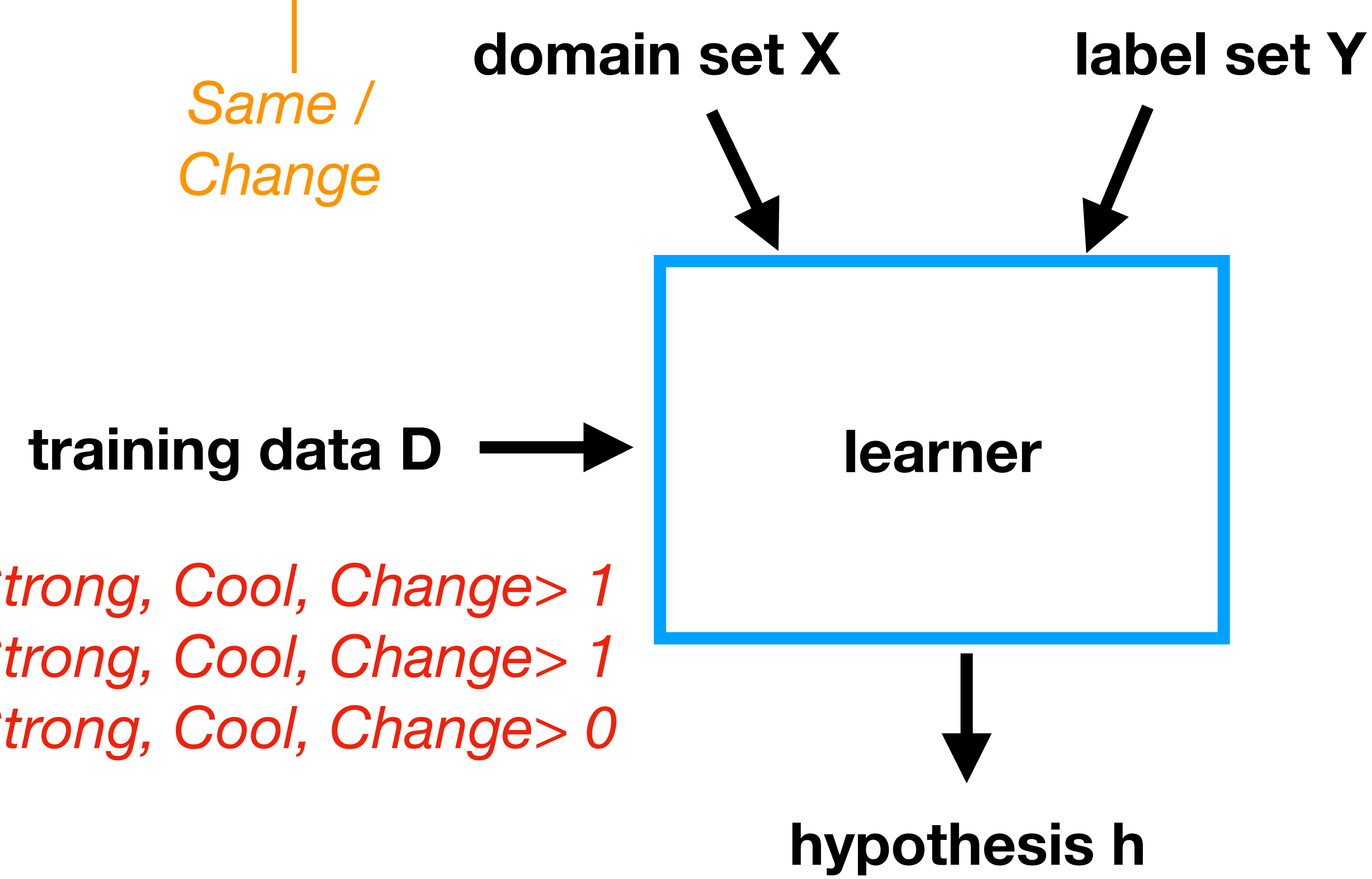
- Thus: every finite H is PAC learnable with sample complexity
$$m_H(\epsilon, \delta) \leq \left\lceil \frac{\log(|H|/\delta)}{\epsilon} \right\rceil$$

No correct h in H

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy
Warm / Cold
Normal / High
Strong / Weak
Warm / Cool
Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day



<Sunny, Warm, Normal, Strong, Cool, Change> 1
<Cloudy, Warm, Normal, Strong, Cool, Change> 1
<Rainy, Warm, Normal, Strong, Cool, Change> 0

conjunction of attribute constraints

ERM Learning

data-generation model

a probability distribution D over X and a function f from X to Y that correctly labels every object

the learner knows neither D nor f

the set of objects we want to label

domain set X

{0,1}

label set Y

the possible hypotheses

hypothesis space H

training data S

a finite sequence of objects from X labeled with elements of Y

ERM learner

hypothesis $h_S \in \arg\min_{h \in H} L_S(h)$

i.i.d. assumption, $S \sim D^m$ **:** S contains m examples that are independently and identically distributed according to D and labeled using f

ERM Learning

with randomly labeled examples

data-generation model

a probability distribution D *over* $X \times Y$

the learner does not know D

training data S

a finite sequence of objects from X labeled with elements of Y

the set of objects we want to label
domain set X

{0,1}
label set Y

the possible hypotheses
hypothesis space H

ERM learner

hypothesis $h_S \in \arg\min_{h \in H} L_S(h)$

i.i.d. assumption, $S \sim D^m$ **:** S contains m examples that are independently and identically distributed according to D

New data generation model

- We now consider a distribution D over labeled objects, e.g., $D((x, y)) = D_X(x) \cdot D_Y(y \mid x)$
- Advantages:
    - can be a more realistic model of the world
    - can handle cases violating the realisability assumption
- Adapt the definition of true error to $L_D(h) = D(\{(x, y) \mid h(x) \neq y\})$
- Goal: a hypothesis that probably approximately minimises $L_D(h)$

The Bayes optimal predictor

- For any D over $X \times \{0,1\}$, the best labeling function is
$$f_D(x) = \begin{cases} 1 & \text{if } P(y=1 \mid x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$
- best = no other $g : X \to \{0,1\}$ has lower error
- but we do not know D …
- instead, we'll aim to learn a predictor whose error is not much larger than the best error in a given class of predictors

Agnostic PAC Learnability

- A hypothesis class H is **agnostic PAC learnable** if there exists a function $m_H : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, for every distribution D over $X \times Y$, if given $m \geq m_H(\epsilon, \delta)$ i.i.d. examples generated by D, the algorithm returns a hypothesis h such that with probability at least $1 - \delta$ over the choice of the examples, the true error $L_D(h)$ is at most ϵ larger than the lowest true error of any hypothesis in H, i.e., $L_D(h) \leq \min_{h' \in H} L_D(h') + \epsilon$.

Remarks

- Agnostic PAC learnability generalises PAC learnability beyond the realisability assumption.
- The whole setup can also be generalised beyond Boolean concept learning *(see the book if interested)*
- The original definition of PAC learnability by Valiant also imposes conditions on the time the algorithm needs to find an answer *(we'll get back to this)*

For next week

- **Mandatory**: revise today's material
    - relevant textbook chapters:
        - Shalev-Shwartz & Ben-David: chapters 2 & 3
        - Mitchell: chapter 2
- **Optional**: look forward
    - Read Shalev-Shwartz & Ben-David, chapters 5 and 6 (excluding proofs), with the following questions in mind:
        - What are the key concepts and ideas introduced?
        - How do they relate to the material covered already?