1 Function approximation

Reducing to a search where goal is to find h

Consider a sample $D \subset \mathcal{X} \times \mathcal{Y}$ where the data is generated by a process, and $\mathcal{X}$ and $\mathcal{Y}$ can be seen as input and output (or target) respectively:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

The goal of function approximation is to find (or learn) a function $h \in \mathcal{H}$ where $h : \mathcal{X} \to \mathcal{Y}$ such that given an x, it "well-approximates" the corresponding y value.

Depending on whether the target is continuous or discrete, the task is called *regression* or *classification* respectively.

Examples include recognition of handwritten digits, prediction of house prices in Christchurch, and prediction of Students' performance in a course.

There are different ways of approaching this problem depending on what assumptions are made. For example y_i can be assumed to be generated by

$\partial i = f(x_i)$

$\partial i \sim F_{0|x}(x_i)$

- a function (i.e. a deterministic/exact process); or

  process generating a deterministic output that doesn't vary.
  Same input -> same output

- a probability distribution (i.e. a stochastic process).

2 A logical approach to binary classification

In binary classification $\mathcal{Y}$ is $\{True, False\}$ (or $\{0, 1\}$, or $\{positive, negative\}$). All classification problems can be cast as one or more binary classification problems.

eg house prices, just looking at area as input,

The target is $y = f(x)$ where f is a deterministic (but unknown) Boolean-valued function: $f : \mathcal{X} \to \{True, False\}$. In other words, f is an unknown *predicate*.

same input can get diff outputs.

When a sample is given, it is assumed that the sample is generated as:

so need to take in more input and probalistic approach

$$D = \{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\}$$

Let $\mathcal{F}$ be the set of all predicates (or functions) that can be defined over $\mathcal{X}$. In this setting we have:

Don't go to all set of F, as F is large
Can't just get f as f is unknown

H, hypothesis $\quad \mathcal{H} \subseteq \mathcal{F}$

$2^{|X|}$

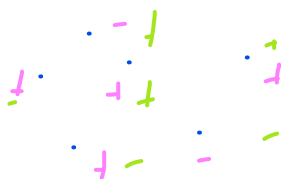**Question** What is the cardinality (size) of $\mathcal{F}$ if $\mathcal{X}$ is finite?

Answer: 2^(size of X)
F is set of predicates, so binary true or false

X in our universe is all possible inputs

eg one function in green, another in pink

efficiency and size will become a consuming problem

1

2.1 Partial order of predicates

We define the binary relation *less general or equal* over $\mathcal{F}^2$ (or $\mathcal{H}^2$) as the following:

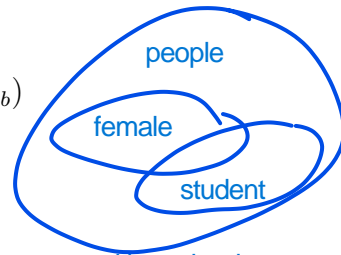$$h_a \underset{g}{\leq} h_b \quad \equiv \quad \forall x \in \mathcal{X} \ \ h_a(x) \implies h_b(x)$$
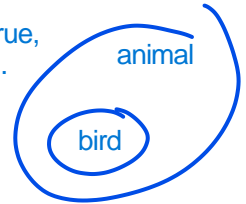
The relation *less general or equal* has the following properties:

- reflexive: $\forall h \in \mathcal{H} : \quad h \underset{g}{\leq} h$

- anti-symmetric: $\forall h_a, h_b \in \mathcal{H} : \quad ((h_a \underset{g}{\leq} h_b \wedge h_b \underset{g}{\leq} h_a) \implies h_a = h_b)$

- transitive: $\forall h_a, h_b, h_c \in \mathcal{H} : \quad ((h_a \underset{g}{\leq} h_b \wedge h_b \underset{g}{\leq} h_c) \implies h_a \underset{g}{\leq} h_c)$

Therefore the relation establishes a partial order over $\mathcal{F}$ (or $\mathcal{H}$).

We define the *support* of a hypothesis to be the set of elements in $\mathcal{X}$ that are accepted by the hypothesis. In other words, $supp : \mathcal{H} \to 2^{\mathcal{X}}$ and
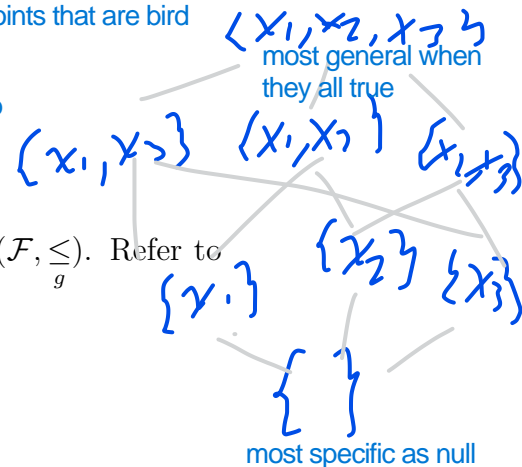
$$supp(h) = \{x \in \mathcal{X} : h(x)\} \ .$$

As a result we have

$$h_a \underset{g}{\leq} h_b \quad \implies \quad supp(h_a) \subseteq supp(h_b) \ .$$

Example suppose $\mathcal{X} = \{x_1, x_2, x_3\}$. Draw the lattice of $(\mathcal{F}, \underset{g}{\leq})$. Refer to the elements of $\mathcal{F}$ by their support sets.

2.2 Consistency and learning

2.2.1 Consistency

For an $h \in \mathcal{H}$, we define consistency as the following:

$$consistent(h, D) \quad \equiv \quad \forall (x, y) \in D : \ h(x) = y$$

Example Write a Python function `consistent(h, D)`.

2

$$2^{(|\mathcal{X}| - |D|)}$$

D in reality could be inconsistent, but in this example we assume not

if D was size of X, if D was empty. are the extremes

Question How many functions in $\mathcal{F}$ can be consistent with D? Half?

Answer: 2^[X - cardinality of D]

2.2.2 Learning one hypothesis

One (inefficient) way of learning is by exhaustively searching $\mathcal{H}$.

finding a h from H

Example Write a Python function `learn_a_hypothesis_by_enumeration(H, D)`. parameters are H and D, for consistent it was h, D

Is it possible for the returned classifier to misclassify a new input?

as only concern with consistency with Dso can misclassify if data not seen before, not from D. a real concern

Often the elements of $\mathcal{H}$ have the same type and encoding. If $\mathcal{R}$ is the set of all *codes* for $\mathcal{H}$, there is a function to map a code to a hypothesis: Use R as set of codes

expecting H to be passed as already made set

$$decode : \mathcal{R} \to \mathcal{H}$$

decode is a function that takes a code (an argument) in R, and return function in H.

Often there is no bijection between $\mathcal{H}$ and $\mathcal{R}$.

-> means it is a mapping from a r in R to a h in H?

The elements of $\mathcal{H}$ are rarely generated explicitly; they come in to existence as necessary. The search is performed in $\mathcal{R}$. This is similar to the way in which vertecies and edges in implicit graphs come to existence.

d for the weight, and 1for the bias

$$R = |\mathbb{R}^{d+1}$$

Example Give $\mathcal{R}$ for the following hypothesis classes:

Decode take the weight and bias and return true or false

Perceptron

- $\mathcal{H}$ is the set of perceptrons over $\mathbb{R}^d$ represented by weights and bias.

Decode is part of the design

- $\mathcal{H}$ is the set of naive Bayes networks where the objects are described by d Boolean input variables and the class label is binary.

c

d number of conditional property...

diff weights can produce same perceptron. sometimes theres lots of effort to remove this redundancy

$$[0,1]^{2d+1}$$

integral, not a list

3 The version space

The *version space* is defined as
where function is deterministic
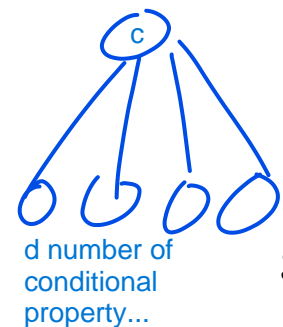
$$VS_{\mathcal{H},D} = \{h \in \mathcal{H} : consistent(h, D)\}$$

Example Write a Python function `learn_all_hypotheses(H, D)` that finds the version space by enumeration.

all elements that are consistent with H will be return
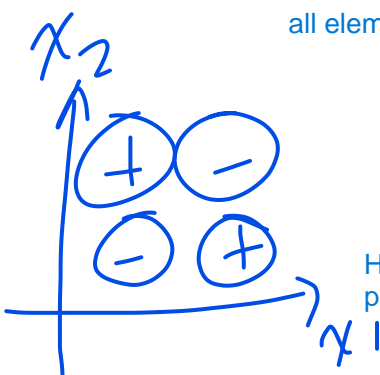
H is always a subset of F

x_2

x_1

H can be though of all the lines that you can draw through this graph space except no line is possible

Example   The entire version space can be used for prediction.

most general

Set G

Set S
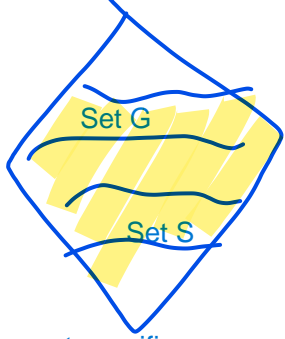
most specific

```python
def predict(VS, x):
    if len(VS) == 0:
        raise ValueError("No hypothesis!")
    if all(h(x) for h in VS):
        return "All positive"
    elif not any(h(x) for h in VS):
        return "All negative"
    else:
        return "positive count:{}, negative count:{}".format(# complete)
```

3.1   Specifying VS with its boundaries

We define two sets:

$$S_{H,D} = \{h \in \mathcal{H} : consistent(h, D) \wedge \forall h' \in \mathcal{H} : (consistent(h', D) \implies \neg(h' \underset{g}{<} h))\}$$

$$G_{H,D} = \{h \in \mathcal{H} : consistent(h, D) \wedge \forall h' \in \mathcal{H} : (consistent(h', D) \implies \neg(h \underset{g}{<} h'))\}$$

See Figure 1 for an illustration of the version space and the sets S and G.

Theorem 1.

$$\forall h \in \mathcal{H} : \quad (h \in VS_{H,D} \Leftrightarrow \exists h_s \in S_{H,D}, \ \exists h_g \in G_{H,D} : h_s \underset{g}{\leq} h \underset{g}{\leq} h_g)$$

4

D is number of positive and negative training examples.

imagine each point has a coordinate of x and y

hypothesis space is all rectangles.

if something falls inside rectangle, it will return true.

cannot have a rectangle smaller than SB that can be consistent with the data.

GB is the biggest it can be before it will return positive for something thats actually negative

GB

SB

to think in terms of a perceptron

+ + +

+ + +

+

most specifc

most general

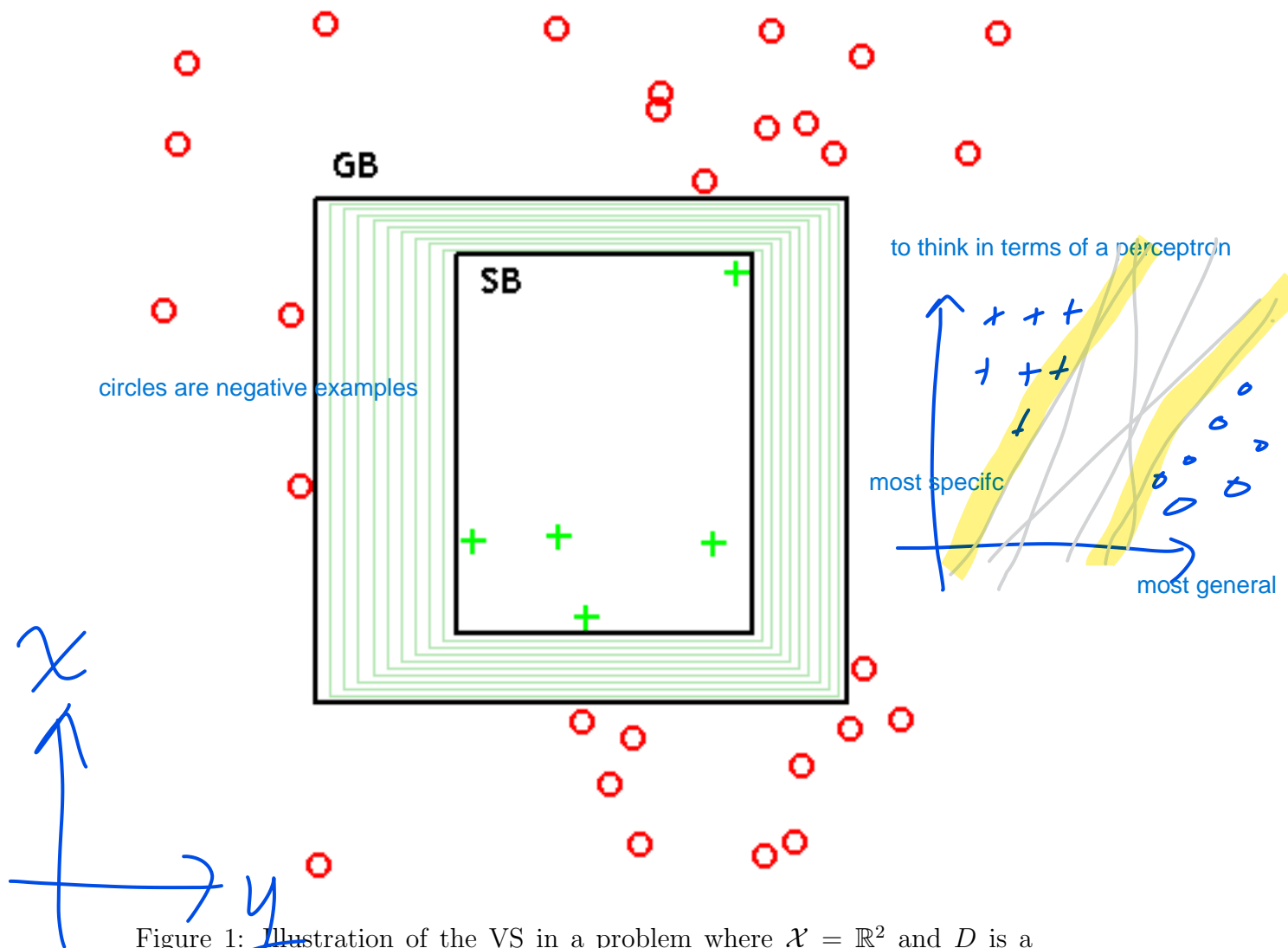circles are negative examples

x

y

Figure 1: Illustration of the VS in a problem where $\mathcal{X} = \mathbb{R}^2$ and D is a collection of positive points (depicted by '+') and negative points (depicted by 'o'). $\mathcal{H}$ is the set of all rectangles in $\mathbb{R}^2$ where a point inside a rectangle will be predicted as positive. The version space can be specified by two sets S and G, each having one member in this example.

5

3.2 Candidate Elimination Algorithm (CEA)

The idea is to incrementally update S and G instead of generating VS by filtering through $\mathcal{H}$

```
Initialize G to the set of most-general hypotheses in H
Initialize S to the set of most-specific hypotheses in H
For each training example, d, do:
    If d is a positive example then:
        Remove from G any hypotheses that do not match d
        For each hypothesis s in S that does not match d
            Remove s from S
            Add to S all minimal generalizations, h, of s such that:
                1) h matches d
                2) some member of G is more general than h    *** more general or equal
        Remove from S any h that is more general than another hypothesis in S
    If d is a negative example then:
        Remove from S any hypotheses that match d
        For each hypothesis g in G that matches d
            Remove g from G
            Add to G all minimal specializations, h, of g such that:
                1) h does not match d
                2) some member of S is more specific than h    ** more specific or equal
        Remove from G any h that is more specific than another hypothesis in G
```

4 Example Problem and Representation

4.1 Representation: Conjunction of Constraints

If $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \mathcal{X}_d$, then we define the representation of *conjunction of constraints* (i.e. the set of all codes) to be

$$\mathcal{R} = \prod_{i=1}^{d} (\mathcal{X}_i \cup \{?, \varnothing\})$$

Each code is a tuple of the from $(c_1, c_2, \ldots, c_d)$ where c_i is a constraint which is either:

- ? (a wild card): meaning that the value of the i-th feature can be anything; or

6

- a specific value from the domain of the i-th feature: meaning that the value of the i-th feature must be exactly the given specific value; or

- $\varnothing$: meaning that no value for the i-th feature is acceptable.

Given a tuple of constraints, the corresponding hypothesis is the conjunction of all the d constraints. More formally let

$$satisfied(c_i, x_i) \mapsto c_i \neq \varnothing \wedge (c_i =? \vee c_i = x_i)$$

then

must be a wild card or exactly that value

$$decode(c)(x) \mapsto satisfied(c_1, x_1) \wedge satisfied(c_2, x_2) \wedge \cdots \wedge satisfied(c_d, x_d)$$

take a contraint that behave in this way. last question in quiz

4.2 Example problem

This problem is about when a certain sport is enjoyable. The input space is defined by 6 attributes: $\mathcal{X} = Sky \times Temp \times Humidity \times Wind \times Water \times Forecast$. The target is a Boolean: $\mathcal{Y} = \{Yes, No\}$. The domain of the variables are as the following:

- $Sky = \{$Sunny, Cloudy, Rainy$\}$

- $Temp = \{$Warm, Cold$\}$

- $Humidity = \{$Normal, High$\}$

- $Wind = \{$Strong, Weak$\}$

- $Water = \{$Warm, Cool$\}$

- $Forecast = \{$Same, Change$\}$

Questions

1. What is the cardinality of $\mathcal{X}$? *3 * 2 * 2 * 2 * 2 * 2 = 3 * 2^5 = 96*

2. What is the cardinality of $\mathcal{F}$? *2^96*

3. If we use the conjunction of constraints as our hypothesis class, what is the size of R? What is the size of $\mathcal{H}$?

   *|R| = (3+2) * (2+2)^5*

   7

   if you have a null, then entire hypothesis is null.
   *|H| = 1 + (3+1) * (2+1)^5*

   |H| is much smaller compare to 2^96
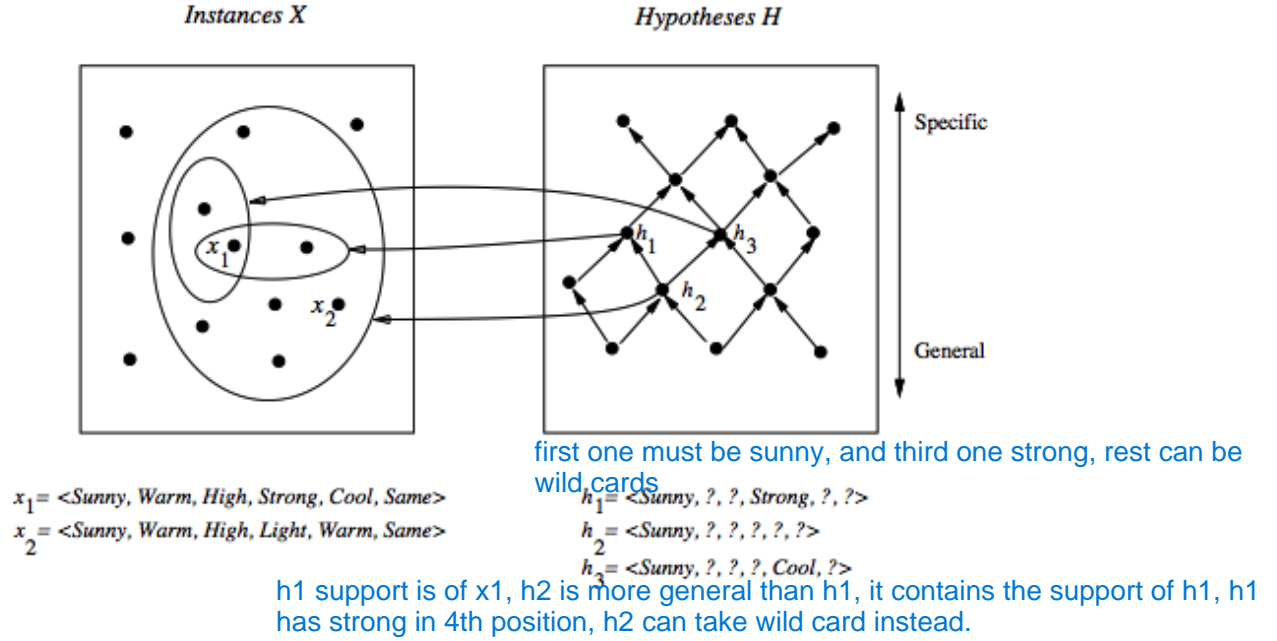   *|H| is almost always smaller than |R|*

   best case |H| = |R|

Figure 2: The relationship between a few hypotheses each expressed as conjunction of constraints and a few points in the input space.

4.3   Applying CEA

Consider the following sample D:

Sky	Temp	Humid	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Figure 3 shows the trace of CEA using the conjunction of constraints and the above D.

The resulting VS is specified with only S and G. See Figure 4.

If S and G become empty sets after the execution of the algorithm, then there is no hypothesis (predicate) in $\mathcal{H}$ that is consistent with the data (i.e. no learning).

from the bottom, the begining G node has wildcard for everything. hypothessi that accept everything and return true for everything.

top S0, ust need one null to return null

able to match these first two

S_0 <Ø, Ø, Ø, Ø, Ø, Ø>

d_1: <Sunny, Warm, Normal, Strong, Warm, Same, Yes>

d_2: <Sunny, Warm, High, Strong, Warm, Same, Yes>

S_1 <Sunny, Warm, Normal, Strong, Warm, Same>

d_3: <Rainy, Cold, High, Strong, Warm, Change, No>

$S_2 = S_3$ <Sunny, Warm, ?, Strong, Warm, Same>

d_4: <Sunny, Warm, High, Strong, Cool, Change, Yes>

S_4 <Sunny, Warm, ?, Strong, ?, ?>

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

G_4 <Sunny, ?, ?, ?, ?, ?>   <?, Warm, ?, ?, ?, ?>

TODO why isnt <??? Strong ?? > an option

G_3 <Sunny, ?, ?, ?, ?, ?> <?, Warm, ?, ?, ?, ?> <?, ?, ?, ?, ?, Same>
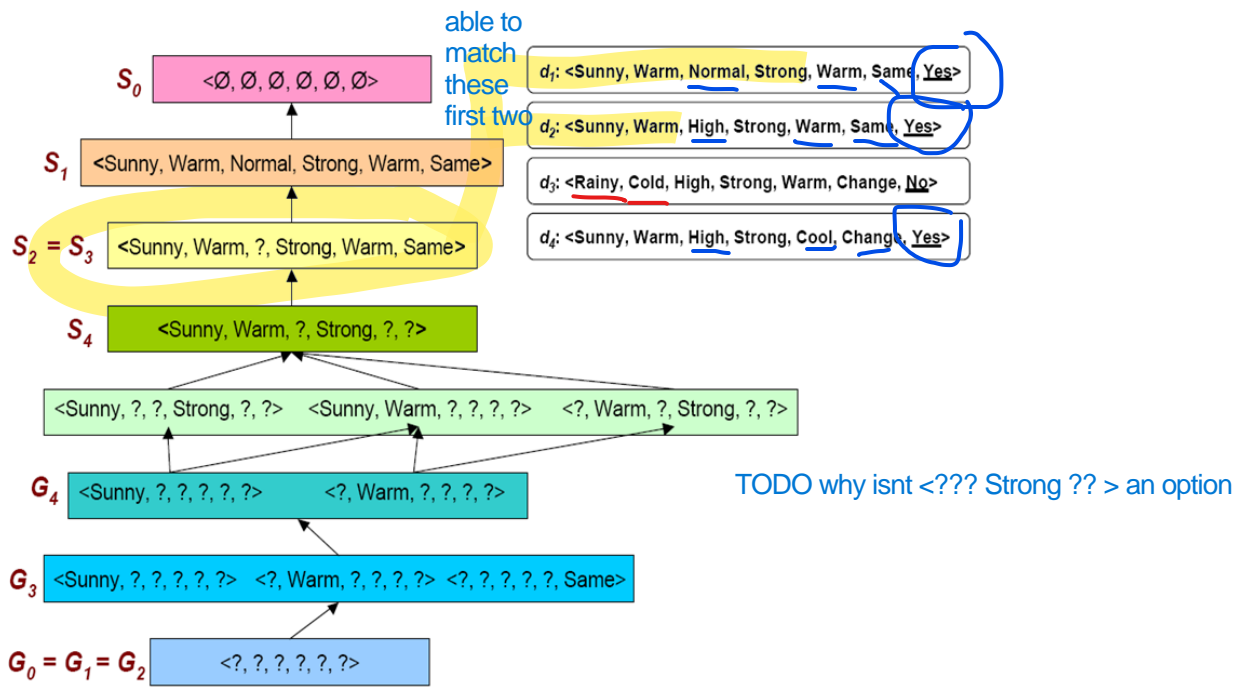
$G_0 = G_1 = G_2$ <?, ?, ?, ?, ?, ?>

Figure 3: The trace of CEA using conjunction of constraints and four training data points.

Making predictions

Using the produced VS, determine how the following cases are classified:

- (Sunny, Warm, Normal, Strong, Cool, Change)

- (Rainy, Cool, Normal, Light, Warm, Same)

- (Sunny, Warm, Normal, Light, Warm, Same)   Y?

- (Sunny, Cold, Normal, Strong, Warm, Same)   Y?

4.4   Language (representation) limitations

What if D is the following set?

9

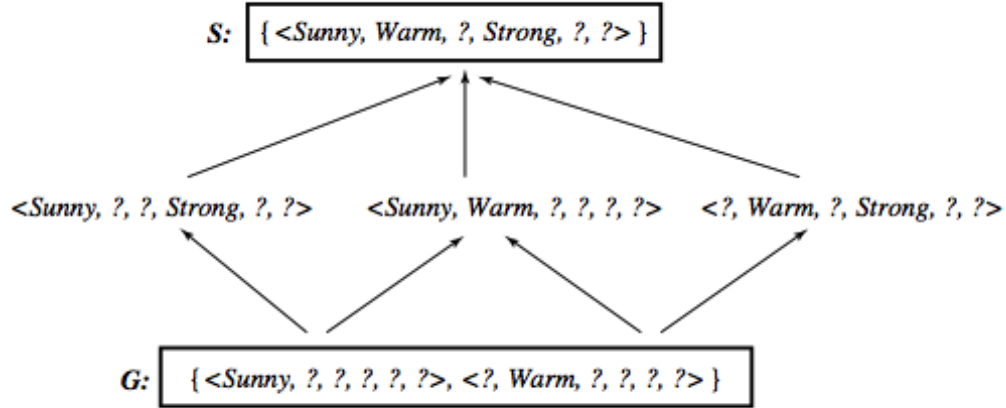Figure 4: The resulting VS which can be specified using only S and G. If necessary, S and G can be used to generate the rest of the hypotheses in the VS.

Sky	Temp	Humid	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Cool	Change	Yes
Cloudy	Warm	Normal	Strong	Cool	Change	Yes
Rainy	Warm	Normal	Strong	Cool	Change	No

- No hypothesis in the form of conjunction of constraints is consistent with data. *because if make first position a wild card it will accept everythig, if makee cloudy then it will be yes for second row but no for first row*

- The hypothesis (?, Warm, Normal, Strong, Cool, Change) is too general.

- Target concept is in a different hypothesis space. For example in space where hypotheses can have disjunction: '(Sky=="Sunny" or Sky=="Cloudy")'

5 Inductive Bias

Any means by which a learning system is limited to choose from only a limited set of functions (hypotheses) is *inductive bias*. There are two major forms of bias:

- Language bias: the representation is such that $\mathcal{H}$ contains only certain functions.

- Search bias: The search algorithm embodies a preference for certain functions and does not consider all options.

5.1 Extending the hypothesis space

We can allow all logical formulas and therefore be able to express any function. What will be the size of $\mathcal{H}$ in this case? What can we learn in this case?

For example, consider $D = \{(x_1, True), (x_2, True), (x_3, True), (x_4, False), (x_5, False)\}$. After running CEA we have:

- $S = \{(x_1 \lor x_2 \lor x_3)\}$

- $G = \{\neg(x_4 \lor x_5)\}$

- We can only classify precisely examples already seen!

- Take a majority vote?

- For any hypothesis h in VS that classifies an unseen x as positive, there is a complementary hypothesis that classifies x as negative.

- Unseen instances are classified positive (and negative) by exactly half of the VS.
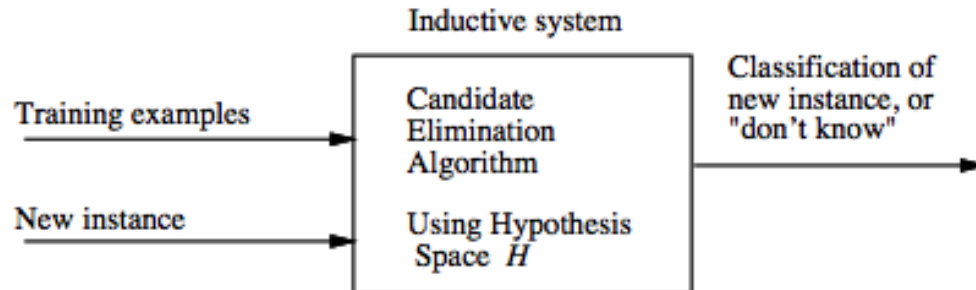
5.2 No inductive inference without a bias

A learner that makes no a priori assumptions regarding the identity of the target concept, has no rational basis for classifying unseen instances.
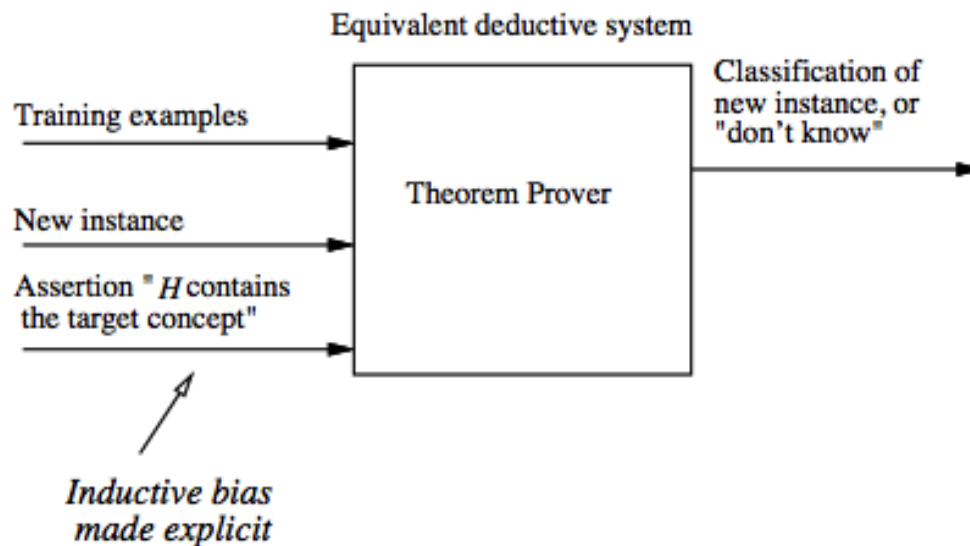
6 Inductive learning as deductive process

Induction is inferring general rules and theories from specific empirical data (this is a broader concept than logical/mathematical induction). The inductive learning process described earlier is depicted in the following diagram.

Inductive system

Training examples ⟶

New instance ⟶

Candidate
Elimination
Algorithm

Using Hypothesis
Space H

⟶ Classification of
new instance, or
"don't know"

Deduction is inferring sound specific conclusions from general rules (axioms) and specific facts. In a logical approach to learning, the inductive process can be achieved using deduction.

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*

7   A probabilistic approach

7.1   Verifying a hypothesis

Given a hypothesis, we want to know the probability of the hypothesis making an error. Let's define a Bernoulli random variable that is 1 when an error happens:

12

$$\mathbb{1}_{h\neq f}(x) = \begin{cases} 1, & h(x) \neq f(x) \\ 0, & otherwise. \end{cases}$$

We define the following quantity as the error of h:

$$E(h) = \mathbb{P}_{\mathcal{X}}[h(x) \neq f(x)] = \mathbb{E}_{\mathcal{X}}[\mathbb{1}_{h\neq f}(x)] \ ,$$

where $\mathbb{P}_{\mathcal{X}}[h(x) \neq f(x)]$ denotes the probability of the event $h(x) \neq f(x)$, and $\mathbb{E}_{\mathcal{X}}[\mathbb{1}_{h\neq f}(x)]$ denotes the expected value of the random variable $\mathbb{1}_{h\neq f}(x)$.

For the following reasons, we cannot directly calculate the above quantity:

- Although in this framework f is fixed, it is unknown to us;

- We do not have access to the entire input space $\mathcal{X}$, but only a sample D from it.

We can however measure the (empirical) error over the sample:

$$E_D(h) = \mathbb{P}_D[h(x) \neq f(x)] = \mathbb{E}_D[\mathbb{1}_{h\neq f}(x)] = \frac{1}{|D|} \sum_{x\in D} \mathbb{1}_{h\neq f}(x) \ .$$

7.2 Hoeffding's inequality

For $\epsilon > 0$, the probability that the empirical error is off (the actual error) by more than ϵ is bounded from above by a decreasing function of $|D|$:

$$\mathbb{P}[|E(h) - E_D(h)| > \epsilon] \leq 2e^{-2\epsilon^2|D|}$$

Remarks:

- The sample points must be generated independently.

- The underlying distribution (process) must be fixed.

- The key factor in the bound is $|D|$.

7.3 Learning a hypothesis

When learning, you are no longer *given* a hypothesis but rather *choose* one from $\mathcal{H}$. For example, you may pick a hypothesis $h^\star$ that has the lowest empirical error on the training data:

$$h^\star = \operatorname*{argmin}_{h \in \mathcal{H}} E_D(h)$$

Theorem 2. *For $\epsilon > 0$, the probability that the empirical error of the chosen hypothesis is off (the actual error) by more than ϵ is bounded from above by a decreasing function of $|D|$ and an increasing function of $|\mathcal{H}|$:*

$$\mathbb{P}[|E(h^\star) - E_D(h^\star)| > \epsilon] \leq 2|\mathcal{H}|e^{-2\epsilon^2|D|}$$

Proof. The event $|E(h^\star) - E_D(h^\star)| > \epsilon$ implies:

$$[|E(h_1)-E_D(h_1)| > \epsilon] \quad \vee \quad [|E(h_2)-E_D(h_2)| > \epsilon] \quad \vee \quad \ldots \quad \vee \quad [|E(h_{|\mathcal{H}|})-E_D(h_{|\mathcal{H}|})| > \epsilon] \ ,$$

that is

$$[|E(h^\star) - E_D(h^\star)| > \epsilon] \quad \subseteq \quad \bigcup_{h \in \mathcal{H}} [|E(h) - E_D(h))| > \epsilon] \ .$$

Thus,

$$\mathbb{P}[|E(h^\star) - E_D(h^\star)| > \epsilon] \quad \leq \quad \mathbb{P}[\bigcup_{h \in \mathcal{H}} [|E(h) - E_D(h))| > \epsilon]] \ .$$

By Boole's inequality:

$$\leq \quad \sum_{h \in \mathcal{H}} \mathbb{P}[|E(h) - E_D(h))| > \epsilon] \ ,$$

where by applying Hoeffding's inequality to each summand, we get

$$\leq \quad 2|\mathcal{H}|e^{-2\epsilon^2|D|} \ .$$

$\square$

14