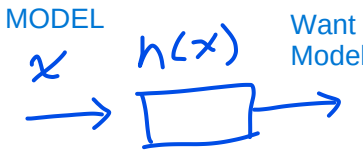


Want to make sure model is good at its predictions.
Modelling come with assumptions on the domains.

Supervised learning. Want to look at number of examples.

Training data being cartesian product of X*Y

Reducing to a search where goal is to find h

1 Function approximation

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

$y_i = f(x_i)$

$y_i \sim F_{y|x}(x_i)$

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

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

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 $\mathcal{H} \subseteq \mathcal{F}$

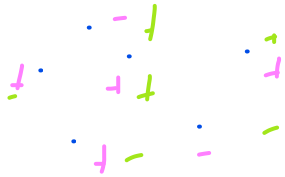
$2^{|X|}$

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

think of functions as its input and output.

2.1 Partial order of predicates

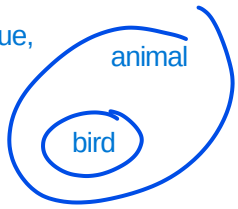
Predicate test if something holds or not

A relation R on a set A is a partial order if it is reflexive, transitive, and anti-symmetric.

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

if first one is true, then second one also true, but not necessarily the other way around.

$$h_a \underset{g}{\leq} h_b \quad \equiv \quad \forall x \in \mathcal{X} \;\; h_a(x) \implies h_b(x)$$



The relation *less general or equal* has the following properties:

Reflexive, aRa aA.

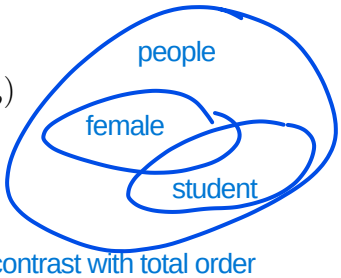
- reflexive: $\forall h \in \mathcal{H}: \quad h \underset{g}{\leq} h$

Antisymmetric aRb and bRa a = b.

- 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 aRb and bRc aRc.

- 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 $\mathcal{H}$).

contrast with total order

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

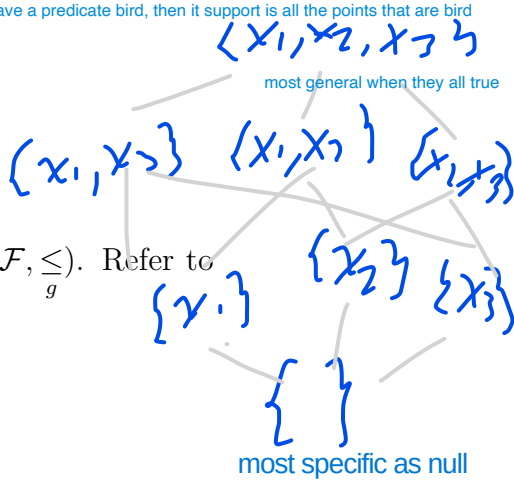
so if have a predicate bird, then it support is all the points that are bird

$$supp(h) = \{x \in \mathcal{X} : h(x)\} \;\; .$$

As a result we have

Support of ha is a subset of hb

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



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)`.

Answer:

```
def consistent ( h, D )
   return all( h(x) == y for x, y in D )
```

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

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.

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

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

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

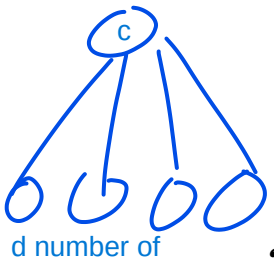
$R = \mathbb{R}^{d+1}$

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

Decode take the weight and bias and return true or false

Decode is part of the design

- $\mathcal{H}$ is the set of perceptrons over $\mathbb{R}^d$ Perceptron represented by weights and bias.
- $\mathcal{H}$ is the set of naive Bayes networks where the objects are described by d Boolean input variables and the class label is binary.



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

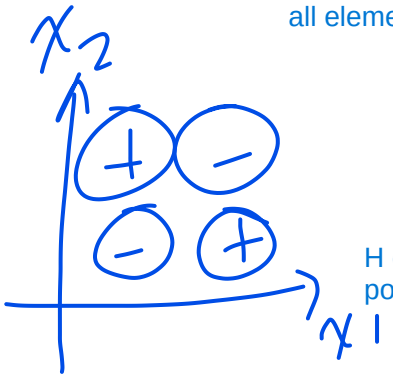
where function is deterministic

$$VS_{\mathcal{H},D} = \{h \in \mathcal{H} : consistent(h, D)\}$$

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



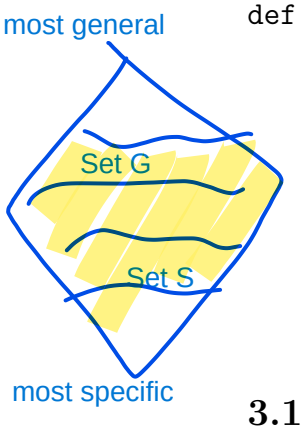
H can be though of all the lines that you can draw through this graph space except no line is possible

consider definition of version space depend on D, H, what set of hypothesis and data you have. need D because talkijing about consistency, need H as talking about functions

partial order with less general and equal to hierarchy graph where there is none less than or equal to general in the same hierarchy row.

can have two bounds
as general as you can be and still be consistent with your data. Highlighted is the version space that is consistent with your data that is the most specific is S, and G is most general that can still be specific

Example The entire version space can be used for prediction.




```
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)
```

may or may not be correct as x could be out of D. if it in D, and do find a hypothesis, then they are going to agree with each other,.

if not in D, they will disagree,

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.

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

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

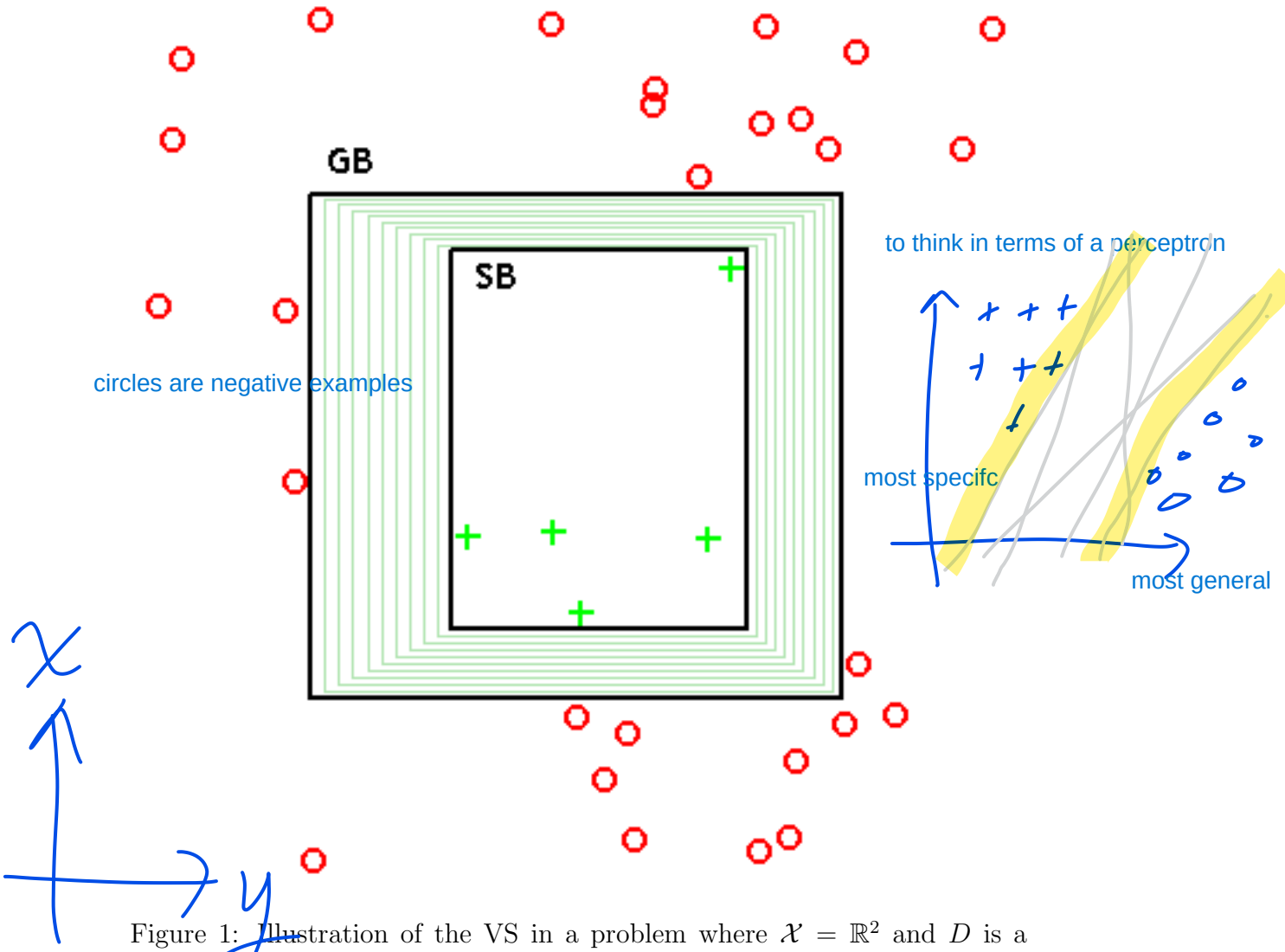


Figure 1: Illustration of the VS in a problem where $\mathcal{X} = \mathbb{R}^2$ and D is a collection of positive points (depicted by '+') and negative points (depicted by 'o'). $\mathcal{H}$ is the set of all rectangles in $\mathbb{R}^2$ where a point inside a rectangle will be predicted as positive.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

the idea is to learn a VS not by enurmerating over al the elements in H as this is expensive, but instead to just look st S and G as they implicityly define the version space.

slowly moving S and G tighter to be more specifc while still being most general as you iterate to see all the data

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
       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
       Remove from G any h that is more specific than another hypothesis in G
```

*** more general or equal

** more specific or equal

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

1-d features in total. position 1 is everything in domain of 1 union with wildcard and null

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

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

- 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

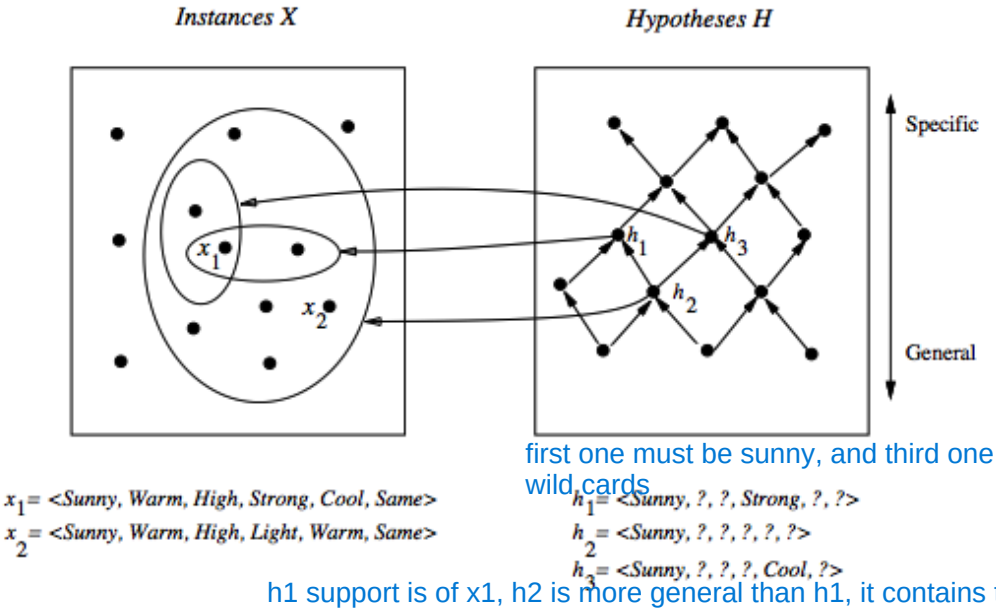
1. What is the cardinality of $\mathcal{X}$? 3 * 2 * 2 * 2 * 2 * 2 = 3 * 2^5 = 96
2. What is the cardinality of $\mathcal{F}$? 2^96
3. If we use the conjunction of constraints as our hypothesis class, what is the size of R? What is the size of $\mathcal{H}$?

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

   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|



first one must be sunny, and third one strong, rest can be wild cards

h1 support is of x1, h2 is more general than h1, it contains the support of h1, h1 has strong in 4th position, h2 can take wild card instead.

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

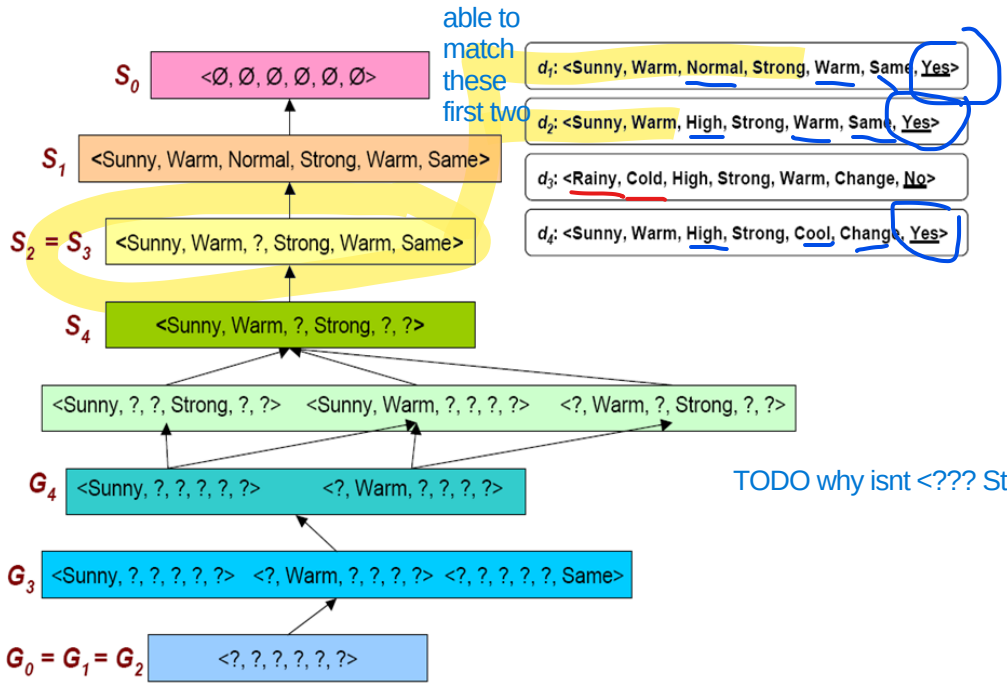


Figure 3: The trace of CEA using conjunction of constraints and four training data points.

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?

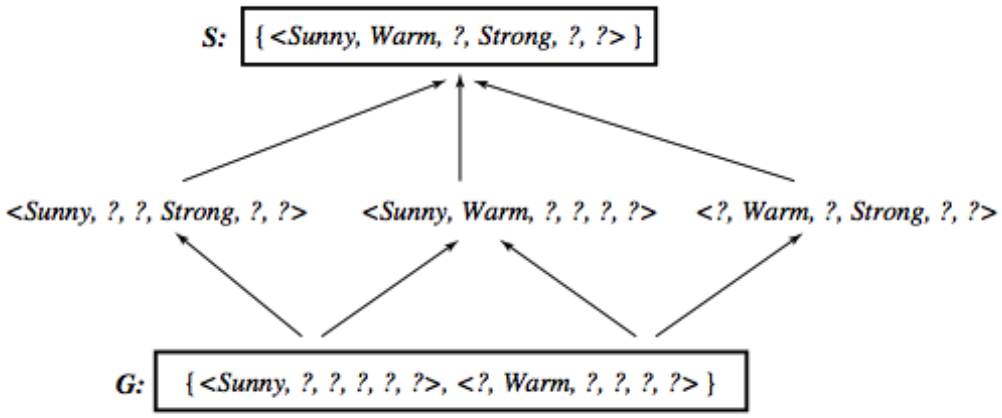


Figure 4: The resulting VS which can be specified using only S and G. If necessary, S and G can be used to generate the rest of the hypotheses in the VS.

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:

lots of freedom to lerning H = F then you just ;not learning more than what we observser where in this course you wouldlike to generize, where this is not possible without 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.

both 2^X

allowaning all logical formulas, able to express any fucntion so any F is in H, H is size of F

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:

true if exactly x1, exacly x2, exactly x3

if tts not x4 or not x5, itll return true

- $S = \{(x_1 \vee x_2 \vee x_3)\}$
- $G = \{\neg(x_4 \vee x_5)\}$

- 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.

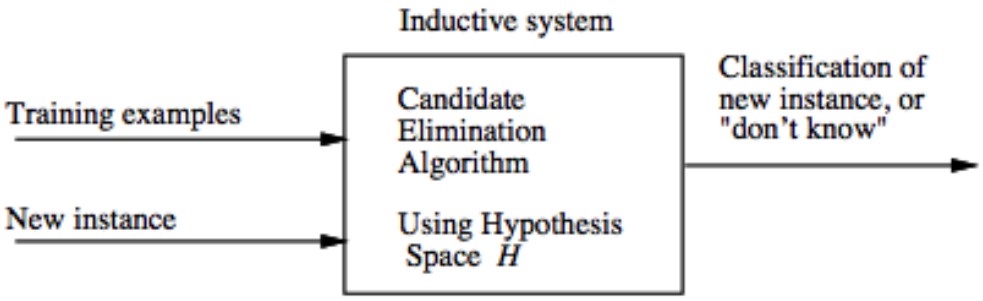
5.2 No inductive inference without a bias

A learner that makes no a priori assumptions regarding the identity of the target concept, has no rational basis for classifying unseen instances.

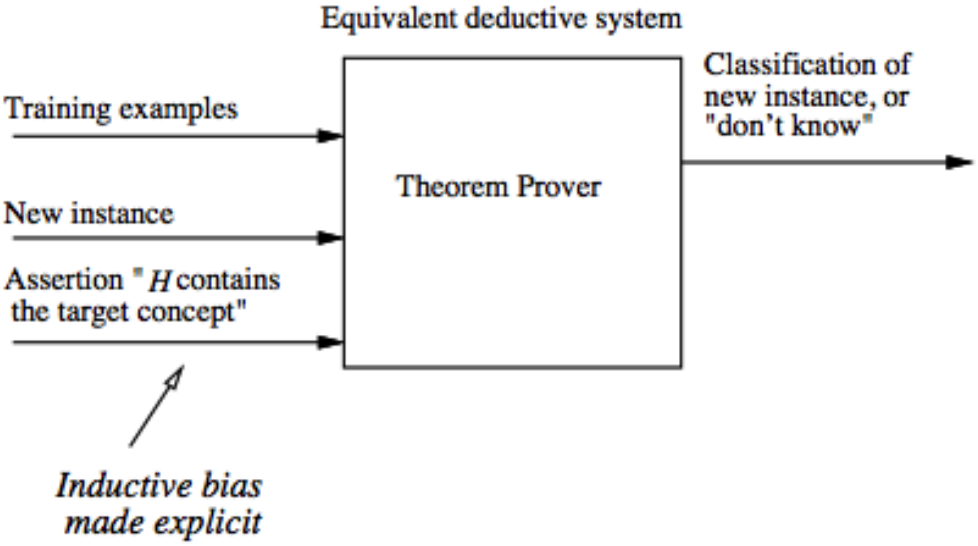
6 Inductive learning as deductive process

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.

if you manage to describe you rbias as a set of assumtpion then can in fact use a deductive process (eg. prolog) to actually do learning to describe functions too - not something we need to do in practise.



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.



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:

different places in the space has different chances of appearing error in this dense area

area is

not all points are equally likely

more likely distrubted in dense, and less likely in corners

indicator function return 0 or 1

name of functin is 1

subscription describe this is about h!=f

return 1 is there is an error for a given input (x)

f is true
h is predictions

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

We define the following quantity as the error of h:

error of h is prob that h does not agree with f = expected vauleof the indicator

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

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 $\mathbb{1}_{h \neq f}(x)$.

imagine f(x) in dense areas, if theres a h that make error in corners but is good in center, then that good h

on other hand, h with lots of error in dense area but correct in corner then not a good h

in way weighing error in high Liki hood error, expected value,

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 $|D|$:

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

upper bound in this form^^ expodential is a negative value, and theres two cases, where it is size of D. as D is larger (so more samples) then bound will approach zero and be smaller

Remarks:

- The sample points must be generated independently.
- The underlying distribution (process) must be fixed.
- The key factor in the bound is $|D|$.

make it general, but general H can work against you, in putting bound in your direction

how to determine size of H when working in the numeric space

use trick to look at it computationally, bit, floating point number as unique number, size of r, and get a value for H
pure mathematical sense, VC notion. would make H very large.

16 bit representation of numbers

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:

could have H that only agree with some data so it not really consistent

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

arg is returning something from the h. using value of the error to pick a hypo that generate the less amount of error. this is the learning algorithm

error of 0

|H| linear |D|

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 $|\mathcal{H}|$:*

actual error and empirical error, difference in upper bound, however now theres the |H| as a coefficient

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

probalistic statement that we dont have too large H or it becomes hard to learn

special case of hoeffding equality

as you have more hypo, you have more |H| so upper bound goes up. way to bring it down is to have more data - so want good balance between having lot data to have more confidence in upper bound - bringing it closer

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

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

if we have lots of H, then want lots of data to help

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

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

□