Concept Learning and Version Space

To understand how a machine can learn from past experiences and predict future based on the experience gained, we need to understand the working of a human brain first. Once we find out the pattern of a human brain to solve a problem, we can make our machine to learn in an almost same way.

What is Learning?

"The activity or process of gaining knowledge or skill by studying, practicing, being taught, or experiencing something". There are various categories of learning methods.

Humans learn through 5 different methods.

1. Rote Learning (memorization): Memorizing things without knowing the concept/ logic behind them
2. Passive Learning (instructions): Learning from a teacher/expert.
3. Analogy (experience): Learning new things from our past experience.
4. Inductive Learning (experience): On the basis of past experience *formulating a generalized concept*.
5. Deductive Learning: *Deriving new facts* from past facts.

In ML, the interest is in inductive learning and it's based on formulating a generalized concept (rule) after observing a number of instances of examples of the concept ie., we can make our machine to learn from past data and make them intelligent to identify whether an object falls into a specific category of our interest or not.

What is a concept?

Assume that we have collected data for some attributes/features of the day like, Sky, Air Temperature, Humidity, Wind, Water, Forecast. Let these set of instances be denoted by X and many concepts can defined over the X. For example, the concepts can be

- Days on which my friend Sachin enjoys his favorite water sport
- Days on which my friend Sachin will not go outside of his house.
- Days on which my friend Sachin will have night drink.

Therefore, a concept is a boolean-valued function defined over a large set of objects or events. So, concept-learning is defined as inferring a boolean-valued function from training examples of input and output of the function.

Target concept — The concept or function to be learned is called the target concept and denoted by c. It can be seen as a boolean valued function defined over X and can be represented as c: X → {0, 1}. Here, X denotes the set of items/objects over which the concept is defined called the set of instances.

For the target concept c , "Days on which my friend Sachin enjoys his favorite water sport", an attribute EnjoySport is included in the below dataset X and it indicates whether or not my friend Sachin enjoys his favorite water sport on that day.

In concept learning, we aim to use data to teach a machine to solve a binary classification problem. That is, to classify a data-point as either belonging to or not belonging to a particular concept or idea.

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

Now the target concept is EnjoySport : X $\rightarrow$ {0,1}. It is the mapping function f:X->y and this represents the exact knowledge defining the relationship between input variables, X and output variable, y. This target concept is unknown. With this, a learner task is to learn to predict the value of EnjoySport for arbitrary day, based on the values of its attribute values. When a new sample with the values for attributes <Sky, Air Temperature, Humidity, Wind, Water, Forecast> is given, the value for EnjoySport (ie. 0 or 1) is predicted based on the previous learning.

Hypothesis:

It is a proposition or statement made on the basis of limited evidence as a starting point for further investigation. Ie., it is a function that best describes the target concept.

A simple representation for a hypothesis:

Each hypothesis is represented as a conjunction of constraints on the instance attributes. ie., each hypothesis is a vector of six constraints, specifying the values of the six attributes <Sky, Air Temperature, Humidity, Wind, Water, Forecast>. In hypothesis representation, value of each attribute could be either

- "?' — that any value is acceptable for this attribute,
- specify a single required value (e.g., Warm) for the attribute, or

- "0" that no value is acceptable.

For example :

- the hypothesis that my friend Sachin enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression < ?,cold, High,?, ? ,?>
- The most general hypothesis — < ?, ? , ? , ?, ? , ?> that every day is a positive example

The most specific hypothesis — < 0, 0, 0, 0, 0, 0>that no day is a positive example.

(x, c(x)) — When learning the target concept, the learner is presented by a set of training examples, each consisting of an instance x from X, along with its target concept value c(x). Instances for which c(x) = 1 are called positive examples and instances for which c(x) = 0 are called negative examples. We will often write the ordered pair (x, c(x)) to describe the training example consisting of the instance x and its target concept value c(x).

H — denotes the set of all possible hypotheses that the learner may consider regarding the identity of the target concept. So, H = {h1, h2, …. }.

The goal of the inductive learning algorithm is to induce a "general rule" from a set of observed instances i.e., to find a hypothesis 'h' in H such that h(x)=c(x), for all x in X. But for a given set of examples, in reality, the learning algorithm returns a function h(x) that only approximates c. Given a set of training examples of the target concept c, the task of the learner is to hypothesize, or estimate, c.

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

Hypothesis Space

Let X denote the instances and H as hypotheses in the EnjoySport learning task.

Each of the 6 attributes <Sky, Air Temperature, Humidity, Wind, Water, Forecast> can have the following distinct values. Sky can have 3 possible values (sunny, cloudy and rainy); Air

temperature can have 2 possible values (warm and cloud) and so on. Hence, the total number of distinct instances is given as

Distinct instances =3*2*2*2*2*2=96.

In hypothesis, the value of each attribute could be either "?" or "0" in addition to the other defined values. So, the total number of syntactically distinct hypothesis (hypothesis space) H is given as

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

Attributes	Values	Count
1. Sky	Sunny, Cloudy, Rainy	3
2. AirTemp	Warm, Cold	2
3. Humidity	Normal, High	2
4. Wind	Strong, Weak	2
5. Water	Warm, Cool	2
6. Forecast	Same, Change	2

$$Distinct\ Observations\ in\ X = 3.2.2.2.2.2 = 96$$

$$Distinct\ hypothesis\ in\ H = 5.4.4.4.4.4 = 5120$$

The number of combinations: 5×4×4×4×4×4 = 5120 syntactically distinct hypotheses. They are syntactically distinct but not semantically. For example, the below 2 hypothesis says the same but they look different.

h1 = <Sky=0 AND Temp=warm AND Humidity=? AND Wind=strong AND Water=warm AND Forecast=same >

h2 = <Sky=sunny AND Temp=warm AND Humidity=? AND Wind=strong AND Water=0 AND Forecast=same>

Neither of these hypotheses accept any "day", so semantically the same. All such hypothesis having same semantic is counted as 1. So, the total number of semantically distinct hypothesis= 1 (hypothesis with one or more 0) + 4×3×3×3×3×3 (add ? to each attribute) = 973 semantically distinct hypotheses

So, concept learning can be viewed as the task of searching through a large space of hypotheses, to find the hypothesis that best fits the training examples.

A hypothesis "h" is consistent with a set of training examples D of target concept c if and only if h(x) = c(x) for each training example in D.

Let h1=(sunny,?,?,strong,?,?) and h2=(sunny,?,?,?,?,?). By looking at the 2 hypothesis representations, h2 has fewer constraints on attributes than h1. The sets of instances that are classified positive by hl will be less than the sets of instances that are classified positive by h2. In fact, any instance classified positive by hl will also be classified positive by h2. – Therefore, we say that h2 is more general than hl or h1 is more specific than h2.

If h1(x)=1 it implies that h2(x)=1 => h2 $\geq_g$ h1 which means h2 is more-general-than-or-equal-to h1.

Let x1=<sunny, warm,high,strong,cool,same>,

And x2=<sunny, warm, high,light,warm, same>

Let h1=<sunny,?,?,strong,?,?>

h2=<sunny,?,?,?,?,?>

h3=<sunny,?,?,?,cool,?>

h1 classifies x1, h2 classifies x1 and x2 and h3 classifies x1. This indicates h2 is more-general-than h1 and h3. By taking advantage of the naturally occuring structure over the hypothesis space, we can design learning algorithm that exhaustively searches even infinite hypothesis spaces without explicitly enumerating every hypothesis.

Example 2:

Consider an example if a person goes to a movie or not based on 4 binary features with 2 values (true or false) possible:

1. Has Money -> <true, false>
2. Has Free Time -> <true, false>
3. It's a Holiday -> <true, false>
4. Has Pending work -> <true, false>

Let the training data have two positive samples and one negative:

Has money	Has free time	It's a holiday	Has pending work	Goes to movie or not
True	True	False	False	Positive
True	False	False	True	Positive
True	False	False	True	negative

Hypothesis Notations:

Total number of distinct instances = 2*2*2*2=16

Total number of syntactically distinct hypothesis are: 4*4*4*4=256.

Total number of semantically distinct hypothesis are: (3 * 3 * 3 * 3) + 1=82 where 3 because one feature can have either true, false or '?' and one hypothesis for rejects all (∅).

Definition of concept learning:
It is the *"Problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples"-Tom Michell.*

Find-s:
Finding a maximally specific hypothesis
The find-S algorithm finds the most specific hypothesis that fits all the positive examples. Note: The algorithm considers only positive training examples. The find-S algorithm starts with the most specific hypothesis and generalizes this hypothesis each time it fails to classify an observed positive training data. Hence, the Find-S algorithm moves from the most specific hypothesis to the most general hypothesis.
Steps:

i) Initialize h to the most specific hypothesis in H
ii) 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
iii) Output hypothesis, h.

Find-S algorithm for Example 1:

i) h=<0,0,0,0,0,0>

ii) First training example x1 = < Sunny, Warm, Normal, Strong ,Warm ,Same>, EnjoySport = +ve. Observing the first training example, it is clear that hypothesis h is too specific. None of the "Ø" constraints in h are satisfied by this example, so each

is replaced by the next more general constraint that fits the example h1 = < Sunny, Warm, Normal, Strong ,Warm, Same>.

iii) Consider the second training example x2 = < Sunny, Warm, High, Strong, Warm, Same>, EnjoySport = +ve. The second training example forces the algorithm to further generalize h, this time substituting a "?" in place of any attribute value in h that is not satisfied by the new example. Now h2 =< Sunny, Warm, ?, Strong, Warm, Same>

iv) Consider the third training example x3 =< Rainy, Cold, High, Strong, Warm, Change>,EnjoySport = — ve. The FIND-S algorithm simply ignores every negative example. So the hypothesis remain as before, so h3 =< Sunny, Warm, ?, Strong, Warm, Same>

v) Consider the fourth training example x4 =<Sunny,Warm,High,Strong, Cool,Change>, EnjoySport =+ve. The fourth example leads to a further generalization of h as h4 =< Sunny, Warm, ?, Strong, ?, ?>

vi) So the final hypothesis is < Sunny, Warm, ?, Strong, ?, ?>

The search begins with h_0, the most specific hypothesis in H, then considers increasingly general hypothesis (h1 through h4) as mandated by the training examples. At each step, the hypothesis is generalized only as far as necessary to cover the new positive example. ie., at each stage, the hypothesis is the most specific hypothesis consistent with the training examples observed upto that point.

The key property of the FIND-S algorithm —

- FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples
- FIND-S algorithm's 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.

Limitations:

1. Find-S finds only one most specific hypothesis. There is no way to determine if there are more hypothesis that is consistent with the data.
2. It ignores the negative examples
3. A good concept learning algorithm should be able to backtrack the choice of hypothesis found so that the resulting hypothesis can be improved over time. But in Find-S, there is no option for that.

Find-S for example-2:

i) H=<0,0,0,0>
ii) After observing the first positive training example, replace the existing hypothesis with the new one.
h=<T,T,F,F>
iii) The second positive example is <T,F,F,T> and current h=<T,T,F,F>
So, perform pairwise conjunctive AND and put '?' where the result of AND is False. So, h=<T,?,F,?>
iv) Ignore the last negative example. Therefore the final hypothesis =<T,?,F,?>

Version space

It represents the set of all hypotheses in H, consistent with the observed training examples, D.

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

List-Then-Eliminate Algorithm

It initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.

The version space of candidate hypotheses thus shrinks as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples. – Presumably, this is the desired target concept.

Steps:

1. Version space ← a list containing every hypothesis in H
2. For each training example, <x,c(x)>, remove from VS any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypothesis in VS.

It has many advantages, including the fact that it is guaranteed to output all hypotheses consistent with the training data.

Disadvantages:

– If insufficient data is available to narrow the version space to a single hypothesis, then the algorithm can output the entire set of hypotheses consistent with the observed data.

- Unfortunately, it requires exhaustively enumerating all hypotheses in H - an unrealistic requirement for all but the most trivial hypothesis spaces. It can be applied only whenever the hypothesis space H is finite.

Candidate -elimination algorithm:

A version space can be represented with its general and specific boundary sets.

The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S. Given only these two sets S and G, it is possible to enumerate all members of a version space by generating hypotheses that lie between these two sets in general-to-specific partial ordering over hypotheses. Every member of the version space lies between these boundaries.

Steps:

i) Initialize G to the set of maximally general hypotheses in H
ii) Initialize S to the set of maximally specific hypotheses in H
iii) For each training example d, do
    If d is a positive example
      Generalize the specific hypothesis
    Else
      make the general hypothesis more specific

Example-1:

Consider the training data

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

Initially, S_0=<0,0,0,0,0,0>, G_0=<??????>

i) First training example — its a positive example and when it is presented to the CANDIDATE-ELIMINTION algorithm, it checks the S boundary and finds that it is overly specific and it fails to cover the positive example. The boundary is therefore revised by moving it to the least more general hypothesis that covers this new example.

S1=<sunny, warm, normal, strong, warm, same> and G1=<??????>

ii) Second is also a positive example. So, generalize S so as to make it consistent with this example too.

S2=<sunny, warm, ?, strong, warm, same> and G2=<??????>

iii) Third is a negative example. So, for every value in negative example that is inconsistent with S, form an individual hypothesis by listing the values of S.

S3=<sunny, warm, ?, strong, warm, same>
G3={ <Sunny?????>, <?warm????>,<?????same>}

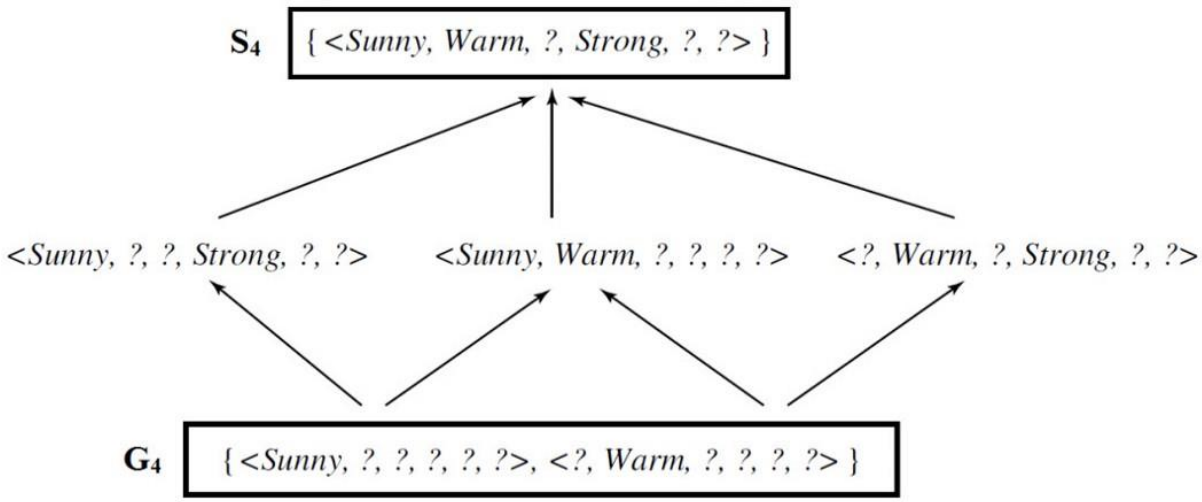
iv) Fourth is a positive example. So,

S4=<sunny, warm,?strong??> and remove from G those hypothesis which are inconsistent with S.

<?????same> is removed because in S itself, the last attribute is ?. But in G, it has a specific value 'same'.

G4={ <Sunny?????>, <?warm????>}

- After processing these four examples, the boundary sets S4 and G4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples. The entire version space, including those hypotheses bounded by S4 and G4.



- This learned version space is independent of the sequence in which the training examples are presented (because in the end it contains all hypotheses consistent with the set of examples).

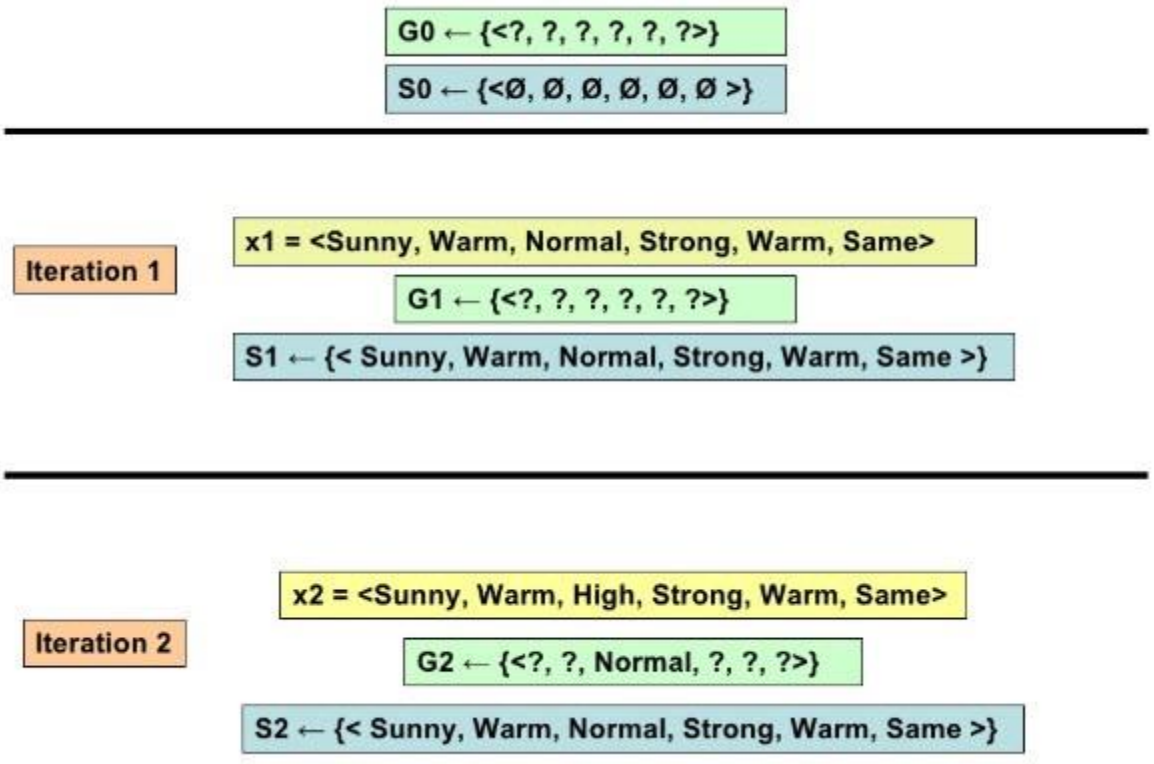
- As further training data is encountered, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.

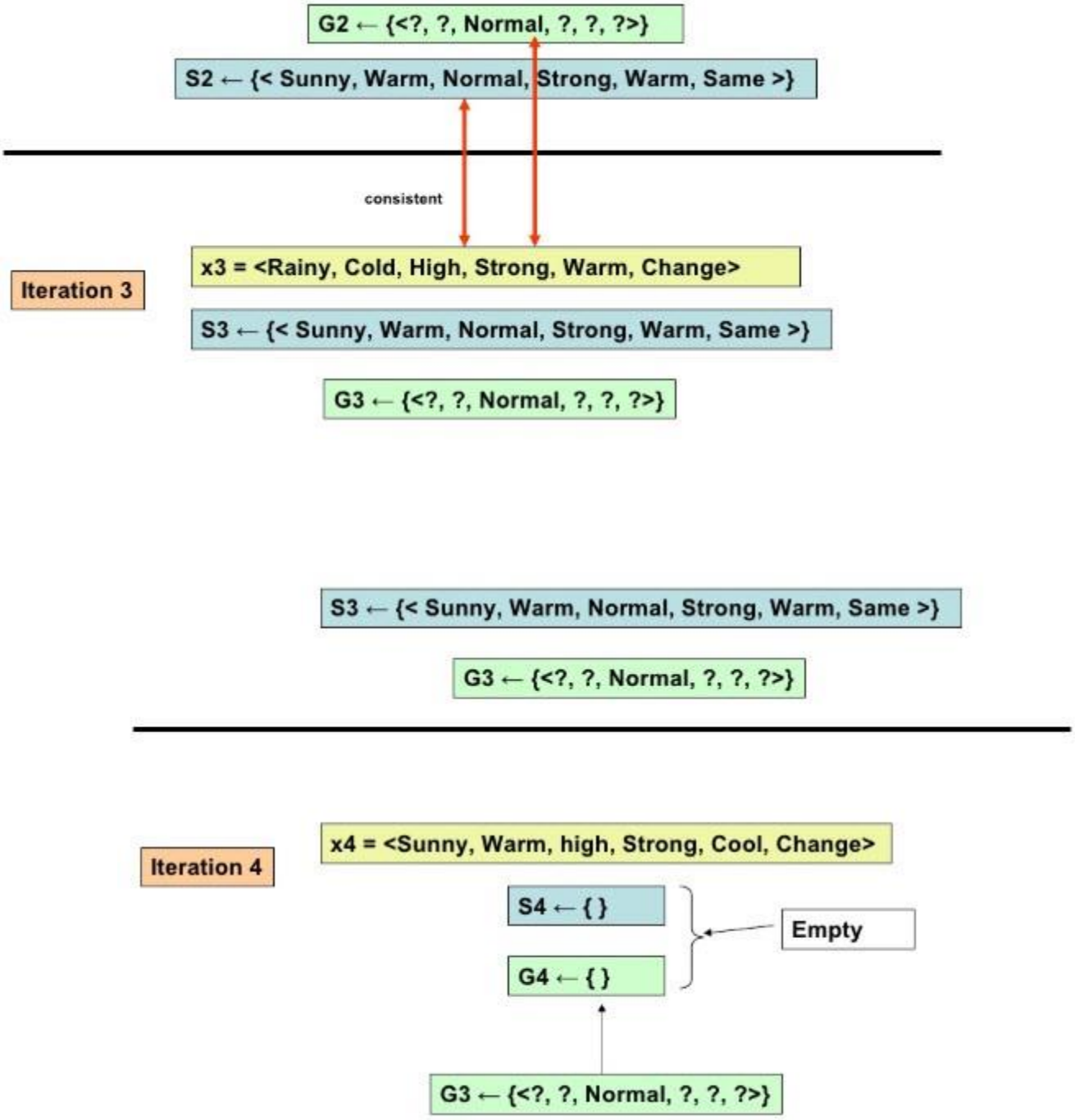
What will happen if the training data contains errors ?.

Suppose, for example, that the second training example above is incorrectly presented as a negative example instead of a positive example.

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

Let's run the candidate elimination algorithm on this data and see the result.





Example-2:

Consider a trading agent trying to infer which articles a user reads, based on keywords for the article. Suppose the learning agent has the following data:

article	*Crime*	*Academic*	*Local*	*Music*	*Reads*
a_1	*true*	*false*	*false*	*true*	*true*
a_2	*true*	*false*	*false*	*false*	*true*
a_3	*false*	*true*	*false*	*false*	*false*
a_4	*false*	*false*	*true*	*false*	*false*
a_5	*true*	*true*	*false*	*false*	*true*

The aim is to learn which articles the user reads.

In this example, Reads *is the target feature*

Initially, S=<0,0,0,0> G=<?,?,?,?>

i) It's a positive example. So, generalize S as S=<T,F,F,T>
G=<?,?,?,?>

ii) It's a positive example. So, again generalize S as S=<T,F,F,?> and G=<?,?,?,?>

iii) It's a negative example. So, specialize G and it should be consistent with S. Compare each value of -ve result with specific hypothesis. If they differ, make an individual pair for each value.
S=<T,F,F,T> and G={<T,?,?,?>,<?,F,?,?>}

iv) It's a negative example. So, eliminate those from G which are inconsistent with the example.
S=<T,F,F,T> and G={<T,?,?,?>}

v) It's a positive example. So, generalize S as S=<T,?,F,?>, G={<T,?,?,?>}

So, VS: S=<T,?,F,?>, G={<T,?,?,?>}

Example-3:

Target concept: whether the book will be bought

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

Initially, S=<00000>
G=<?????>

i) First itself is a negative example.
So, specialize G by creating an individual pair for each value of the attributes which are not in the negative example.
Eg., 'citations' has 2 different values. So, exclude 'some' as it is in the negative example; 'size' has 3 different values -small, big and medium. Out of these, exclude small and form individual pairs with the remaining values. Likewise, do for the other attributes too.
G={<many????>,<?big???>, <?medium???>,<???expensive?> , <????many>, <????few>

ii) Second is a positive example. So, S=<many,big, no,expensive, many> and remove from G, those are inconsistent with S.
G={<many????>,<?big???>,<???expensive?> , <????many>}
<?medium???> and <????few> are removed from G as they are inconsistent.

iii) Third is also a positive example. So, S=<many,?,no, expensive, ?>
G={<many????>,<???expensive?> }

iv) Fourth is also a positive example. So, S=<many,?,no, ?, ?>, G=<many????>

Example-4:

Learning the concept of "Japanese Economy Car"

Features: Country of Origin, Manufacturer, Color, Decade, Type

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

Initially, G=<?,?,?,?,?> S=<0,0,0,0,0>

i) First example (Japan, Honda, Blue, 1980, Economy) is a positive example.
So, S=< Japan, Honda, Blue, 1980, Economy>, G=<?,?,?,?,?>

ii) Second is a negative example. So, specialize G and ensure that it is consistent with S. Compare each value of -ve result with specific hypothesis. If they differ, make an individual pair for each value.
S=< Japan, Honda, Blue, 1980, Economy>, G={<?,Honda,?,?,?> , <?,?,Blue,?,?>, <???1980 ?>, <????Economy>}

iii) Third is a positive example. S=<Japan, ?, Blue, ?, Economy> and in G, remove those that are inconsistent with S.
a)This hypothesis <?,Honda,?,?,?> should be removed because in S itself, the second parameter is '?' but in G, it requires that the second parameter should be 'Honda'.
b) <?,?,Blue,?,?> is consistent with S. So, include it.
c) <???1980 ?> should be removed because in S itself, the fourth parameter is '?' but in G, it requires that the fourth parameter should be '1980'.
d) <????Economy> is consistent with S. So, include it.
So, S=<Japan, ?, Blue, ?, Economy> and G= {<?,?,Blue,?,?>, <????Economy>}

iv) Fourth is a negative example. Specialize G to exclude the negative example but stay consistent with S.
So, S=<Japan, ?, Blue, ?, Economy> and G= {<?,?,Blue,?,?>, **<Japan???Economy>**} [Note: we are not removing <????Economy> because there is a specialization for first parameter 'Japan' in S… We have to remove a hypothesis only if S has ? and G has some specific value]

v) Fifth is a positive example. Prune G to exclude hypothesis inconsistent with positive example.
S=<Japan,???Economy> and G=<Japan ???Economy>

vi) Sixth is again a positive example. S=Japan???Economy> and G=<Japan???Economy>

vii) Seventh is a negative example. Example is inconsistent with the VS. G cannot be specialized and S cannot be generalized. The VS collapses.

Conclusion: No conjunctive hypothesis is consistent with dataset.