MACHINE LEARNING

(Course Code: B20AM2103)

Lecture Notes

Concept Learning

K V S MURTHY

Department of Computer Science and Engineering

Sagi RamaKrishnam Raju Engineeirng College

Bhimavaram, Andhrapradesh-534202

What is concept learning in machine learning?

In Machine Learning, concept learning can be applied in training computer programs.

Concept Learning: **Inferring a Boolean-valued function from training examples of its input and output**.

"A task of acquiring potential hypothesis (solution) that best fits the given training examples is Concept Learning Task."

We will discuss the two most popular approaches to find a suitable hypothesis, they are:

1. Find-S Algorithm
2. List-Then-Eliminate Algorithm

Find-S Algorithm:

The find-S algorithm is a basic concept learning algorithm in machine learning.
The find-S algorithm finds the most specific hypothesis that fits all the positive examples.
Note here that the algorithm considers only those positive training example.

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.

Following are the steps for the Find-S algorithm:

1. Initialize h to the most specific hypothesis in H

2. For each positive training instance x

For each attribute constraint a, in h

If the constraint a, is identical in h and x   Then do nothing

Else replace a, in h by the more general constraint " ? "

3. Output hypothesis h

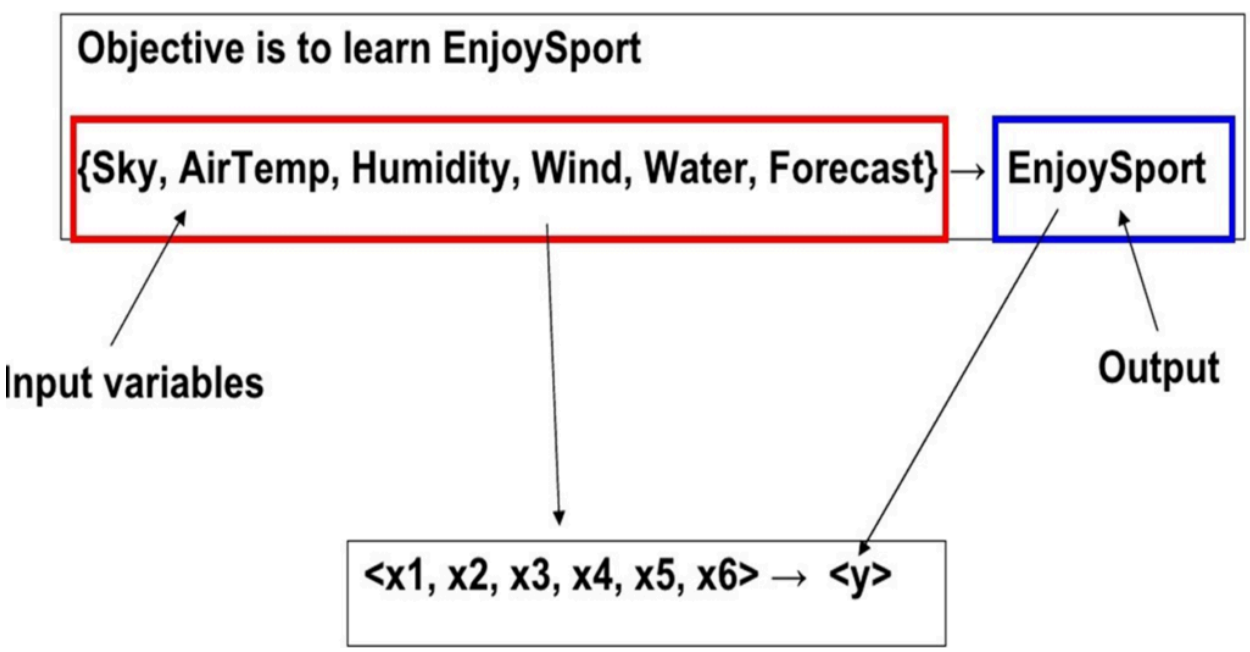
Consider the example task of learning the target concept

“Days on which my friend Akshay playing his favorite Sport.”

We have some attributes/features of the day like, ***Sky, Air Temperature, Humidity, Wind, Water, Forecast*** and based on this we have a target Concept named **EnjoySport**.

We have the following training example available:

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



Instances (X)

Target Concept (C)

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

Training examples (D)

Let's Design the problem formally with **TPE**(Task, Performance, Experience):

Problem: Leaning the day when Akshay enjoys the sport.

Task T: Learn to predict the value of EnjoySport for an arbitrary day, based on the values of the attributes of the day.

Performance measure P: Total percent of days (EnjoySport) correctly predicted.

Training experience E: A set of days with given labels (EnjoySport: Yes/No)

Here We get a hypothesis h_i for our training set as below:

hi(x) := <x1, x2, x3, x4, x5, x6>

where x1, x2, x3, x4, x5 and x6 are the values of **Sky**, **AirTemp**, **Humidity**, **Wind**, **Water** and **Forecast**.

Hence h1 will look like(the first row of the table above):

h1(x=1): <Sunny, Warm, Normal, Strong, Warm, Same >

Note: x=1 represents a positive hypothesis / Positive example

x=0 represents a Negative hypothesis / Negative example

Here

The most generic hypothesis will be < ?, ?, ?, ?, ?, ?> where every day is a positive example

The most specific hypothesis will be <0,0,0,0,0,0 > where no day is a positive example.

We want to find the most suitable hypothesis which can represent the concept. The data in **Training example 1** is {SUNNY, WARM, NORMAL, STRONG, WARM, SAME}. We see that our initial hypothesis is more specific and we have to generalize it for this example. Hence, the hypothesis becomes:

h = { SUNNY, WARM, NORMAL, STRONG, WARM, SAME }.

Consider Training example 2 :

Here we see that this example has a positive outcome. So, We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (" ? ").

The example is { SUNNY, WARM, HIGH, STRONG, WARM, SAME } we observe that only humidity attribute value is different, so we replace it with general case (" ? "). After doing the process the hypothesis becomes:

h = { SUNNY, WARM, ?, STRONG, WARM, SAME }.

Consider Training example 3:

Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.

h = { SUNNY, WARM, ?, STRONG, WARM, SAME }.

Consider Training example 4:

The data present in Row 4 is { SUNNY, WARM, HIGH, STRONG, COOL,CHANGE } We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (" ? ").

After doing the process the hypothesis becomes:

h = { SUNNY, WARM, ?, STRONG, ?, ? }.

this is our final hypothesis.

Example :

Consider the following data set having the data about which particular seeds are poisonous.

EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES

ATTRIBUTES ON WHICH THE CONCEPT DEPENDS ON **CONCEPT**

The data in **Training example** 1 is { GREEN, HARD, NO, WRINKLED }. We see that our initial hypothesis is more specific and we have to generalize it for this example. Hence, the hypothesis becomes :

h = { GREEN, HARD, NO, WRINKLED }

Consider Training example 2 :

Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.

h = { GREEN, HARD, NO, WRINKLED }

Consider Training example 3 :

Here we see that this example has a negative outcome. Hence we neglect this example and our hypothesis remains the same.

h = { GREEN, HARD, NO, WRINKLED }

Consider Training example 4 :

The data present in example 4 is { ORANGE, HARD, NO, WRINKLED }. We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (” ? ”). After doing the process the hypothesis becomes :

h = { ?, HARD, NO, WRINKLED }

Consider Training example 5 :

The data present in example 5 is { GREEN, SOFT, YES, SMOOTH }. We compare every single attribute with the initial data and if any mismatch is found we replace that particular attribute with a general case (” ? ”). After doing the process the hypothesis becomes :

h = { ?, ?, ?, ? }

Since we have reached a point where all the attributes in our hypothesis have the general condition, example 6 and example 7 would result in the same hypothesizes with all general attributes.

h = { ?, ?, ?, ? }

Hence, for the given data the final hypothesis would be :

Final Hyposthesis: h = { ?, ?, ?, ? }

Consistent Hypothesis, Version Space and List-Then-Eliminate Algorithm

An hypothesis *h* is said to be consistent hypothesis **with a set of training examples *D* iff *h*(*x*) = *c*(*x*) for each example in *D*,**

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

For Example:

Example	Citations	Book Size	In Library	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

Now consider the hypothesis:

h1 = (?, ?, No, ?, Many) – **Consistent Hypothesis** as it is consistent with all the training examples

Explanation: if the hypothesis is matched with training example we expect YES as target concept otherwise we expect NO.

Here our hypothesis h1 not matched with training example 1 so we expect NO and target in example1 is the same…

Similarly our hypothesis h1 is matched with training example 2 so we expect YES and target in example2 is the same…

Here for all training examples we have the desired output in the target concept and hence h1 is consistent

h2 = (?, ?, No, ?, ?) – **Inconsistent Hypothesis** as it is inconsistent with first training example

Explanation: if the hypothesis is matched with training example we expect YES as target concept otherwise we expect NO.

Here our hypothesis h2 is matched with training example 1 so we expect YES but the target concept is NO and so h2 is inconsistent.

Version Space

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

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

List-Then-Eliminate algorithm

1. V*ersionSpace* = a list containing every hypothesis in *H*

2. For each training example, <a(x), c(x)> Remove from *VersionSpace* any hypothesis *h* for which *h*(*x*) != *c*(*x*)

3. Output the list of hypotheses in *VersionSpace*.

Example:

Consider F1 and F2 are two features (attributes) with two possible values for each feature or attribute.

F1 – > A, B

F2 – > X, Y

Instance Space: (A, X), (A, Y), (B, X), (B, Y) – **4 Examples**

Hypothesis Space: (A, X), (A, Y), (A, **ø**), (A, ?), (B, X), (B, Y), (B, **ø**), (B, ?), (**ø**, X), (**ø**, Y), (**ø**, **ø**), (**ø**, ?), (**?**, X), (**?**, Y), (**?**, **ø**), (**?**, ?) – **16 Hypothesis**

Semantically Distinct Hypothesis: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (**?**, X), (**?**, Y), (**?**, ?), (**ø**, **ø**) – **10**

List-Then-Eliminate Algorithm Steps

Version Space: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y) (?, ?), (**ø**, **ø**),

Training Instances

F1	**F2**	**Target**
A	**X**	**YES**
A	**Y**	**YES**

Problem is to find all the consistent hypothesis.

Compare each hypothesis in version space with training examples and if it is not consistent we have to remove it.

Consider the hypothesis (A,X) from the version space and compare it with the two training examples.

When we compare (A,X) from the version space with first training example (A,X) it was matched and so expect YES in training example and the same is there.

When we compare (A,X) from the version space with Second training example (A,Y) it was not matched and so expect NO in training example and it is YES there.

So the hypothesis (A,X) is not consistent. And hence we remove it from the version space.

Sly we remove (A, Y), (B, X), (B, Y), (B, ?), (?, X), (?, Y), (**ø**, **ø**),

Consistent Hypothesis are (Version Space): (A, ?), (?, ?)

This is is the output from List-Then-Eliminate algorithm

Problems with List-Then-Eliminate Algorithm

The hypothesis space must be finite

Enumeration of the entire hypothesis or listing of the entire hypothesis is inefficient or taking more time.

Candidate Elimination algorithm

The candidate elimination algorithm incrementally builds the version space given a hypothesis space H and a set E of examples.

The examples are added one by one; each example possibly shrinks the version space by removing the hypotheses that are inconsistent with the example.

The candidate elimination algorithm does this by updating the general and specific boundary for each new example.

- **You can consider this as an extended form of Find-S algorithm.**
- **Consider both positive and negative examples.**
- **Actually, positive examples are used here as Find-S algorithm (Basically they are generalizing from the specification).**
- **While the negative example is specified from generalize form.**

Terms Used:

- **Concept learning: Concept learning is basically learning task of the machine (Learn by Train data)**
- **General Hypothesis: Not Specifying features to learn the machine.**
- **G = {'?', '?','?','?'…}: Number of attributes**
- **Specific Hypothesis: Specifying features to learn machine (Specific feature)**
- **S= {'pi','pi','pi'…}: Number of pi depends on number of attributes.**

- **Version Space: It is intermediate of general hypothesis and Specific hypothesis. It not only just written one hypothesis but a set of all possible hypothesis based on training data-set.**

Algorithm:

Step1: Load Data set

Step2: Initialize General Hypothesis and Specific Hypothesis.

Step3: For each training example

Step4: If example is positive example

```
if attribute_value == hypothesis_value:
   Do nothing
else:
   Replace attribute value with '?' (Basically generalizing it)
```

Step5: If example is Negative example

Make generalize hypothesis more specific.

Initialize the generic and specific boundary

S_0: $\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle$

G_0: $\langle ?, ?, ?, ?, ?, ? \rangle$

For each training example d, do:

If d is *positive* example

Remove from G any hypothesis h inconsistent with d

For each hypothesis s in S not consistent with d:

- Remove s from S
- Add to S all minimal generalizations of s consistent with d

If d is *negative* example

Remove from S any hypothesis h inconsistent with d

For each hypothesis g in G not consistent with d:

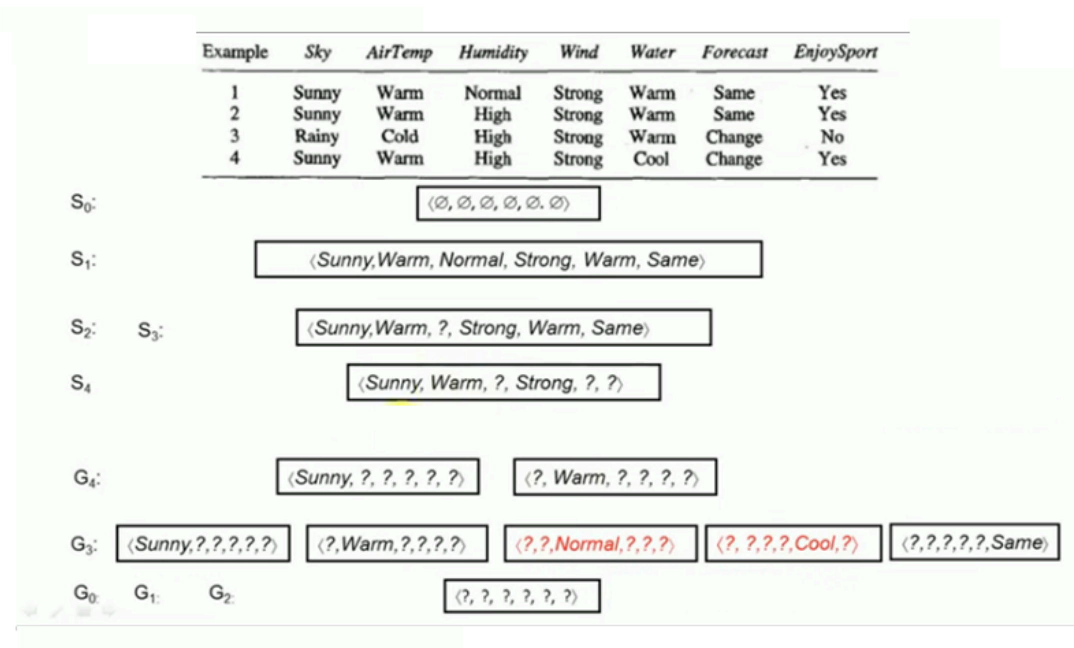
- Remove g from G
- Add to G all minimal specializations of g consistent with d

Example:

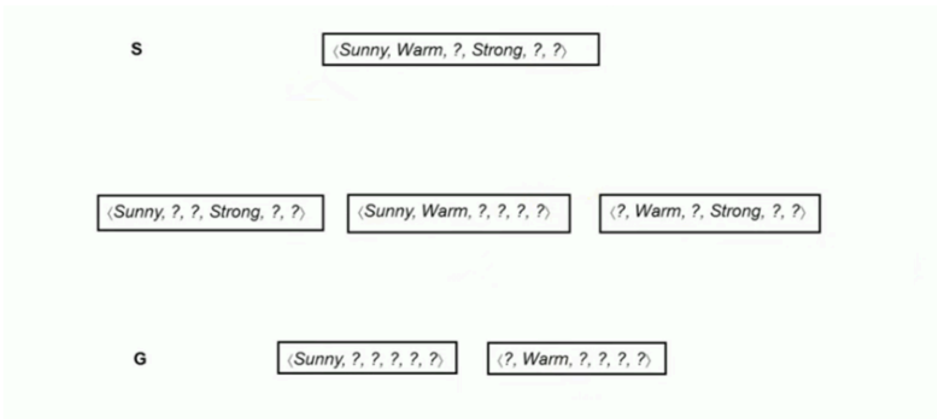
Consider the dataset given below:

Sky	Temperature	Humid	Wind	Water	Forest	Output
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

The version space will be constructed as follows.



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



This is the final version space.

Algorithmic steps: **Explanation**

Initially : G = [[?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?]]

S = [Null, Null, Null, Null, Null, Null]

For instance 1 : <'sunny','warm','normal','strong','warm ','same'> and positive output.

G1 = G

S1 = ['sunny','warm','normal','strong','warm ','same']

For instance 2 : <'sunny','warm','high','strong','warm ','same'> and positive output. Here the third attribute is different so we replace it with ? in S2

G2 = G

S2 = ['sunny','warm',?,'strong','warm ','same']

For instance 3 : <'rainy','cold','high','strong','warm ','change'> and negative output.

G3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, 'same']]

S3 = S2

For instance 4 : <'sunny','warm','high','strong','cool','change'> and positive output.

G4 = G3

S4 = ['sunny','warm',?,'strong', ?, ?]

At last, by synchronizing the G4 and S4 algorithm produce the output.

Output :

G = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?] [?, ?, ?, ?, ?, ?],
[?, ?, ?, ?, ?, 'same']]

S = ['sunny','warm',?,'strong', ?, ?]

Example 2 for Candidate Elimination algorithm.

Candidate Elimination Algorithm Solved Example

- S0: (0, 0, 0, 0, 0)
- S1: (0, 0, 0, 0, 0)
- S2: (Many, Big, No, Exp, Many)
- S3: (Many, ?, No, Exp, ?)
- S4: (Many, ?, No, ?, ?)

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

Final Hypothesis Set: (Many, ?, No, ?, ?) (Many, ?, ?, ?, ?)

- G4: (Many,?,?,?, ?)
- G3: (Many,?,?,?, ?) (?,?,?,Exp,?)
- G2: (Many,?,?,?, ?) (?, Big,?,?,?) (?,?,?,Exp,?) (?,?,?,?,Many)
- G1: (Many,?,?,?, ?) (?, Big,?,?,?) (?,Medium,?,?,?) (?,?,?,Exp,?) (?,?,?,?,Many) (?,?,?,?,Few)
- G0: (?, ?, ?, ?, ?)

Example 3 for Candidate Elimination algorithm.

Candidate Elimination Algorithm Solved Example

S0: (0, 0, 0)

S1: (0, 0, 0)

S2: (0, 0, 0)

S3: (Small, Red, Circle)

S4: (Small, Red, Circle)

S5: (Small, ?, Circle)

S: G: (Small, ?, Circle)

G5: (Small, ?, Circle)

G4: (Small, ?, Circle)

G3: (Small, ?, Circle)

G2: (Small, Blue, ?) (Small, ?, Circle) (?, Blue, ?) (Big, ?, Triangle) (?, Blue, Triangle)

G1: (Small, ?, ?) (?, Blue, ?) (?, ?, Triangle)

G0: (?, ?, ?)

Size	Color	Shape	Class / Label
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes