Module--1

Concept Learning

Contents

Introduction

Concept - An abstract idea

Much of learning involves acquiring **general concepts(idea)** from specific training examples.

Each concept can be viewed as describing some subset of objects or events defined over a larger dataset.

Ex: Animals is an object and **birds** is a subset of it.

Vehicle is an object and **car** is a subset of it.

Contd….

Each concept can be thought of as a **boolean valued function** defined over larger set.

Basically the result of the boolean function is ***Yes or No***

Ex: A function defined over all animals, whose value is true for birds and false for other animals.

Hence a concept is identifying the birds among the datasets of animals.

So in this example the CONCEPT is a function that can identify the birds.

“Concept learning is a process of inferring a boolean valued function from training examples of its input and output”.

Contd….

Goal of this chapter:

Automatically inferring the general definition of some concept, given examples labeled as members(Yes/True) or nonmembers(No/False) of the concept.

This task is commonly referred to as “concept learning or approximating a boolean valued function from examples”.

Concept Learning Tasks

Consider the example of learning the target concept “Days on which my friend xyz enjoys his favorite water sport”

The task is to learn to predict the value of EnjoySport for an arbitrary day, based on the input dataset.

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

Keywords

Hypothesis: (Assumption/model/General function)

Proposed explanation made on the basis of limited evidence as a starting point for further investigation.

So to represent/express a hypothesis we need some notations.

By using the following ***notations*** we construct the hypothesis,

- Number of constraints i.e, number of features or attributes in the input dataset
- ‘?’ any value it may take
- ‘θ’ No value it can accept
- Specific value
- Example: h = { Sunny, ?, ?, Strong, ?, ? }

Note: hypothesis is a conjunction of all the constraints/attributes.

Notations

X: Set of instances (all possible instances)

D: Set of available training dataset (given input)

x: instance (single row from input dataset)

c: target concept   c : EnjoySport : X ------> { 0, 1}

h: hypothesis -example: h1 = { Sunny, ?, ?, Strong, ?, ? }

H:hypotheses space (has all possible hypothesis)

Note: The goal of the learner is to find a hypothesis h such that h(x) = c(x)

We have to learn c and arrive at h

The Inductive Learning Hypothesis

"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".

- There are 100 input dataset.
- We use only 70 to infer the model/function.
- We assume that this function will correctly predict the result for remaining 30 inputs

Concept Learning as Search

Concept learning can be viewed as the task of **searching** through a large space of **hypotheses** implicitly defined by the **hypothesis representation**.

The **goal** of this search is to find the **hypothesis** that best fit the training examples.

If we view learning as a search problem, then it is natural that our study of learning algorithms will examine different strategies for searching the hypothesis space.

General-to-Specific Ordering of Hypotheses

Will consider the following two hypothesis

h1 = { Sunny, ?, ?, Strong, ?, ? }

h2 = { Sunny, ?, ?, ?, ?, ? }

h2 is more general than h1 i,e. An instance which is classified as true with h1 will also classified as true with h2.

Hence h2 is more general than h1.

Inverse of this rule is more specific i,e. h1 is more specific than h2.

Definition

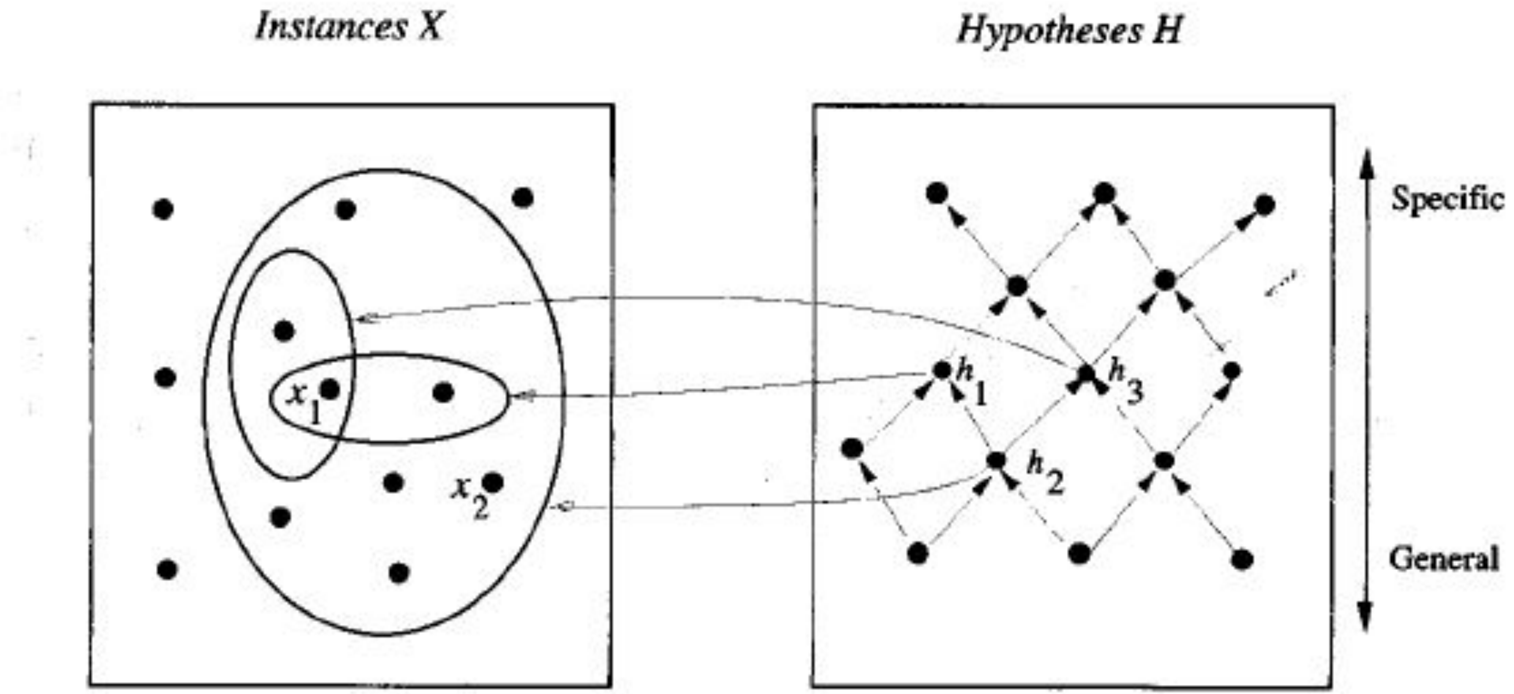
Let h_j and h_k be boolean-valued functions defined over X.

Then h_j is more_general_than_or_equal_to h_k (**written $h_j \geq_g h_k$**) if and only if

$$(\forall x \in X) [(h_k(x) = 1) \text{-------}> (h_j(x) = 1)]$$

For all x belongs to instance space X, when hk(x) result is true then it implies that hj(x) is also true.

More-general-than relation



x_1 = <*Sunny, Warm, High, Strong, Cool, Same*>

x_2 = <*Sunny, Warm, High, Light, Warm, Same*>

h_1 = <*Sunny, ?, ?, Strong, ?, ?*>

h_2 = <*Sunny, ?, ?, ?, ?, ?*>

h_3 = <*Sunny, ?, ?, ?, Cool, ?*>

FIND - S: Finding a Maximally Specific Hypothesis

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 satisfied by x

Then do nothing

Else replace a, in h by the next more general constraint that is satisfied by x

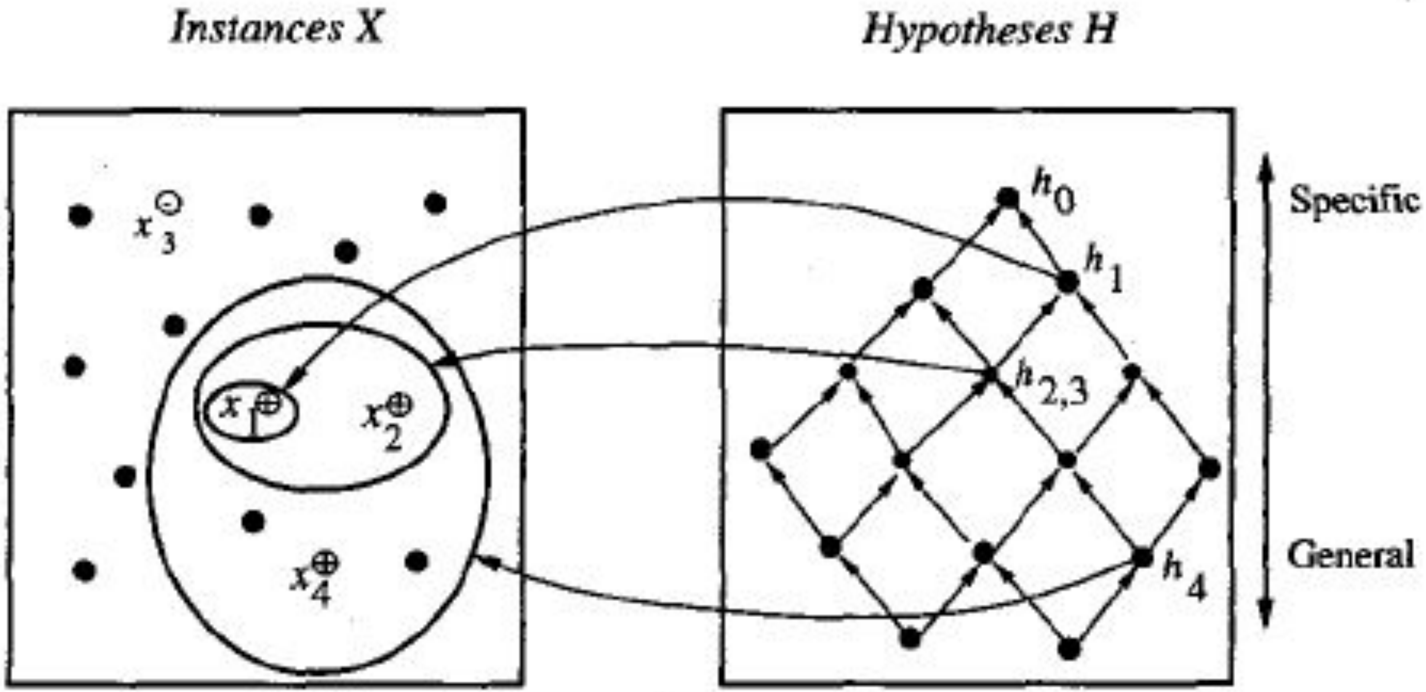
3. Output hypothesis h

Example

Water sport data set

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**

Here **EnjoySport** is the concept to be learnt.



x_1 = <*Sunny Warm Normal Strong Warm Same*>, +
x_2 = <*Sunny Warm High Strong Warm Same*>, +
x_3 = <*Rainy Cold High Strong Warm Change*>, -
x_4 = <*Sunny Warm High Strong Cool Change*>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
h_1 = <*Sunny Warm Normal Strong Warm Same*>
h_2 = <*Sunny Warm ? Strong Warm Same*>
h_3 = <*Sunny Warm ? Strong Warm Same*>
h_4 = <*Sunny Warm ? Strong ? ?* >

Disadvantages

- Has the learner converged to the correct concept?
- Why prefer the most specific hypothesis?
- Are the training examples consistent?
- What if there are several maximally specific consistent hypotheses?

Version Spaces

The version space, denoted $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H **consistent** with the training examples in D.

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

How to Create Version Space

Version space can be generated by using the following algorithm

- LIST THEN ELIMINATE Algorithm
- CANDIDATE ELIMINATION Algorithm

List Then Eliminate

Version Space $\longleftarrow$ a list containing every hypothesis in H

For each training example, { x, c(x) }

remove from VS any hypothesis h for which $h(x) \neq c(x)$

Output the list of hypotheses in VS

Candidate Elimination

Initialize G to set of maximally general hypotheses

Initialize S to set of maximally specific hypotheses

For each training example d, do:

If d is a positive example:

Remove from G any hypothesis inconsistent with d

For each hypothesis s in S that is not consistent with d

Remove s from S

Add to S all minimal generalizations h of s such that

h consistent with d

Some member of G more general than h

Remove from S any hypothesis that is more general than another hypothesis in S

If d is a negative example:

Remove from S any hypothesis inconsistent with d

For each hypothesis g in G that is not consistent with d

Remove g from G

Add to G all minimal specializations h of g such that

h consistent with d

Some member of S more specific than h

Remove from G any hypothesis that is less general than another hypothesis in G

Initialise G ∈ VS to the most general hypothesis: h ← ‹a1,…,an›, (∀i) ai = ?.

Initialise S ∈ VS to the most specific hypothesis: h ← ‹a1,…,an›, (∀i) ai = 0.

FOR each training instance d ∈ D, do:

IF d is a positive example

Remove from G all h that are not consistent with d.

FOR each hypothesis s ∈ S that is not consistent with d, do:

- replace s with all h that are consistent with d, **h >g s, h! ≥g g ∈ G,**
- remove from S all s being more general than other s in S.

IF d is a negative example

Remove from S all h that are not consistent with d.

FOR each hypothesis g ∈ G that is not consistent with d, do:

- replace g with all h that are consistent with d, **g >g h, h >g s ∈ S,**
- remove from G all g being less general than other g in G.

Output hypothesis G and S.

Candidate Elimination: example

S: ~~{<∅, ∅, ∅, ∅, ∅, ∅ >}~~

G: {<?, ?, ?, ?, ?, ?>}

x_1 = <Sunny Warm Normal Strong Warm Same> +

S: ~~{< Sunny Warm Normal Strong Warm Same >}~~

G: {<?, ?, ?, ?, ?, ?>}

x_2 = <Sunny Warm High Strong Warm Same> +

S: {< Sunny Warm ? Strong Warm Same >}

G: {<?, ?, ?, ?, ?, ?>}

Candidate Elimination: example

S: {< Sunny Warm ? Strong Warm Same >}

G: ~~{<?, ?, ?, ?, ?, ?>}~~

x_3 = <Rainy Cold High Strong Warm Change> -

S: ~~{< Sunny Warm ? Strong Warm Same >}~~

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?>, ~~<?,?,?,?,?,Same>~~}

x_4 = <Sunny Warm High Strong Cool Change> +

S: {< Sunny Warm ? Strong ? ? >}

G: {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?> }