Machine Learning

Concept Learning and The General-To-Specific Ordering

Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Overview

- Concept Learning
- Find-S Algorithm
- Version Space (List-then-Eliminate Algorithm)
- Candidate-Elimination Learning Algorithm
- Inductive Bias

Concept Learning

- Concept
  - ‘car’ , ‘bird’
  - ‘situations in which I should study more in order to pass the exam’

- Concept Learning
  – Inferring a boolean-valued function from training examples of its input and output

Concept Learning Task

- Example target concept
  - days on which my friend Aldo enjoys his favorite water sport

- Hypothesis representation
  - ? : any value is acceptable for this attribute
  - Single value (e.g. Warm, Strong etc.)
  - Φ : no value is acceptable

  - ex)
    - < *?, Cold, High, ?, ?, ?* >
    - < *?, ?, ?, ?, ?, ?* > ➡ most general
    - <Φ,Φ,Φ,Φ,Φ,Φ> ➡ most specific

Concept Learning Task (cont.)

- Example: *EnjoySport* learning task
  - Given:
    - Instance X : Possible days, each described by the attributes
    - Hypothesis H : Each hypothesis is described by a conjunction of constraints on the attributes. The constraints may be "?", "Φ", or specific value.
    - Target concept c : *EnjoySport*: $X \rightarrow \{0,1\}$
    - Training Example D
  - Determine:
    - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

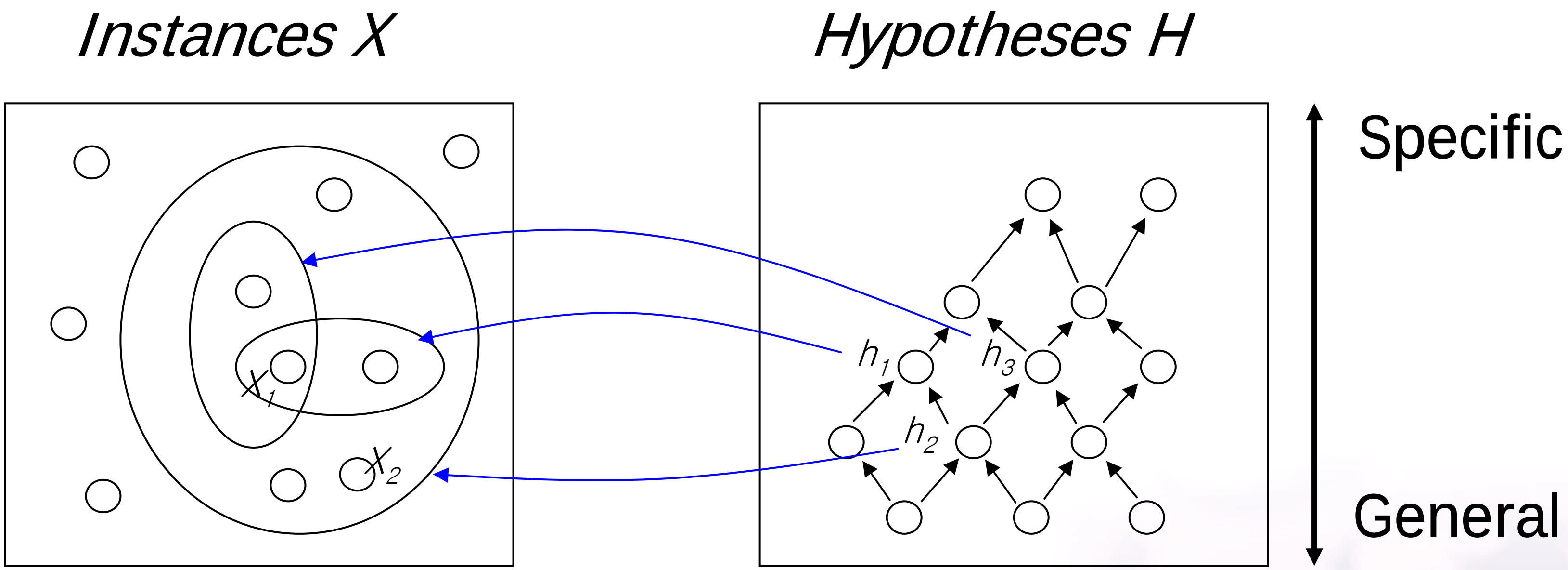
Training examples for EnjoySport

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

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.

General-To-Specific Ordering



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, ? >

Note the subset of instances characterized by h_2 subsumes the subset characterized by h_1, hence h_2 Is more_general_than h_1 !!!

General-To-Specific Ordering (cont.)

- Let h_j and h_k be boolean-valued functions defined over X
  Then,
  - h_j is *more_general_than_or_equal_to* ($h_j \geq_g h_k$) h_k
    if and only if $(\forall x \in X)[(h_k(x)=1) \rightarrow (h_j(x)=1)]$
  - h_j *more_general_than* ($h_j >_g h_k$) h_k
    if and only if $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$

- The $\geq_g$ relation defines a *partial order* over the hypothesis space H (The relation is reflexive, antisymmetric, and transitive).
  "The structure is a partial order" $\Rightarrow$ There may be pairs of hypotheses such as h_1 and h_3, such that $h_1 \not\geq_g h_3$ and $h_3 \not\geq_g h_1$.

Find-S Algorithm

- **Algorithm**

1. Initialize h to be the most specific hypothesis in H
2. 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
3. Output hypothesis h

- **Steps**
  - $h \leftarrow$ < Φ, Φ, Φ, Φ, Φ, Φ > : most specific
  - $h \leftarrow$ < *Sunny,Warm,Normal,Strong,Warm,Same*>
  - $h \leftarrow$ < *Sunny, Warm, ?, Strong, Warm, Same*>
  - $h \leftarrow$ < *Sunny, Warm, ?, Strong, Warm, Same*>
  - $h \leftarrow$ < *Sunny, Warm, ?, Strong, ?, ?* >

- Find-S Algorithm simply ignores every negative example!
- Find-S is guaranteed to output the most specific hypothesis within *H* that is consistent with the positive training examples.

Key Property of Find-S Algorithm

- For hypothesis space described by conjunctions of attribute constraints (such as *H* for the *EnjoySport* task) , Find-S is guaranteed to output the most specific hypothesis within *H* that is consistent with the positive training examples.
- Its 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.

Problems

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

Version Space and The Candidate Elimination Algorithm

- Key idea
  - Output a description of the **set** of *all hypotheses consistent with the training examples.*
- Limit
  - Performs poorly when given noisy training data

    both Candidate Elimination algo. And Find-S

Representation

- Consistent
  - A hypothesis *h* is **consistent** with a set of training examples *D* if and only if $h(x) = c(x)$ for each example $<x, c(x)>$ in *D*.
  - $Consistent(h,D) \equiv (\forall <x,c(x)> \in D) h(x)=c(x)$
- Version Space
  - $VS_{H,D} = \{ h \in H \mid Consistent(h, D) \}$

List-Then-Eliminate Algorithm

- Initializes the version space to contain all hypotheses in *H*, then eliminates any hypothesis found inconsistent with any training example.
- When hypothesis space *H* is finite.
- Exhaustive!

Version Space (Diagram)

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

{ < *Sunny, ?, ?, Strong, ?, ?* > }   { < *Sunny, Warm, ?, ?, ?, ?* > }   { < *?, Warm, ?, Strong, ?, ?* > }

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

Compact Representation of Version Space

- Represented by its most general and least general members. (general/specific boundary)
- general boundary *G*, with respect to hypothesis space *H* and training data *D*, is the set of maximally general members of *H* consistent with *D*.
  - $G \equiv \{ g \in H \mid Consistent(g, D) \wedge (\nexists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$
- specific boundary *S*, with respect to hypothesis space *H* and training data *D*, is the set of maximally specific members of *H* consistent with *D*.
  - $S \equiv \{ s \in H \mid Consistent(s, D) \wedge (\nexists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$
- Version Space representation(thm)
  - $VS_{H,D} = \{ h \in H \mid (\exists s \in S)(\exists g \in G)\ (g \geq_g h \geq_g s)\}$

Candidate-Elimination Learning Algorithm

Initialize *G* to the set of maximally general hypotheses in *H*
Initialize *S* to the set of maximally specific hypotheses in *H*
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* is consistent with *d*, and some member of *G* is 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* is consistent with *d*, and some member of *S* is more specific than *h*
    Remove from *G* any hypothesis that is less general than another hypothesis in *G*

Process making Version Space

S_0:	{ < Φ, Φ, Φ, Φ, Φ, Φ > }
S_1:	{ < *Sunny, Warm, Normal, Strong, Warm, Same* > }
$S_{2,3}$:	{ < *Sunny, Warm, ?, Strong, Warm, Same* > }
S_4:	{ < *Sunny, Warm, ?, Strong, ?, ?* > }
G_4:	{ < *Sunny, ?, ?, ?, ?, ?* > }, { < *?, Warm, ?, ?, ?, ?* > }
G_3:	{ < *Sunny, ?, ?, ?, ?, ?* > }, { < *?, Warm, ?, ?, ?, ?* > }, { < *?, ?, ?, ?, ?, Same* > }
$G_{0,1,2}$:	{ < *?, ?, ?, ?, ?, ?* > }

Process making Version Space (cont.)

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

Remarks on Version Space and Candidate-Elimination

- Will the C-E algorithm converge to the correct hypothesis?
- What training example should the learner request next?
- How can partially learned concepts be used?

(details on next page)

Remarks on Version Space and Candidate-Elimination (cont.)

Will the C-E algorithm converge to the correct hypothesis?

- Converge if...
  - there are no errors in the training examples
  - there is some hypothesis in *H* that correctly describes the target concept
- Error example may result empty version space
- Similar symptom when target concept cannot be described in the hypothesis representation.

What training example should the learner request next?

- The term ***'query'*** to refer to such instances constructed by the learner, which are then classified by an external oracle.
- to find an optimal hypothesis among all hypotheses of *VS* , queries must be classified as positive by some of hypothesis in version space, but negative by others.
- the optimal query is to generate instances that satisfy exactly half the version space.
  - $\lceil \log_2 |VS| \rceil$ experiments required to find correct target function

Remarks on Version Space and Candidate-Elimination (cont.)

How can partially learned concepts be used?

- The instance is classified as positive if and only if the instance satisfies every member of *S*.
- The instance is classified as negative if and only if the instance satisfies none of the members of *G*.
- When Classified as pos. by some members of *VS*, as neg. by the other members of *VS*
  - don't know!!

    (Note that in this case, the Find-S algorithm outputs "negative")
  - Majority voting : not exact (just probability)

Inductive Bias

Question:

- As discussed above we assumed that initial hypothesis space contain the target concept.
- What if the target concept is not in the hypothesis space?

A Biased Hypothesis Space:

- Bias the learner to consider only conjunctive hypotheses.
- Hypothesis space is unable to represent even simple disjunctive target concepts such as "*Sky=Sunny* or *Sky=Cloudy*".
- So, we need more expressive hypothesis space

Unbiased Learner

- Extend hypothesis space to the *power set* of *X(every teachable concept!)*
  - e.g: <*Sunny,?,?,?,?,?*> $\vee$<*Cloudy,?,?,?,?,?*>

- Problem: Unable to generalize beyond the observed examples.
  - Positive example (x_1, x_2, x_2)
  - negative example (x_4, x_5)
  - S:$\{(x_1 \vee x_2 \vee x_2)\}$, G:$\{\neg(x_4 \vee x_5)\}$
  - *S* boundary will always be simply the disjunction of the observed positive examples, while the *G* boundary will always be the negated disjunction of the observed negative examples.
  - The only examples that will be classified by *S* and *G* are the observed training examples themselves.
  - In order to converge to a single, final target concept, we will have to present every single instance in *X* as a training example!

Futility of Bias-Free Learning

- Property of inductive inference:
  – a learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.
- $(D_c \wedge x_i) \succ L(x_i \wedge D_c)$
  – $y \succ z$: z is inductively inferred from y
- $(B \wedge D_c \wedge x_i) \vdash L(x_i \wedge D_c)$
  – $y \vdash z$: z follows deductively from y

cf) L : An inductive learning algorithm

$L(x_i \wedge D_c)$: the classification that L assigns to x_i after learning from the training data D_c

Inductive Bias

- Consider
  - concept learning algo. L
  - instance X, target concept c
  - training examples $D_c=\{<x, c(x)>\}$
  - Let $L(X_i, D_c)$ denote the classification assigned to the instance x_i by L after training on the dada D_c.

- *Definition*:
  - inductive bias B of L is minimal set of assertion B such that for any target concept c and corresponding training example D_c
  - $\forall(x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i \wedge D_c)]$

- Inductive bias of C-E algorithm
  - The target concept c is contained in the given hypothesis space H.

Inductive Bias (cont.)

- Advantage of inductive bias
  - provides nonprocedural means of characterizing their policy for generalizing beyond the observed data
  - comparison of different learners according to the strength of the inductive bias
- Consider three learning algorithms, which are listed from weakest to strongest bias.
  1. Rote-learning : storing each observed training example in memory.
     If the instance is found in memory, the store classification is returned.
     Inductive bias : nothing – **Weakest bias**
  2. Candidate-Elimination algo : new instances are classified only in the case where all members of the current version space agree in the classification.
     Inductive bias : Target concept can be represented in its hypothesis space

Inductive Bias (cont.)

3. Find-S : find the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

Inductive bias : Target concept can be represented in its hypothesis space + All instances are negative instances unless the opposite is entailed by its other knowledge – **Strongest bias**

- More strongly biased methods make more inductive leaps, classifying a greater proportion of unseen instances!!