*20CS6037-001*

*Machine Learning*

*Anca Ralescu*

**Concept Learning**

**The Concept Learning Problem (informal statement):** Given a sample of

**+**  examples and

**-** examples

(training examples)

of a category learn (acquire/ infer) a definition for that category.

Selection of the **attributes** and their values (features) is important.

**Inductive bias**: **making assumptions about the representation of the concept.**

Function view of the concept learning task: infer a boolean function

**Training examples  🡪  {0,1}**

More precisely,

X: set of  instances (described in terms of the same attribute set)

**c**: a concept       **c: X 🡪 {0,1}**

        such that

**c(x) =**

**1 if x is an instance of c;**

**0 otherwise**

Example of a learning problem: The *EnjoySportDay*

|  |  |
| --- | --- |
| **Attributes**  **(input)** | **Values** |
| Sky | Sunny(S),  Rainy(R), \*, nil |
| Temp | Warm(W), Cold(C),\*, nil |
| Hum | Normal(N), High(H), \*, nil |
| Wind | Strong(S), \*, nil |
| Water | Warm(W), Cool(C), \*, nil |
| Forecast | Same(S), Change(C), \*, nil |
|  | |
| **Attributes**  **(output)** | **Values** |
| EnjoySportDay | +, - |

**where**

* \* (any value)
* nil (no value)

Training Data set for *EnjoySportsDay* Problem

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ex. | Sky | Temp | Hum | Wind | Water | Forecast | *Enjoy*  *Sport*  *Day* |
| 1 | Sunny | Warm | Normal | Strong | Warm | Same | **+** |
| 2 | Sunny | Warm | High | Strong | Warm | Same | **+** |
| 3 | Rainy | Cold | High | Strong | Warm | Change | **-** |
| 4 | Sunny | Warm | High | Strong | Cool | Change | **+** |

A **hypothesis, h** is a  **conjunction of constraints** on  the values of attributes

For example, the hypothesis   **<Rainy, Warm, High, \*, \*, \*>** describes the "*sky as rainy, temperature as warm, humidity as high while the wind, water temperature and forecast can be anything***"**

**h: X -> {0,1}**

such that

**h(x) = 1 if x is a + example;**

**h(x) = 0 otherwise.**

 We say that an instance **x satisfies h** if **h(x) =1.**

**Partial order on the hypotheses space**

Given two hypotheses, **h** and **h'** we say that

* h is **more-general-or-equal-than** h' if and only if ***for each instance x in X which satisfies h' it follows that x satisfies h***.
* We write this as: **h >=g h'  <=> For all x in X  ( h'(x) =1  -> h(x) = 1)**  **>=**g  is a **partial order relation** on H, the space of ***all possible  hypotheses*** (for a given problem) : there may be hypotheses, h and h' such that neither **h >=g h' nor h' >=g h** are true.
* h is **more-general-than** h' if and only if h **more-general-or-equal-than** h' but h' is not **more-general-or-equal-than** h.

We write this as:

**h >g h'  <=> (h >=g h' ) & NOT(h' >=g h)**

* **h' is more-specific-than h if** h is **more-general-than** h' we also say that .
* We write this as **h' =<g h <=> h >=g h'**
* Two special hypotheses

**h\* =    <\*, \* , \*, \*, \*, \*>   =   anything**

**h**nil **= < nil, nil, nil, nil, nil, nil> =  no value**

* With **h\*** and  **h**nil and the partial order relation, H acquires  a **lattice structure:**

          Given any two hypotheses h1 and h2 in H there exist hypotheses h3  and h4 such that

* + h3 =<g h1 , h3 =<g h2 and h3 is the ***most general hypothesis*** with this property;
  + h4 >=g  h1 ,  h4 >=g  h2 and h4 is the ***most specific hypothesis*** with this property.

 Examples:

**<Sunny, \*, \*, Strong, \*, \*> =<g  <Sunny, \*, \*,  \* , \*, \*>**

**The Concept Learning Problem (formal statement):**

* Given
  + **X** a collection of instances described by a set of attributes
  + **H** a collection of hypotheses
  + **c** a target concept, **c: X -->{0,1}**
  + **D** a collection of training  (+ and -)  examples
* Find
  + **h** in **H** such that **h(x) = c(x)** for all **x in X.**

**The inductive learning hypothesis**

**Any hypothesis which accounts (correctly classifies) a sufficientluy large training set will correctly  classify examples not previously observed.**

**The inductive bias:**  making assumptions (restricting) on thewhat the hypothesis space can represent.  For example, here each **hypothesis is a conjunction of constraints on the attribute values.**

The **inductive learning hypothesis** +  the **inductive bias** allow us to actually learn.

Three algorithms, a**ll of which are search-type which exploit the structure of the hypothesis space.**

1. **Find-S**
2. **Version Space**

* **List-then-eliminate**
* **Candidate elimination**

**The Find-S Algorithm**

 This algorithm finds the **maximally specific hypothesis consistent with the training examples**.

**Idea:** **Start with the most specific and update this iteratively when a new + example (only) not already covered by the hypothesis is presented.**

**Find-S Algorithm**

* **h = (h1,...,hn)  <-  hnil** ( **hnil**  is most specific hypothesis in H )
* For each positive example **e**+ = (**e**+1,....,**e**+n)

For each *i = 1,..., n*

    If **hi**= **e**+i

    then do nothing

    else replace **hi** by the **next more general constrain**t that is satisfied by x

* Output the hypothesis  **h.**

Note that for the current problem the replacement by the next more general constraint means one of the following two situations:

* replace **nil** by one of the attribute values in the example
* replace an attribute value by \*

Tracing Find-S for the *EnjoySportsDay* problem:

|  |  |  |
| --- | --- | --- |
| **Current hypothesis h** | **Example: +/-** | **Action** |
| **h =   hnil =** (nil nil nil  nil nil  nil) | (S W N S W S) **+** | h = (S W N S W S) |
| (S W N S W S) | (S W H S W S) **+** | replace the value for Hum by \*  h = (S W \* S W S) |
| (S W \* S W S) | (R  C  H S W C ) **-** | ignore |
| (S W \* S W S) | (S W H S C C) **+** | replace values for Water and for Forecast by \*  (S W \* S \* \*) |

Issues in Find-S:

* **What is good about Find-S?**
* It is guaranteed to output the most specific hypothesis within H which is consistent with the training set
* Final hypothesis will also be consistent with the negative example provided that H does contain c
* **What cannot be answered about Find-S:**
  + Convergence: Is the final hypothesis the only correct hypothesis?
  + Why prefer the most specific hypothesis?
  + It assumes that training set is consistent (no noise)
  + Is the maximally specific hypothesis always unique?

**The VersionSpace**

* **Consistent hypothesis:**
* D training set, h is consistent with D iff for all pairs **(x, c(x)), h(x) = c(x).**

* **Version Space with respect to  H and D (VSH,D) :**
* the collection of hypotheses in H consistent with D

**The LIST-THEN-ELIMINATE ALGORITHM**

**Idea: Eliminate from the version space all hypotheses inconsistent with D.**

* Initialize **VSH,D** to H   (Wow!!!!)
* For each (x, c(x) ) in D
* Remove h from **VSH,D** if h(x)  is not equal to c(x)
* Ouput **VSH,D**
* What is good about **LIST-THEN-ELIMINATE?**
* It will not output only one hypothesis - but a collection of all those that are consistent with D.
* What is "bad" about **LIST-THEN-ELIMINATE?**
* The initialization step: setting the version space to H.  Even when H, and therefore **VSH,D** is finite it can still be quite expensive to enumerate all the hypotheses consistent with D.

This leads to

The **CANDIDATE-ELIMINATION** algorithm

Idea - Exploit the structure of H in order to achieve a more **compact** representation for  **VSH,D**.

More precisely, represent  **VSH,D** by two bounds :

* S -set :  the **specific boundary** with respect to H and D : the collection of maximally specific (minimally general) hypotheses from H consistent with D.

**S = { h in H ; (h is consistent with D ) and (there is no hypothesis h' in H such that ( h >g h' ) and ( h' is consistent with D)}**

* G-set:  the **general boundary** - with respect to H and D: the collection of maximally general (minimally specfic) hypotheses from H consistent with D.

**G = { h in H ; (h is consistent with D ) and (there is no hypothesis h' in H such that ( h' >g h ) and ( h' is consistent with D)}**

The **CANDIDATE-ELIMINATION** uses + and - examples to update the S-set and the  G-set  in a way which leads to the **shrinking** of the version space.

* Initialize G : G =**h\***
* Initialize S : S = **hnil**
* For each training example ***d*** do
* **if d is a positive example (pup)**

Update G: remove from G all hypotheses inconsistent with ***d***

Update S: Remove ***s*** from **S** if **s** is inconsistent with ***d***

* Obtain all minimal generalizations ***hs*** of  ***s*** consistent with  ***d***
* Add to **S** all  ***hs*** (that is all minimal generalizations consistent with ***d***)  and less general than some member of **G**)**.**
* Remove from S any hypothesis that  is more general than  some hypothesis in **S.**
* **If d is a negative example(nup)**
  + Update S: Remove any hypothesis from **S** inconsistent with ***d.***
  + Update G: Remove ***g*** from **G** if it is inconsistent to ***d.***
* Generate all minimal specializations ***hg*** of ***g*** (that is all specializations for which there is a member in S more specific)***.***
* Add  ***hg***  to **G** if ***hg***  is consistent to ***d*** and some hypothesis in **S** is more specific than ***hg*** .
* Remove those which are less general than some hypotheses in G.

Illustration of the **CANDIDATE-ELIMINATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **G-set** | **S-set** | **Example: +/ -** | **Action** |
| (\* \* \* \* \* \* ) | (nil nil nil nil nil nil) | (S W N S W S)  + | update S |
| (\* \* \* \* \* \* ) | (S W N S W S) | (S W H S W S) + | update S |
| (\* \* \* \* \* \* ) | (S W \* S W S) | (R C H S W C) - | update G**(1)** |
| (S \* \* \* \* \* )  (\* W \* \* \* \* )  (\* \*  \* \* W \*)  (\* \* \* \* \* S ) | (S W \* S W S) | (S W H S C C) + | Update both**(2)** |
| **(S \* \* \* \* \* )**  **(\* W \* \* \* \* )** | **(S W \*S \*  \*)** |  |  |

1. Remove hypotheses inconsistent with the example: indeed the hypothesis (\* \* \* \* \* \*) will predict that (R C H S W C)  is + .

Therefore, remove  (\* \* \* \* \* \*) and replace it by specializations. There are many minimal specializations for (\* \* \* \* \* \* ) : However, most of those will be inconsistent with the previously seen **+** examples - that is with **S**.  More precisely:

|  |  |
| --- | --- |
| Hypothesis  minimally  specialized from  (\* \* \* \* \* \*) | Consistent with S?  S-set=(S W \* S W S) |
| **(S  \* \* \* \* \*)** | **YES** |
| (R \* \* \* \* \* ) | NO |
| **(\* W \* \* \* \* )** | **YES** |
| (\* C \* \* \* \* ) | NO |
| (\* \* N \* \* \*) | NO |
| (\* \* H \* \* \* ) | NO |
| **(\* \* \* S \* \*)** | **YES** |
| **(\* \* \* \* W \*)** | **YES** |
| (\* \* \* \* C \*) | NO |
| (**\* \* \* \* \* S)** | **YES** |
| (\* \* \* \* \* C) | NO |

**(2)** Update S:  Since the new positive example and S-set are inconsistent the hypothesis from the S-set is removed and replaced by the minimally generalized **h**s = **(S W \*S \*  \*)** which is not more general than any of the hypotheses in the G-set.

**(2)** Update G: Hypotheses (\* \*  \* \* W \*), (\* \* \* \* \* S ) from the G-set are inconsistent with the G-set.  Therefore they are removed.

**Version Space**

Another example of the version space in the domain of playing cards:

|  |  |
| --- | --- |
| Attribute | Value |
| rank | n: 2-4; face: Q,K, \*, nil |
| Suit/color | clubs, spade (both black)  hearts, diamond (both red) , \*, nil |

Q: How big is the initial versions space (pretty big )

**Answer**: as big as the instance space, that is the collection of all instances that can be generated with these attributes and corresponding attribute values: N attributes (A1, …, AN) such that Ai takes on ki values each, then the size of the instance space is k1 x k2 x … x kn .

In this case we have two attributes, rank which can take on 7 values, and Suit-color that can take on 6 values. Therefore, the instance space has size 7 x 6 = 42.

In this example, we also have the relations

**For attribute Suit:**

**\* >g black >g clubs, spade >g nil**

**\* >g red >g hearts, diamonds**

**For attribute Rank:**

**\* >g n >g 2,3,4 >g nil**

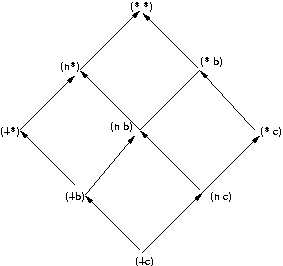
**\* >g face >g Q, K >g nil**

|  |  |
| --- | --- |
| G-set | S-set |
| (\* \*) | (nil nil) |

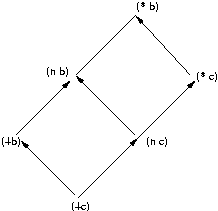
1st positive  example : **"Four of clubs"  = (4c)+**

|  |  |  |  |
| --- | --- | --- | --- |
| G-set | S-set | example | action |
| (\* \*) | (nil nil) | **( 4 c) +** | **update S-set** |
| **(\* \*)** | **( 4 c)** |  |  |

Updated version space:



Negative example: **"Five of hearts" : (5 h ) -**

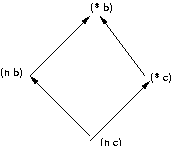


|  |  |  |  |
| --- | --- | --- | --- |
| G-set | S-set | example | action |
| (\* \*) | (nil nil ) | ( 4c) + | update S-set |
| (\* \*) | ( 4c) | **(5h)-** | update  G-set |
| **(\* b)** | **(4c)** |  |  |

Second positive example:  **"Seven of clubs" : (7 c )+**

|  |  |  |  |
| --- | --- | --- | --- |
| G-set | S-set | example | action |
| (\* \*) | (nil nil ) | ( 4c) + | update S-set |
| (\* \*) | ( 4c) | **(5h)-** | update  G-set |
| **(\* b)** | **(4c)** | **(7c) +** | **update S-set** |
| **(\*b)** | **(nc)** |  |  |

Updated version space is:



Issues in the CANDIDATE-ELIMINATION Algorithm:

* **Convergence issues**
* **If**
  1. (1) the training set contains no errors, and
  2. (2) H is correct, that is,  there is some hypothesis in H that describes the concept

**then** CE will converge towards the correct hypothesis.

VS captures the ambiguity in the concept which is **everything between S and G**.

Ideally, G **🡪** c 🡨S

If (1) is not satisfied the algorithm may remove the correct hypothesis.

Inconsistency corresponds to the situation in which VS is empty.

* **Training examples (how to select)**
* **Use of *queries* to guide learning**: these are hypotheses generated by the learning algorithm: **Idea** : select an instance which can reduce as much as possible the version space - that is, either the G-set or the S-set.  That is, it is inconsistent with as many hypotheses from these as possible.  The result is validated by the teacher.
* **Strategy** (most natural for a computer scientist):
  + - * Generate hypotheses such that the version space is halved - this means generating instances  which are satisfied by half of the  hypotheses (candidate set) in the version space.  If the hypotheses are equally likely, this strategy will lead to the shortest sequence of experiments necessary to identify the correct candidate: **log2 |VS|** steps to find the target concept.
        1. **However**, **the strategy itself is computationally expensive**:

**Worst case**: Compare each instance with the concept in order to determine if the instance satisfies the concept: ***m*** instances, ***n*** candidates will lead to ***mn***  steps.

When there is no instance which halves the VS the one coming closest to doing it is selected (this can be formally identified by defining the **information value** of an instance w.r.t. a candidate set).

* 1. **Computational improvement**:
     + - **Factorization** of the version space :
         1. this corresponds to the factorization of the concept into independent concepts, each defined in terms
         2. of non-overlapping subsets of attributes.
  2. For example, in the cards example, or the *GoodSportDay* example, this may mean considering
  3. concepts defined with one attribute only, therefore the original concept would be a "product" of two
  4. or six concepts respectively.
  5. **Factored/Factoring concepts leads to a factored version space**.
  6. The cards example:
  7. c1 : attribute rank : 2-4; Q,K, \* :  **2-4   <g n**   **<g   \***, **Q, K   <g   \***
  8. c2 : attribute color : **c, s   <g b**(black)  **<g   \*** ; **d, h   <g  r**(red)  **<g   \***
  9. Then each of the examples considered above become examples of c1 and c2 respectively as follows:
  10. for c1: 4+, 5-, 7+
  11. for c2: c+, h-, c+
  12. The corresponding version spaces when **(4c)+** is given as an example are:
  13. for c1 is  4 **🡪** n**🡪** \*
  14. for c2 is  **c 🡪 b 🡪 \***

**Definition**

* Two version spaces are said **independent** if membership in one tells **nothing** about membership in the other (does not imply it, nor does it rule it out).
* The **product** of two independent version spaces is formed by their intersection.

**Exercises:**

(1) Develop the version spaces for c1 and c2 as the remaining examples are provided.

Are these independent?  What is the product version space?

(2) Change the second attribute to **suit** (with values c, h, s, d).  Add a third attribute **color** (with values b, r).  Now, consider the three examples defined each with one of the above attributes and construct their version spaces.  Are any of these version spaces independent?  Explain.

**Learning partial concepts**

* When |**VS| > 1**it means that there were insufficient examples to allow for learning the concept fully.  When a new instance, *Instance*,  is presented for recognition the following is the result: Recognition (*Instance*):
  1. (a) positive if *Instance* is classified as positive by every hypothesis in VS
  2. (b) negative is *Instance* is classified as negative by every hypothesis in VS
* otherwise
  1. *Instance* is ambiguous and it can be classified using a "**majority vote**" algorithm.
* **Operational approach**: The structure of H induced by the *more-general-than* partial order implies that it is enough if:
  1. (a) *Instance* is classified as positive by every hypothesis in the S-set.
  2. (b) *Instance* is classified as negative by every hypothesis in the G-set.

**Inductive bias**

* Recall the notion of inductive bias:

**Assumption on the representation of the concept** (conjunctions, disjunctions, mixtures, etc.)

This assumption translates directly into the size of the version space and it may affect the assumption that H contains the target concept.

For example, the concept may be a conjunction of disjunctions (CNF) whereas H may be such that only conjunctions of literals can be represented. (Biased **hypothesis space)**

**Q: What stops us from including everything into H?**

A: In general, the size of the hypothesis space is closely related to the size of the training examples.

The larger H, more examples are needed. And, the most general hypothesis (**Unbiased learner)** space, is actually exponential in the size of the instance space:

**EXAMPLE** (*GoodSportDay)* problem:

Six attributes taking 2, 2, 2, 1, 2, 2 values respectively, for a total of 25 = 32 possible combinations of attribute values.

Considering in addition for each attribute the values nil and \* the total number of possible combinations, or the instance space size, is 3 \* 45 = 3072.

The number of possible concepts that can be defined on this set is the same as the *power set* of the instance space (each concept corresponds to a subset of this set).

Therefore **2|instance space|** concepts are possible.

**However, the conjunctive hypothesis space can represent only a small part of this space.**

Question: **How many?**

Answer: **All the combination of attribute values, including \* but excluding any combination containing nil (conjunction with nil is nil; nil=false),** that is, 1+ 2\*35 = 487.

* **Futility of bias-free learning:** A learner must make prior assumptions about what it is to be learned (*What is it that I am looking for*?)
* **Inductive bias :**

in Find-S: the target concept can be described by H;  by default all instances are negative.

in CE (weaker than Find-S):  the target concept can be represented in H.

Formally, the inductive bias can be defined as follows:

* Let **L** be a learning algorithm for a set of instances **X = {xi, i=1,...,n}**.
* Let **Dc** be the training set for the concept c, **Dc = { (x, c(x)}**.
* Let **L(xi, Dc)** be the result of classifying **xi**, when the learner has been trained on **Dc**.

The **inductive bias is the minimal set of assertions, B, when added to Dc will classify xi according to L:**

**For all xi in X (B AND  Dc AND xi  ENTAIL L(xi, Dc))**

It follows that for **CE** the inductive bias is simply **that c belongs to H**.

**Q: why?**

A: Because of the **universal voting procedure used by the VS to decide the classification of a new instance**.