Machine Learning
Concept Learning

Dr. Pratyay Kuila

Dept. of Computer Science & Engineering
NIT Sikkim, Ravangla-737139

Concept Learning

- The learning involves acquiring general *concepts* from specific training examples.
- Each concept can be thought of as a *Boolean-valued function defined over the larger set* (e.g., a function defined over all animals, whose value is true for birds and false for other animals).
- Consider *the problem of automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept*. This task is commonly referred to as ***concept learning, or approximating*** a Boolean-valued function from examples.
- Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- **Concept learning:** Acquiring the definition of a general category given a sample of positive and negative training examples of the category.
- **Concept learning.** Inferring a Boolean-valued function from training examples of its input and output.

Concept Learning

- Consider the example task of learning the target concept "days on which *Tom* enjoys his favorite water sport."
- The attribute ***EnjoySport*** indicates whether or not *Tom* enjoys his favorite water sport on this day.
- The task is to learn to predict the value of ***EnjoySport*** for an arbitrary day, based on the values of its other attributes.

Table 1: 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

What *hypothesis representation* shall we provide to the learner in this case?

Concept Learning

Table 1: 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

- **Hypothesis representation:** a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.
- Each hypothesis be a vector of six constraints, specifying the values of the six attributes: ***Sky*, *AirTemp*, *Humidity*, *Wind*, *Water***, and ***Forecast***.
- Therefore, a hypothesis can be represented as $<\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{a_3}, \boldsymbol{a_4}, \boldsymbol{a_5}, \boldsymbol{a_6}>$, where, a_i be the possible value of i^{th} attribute.
- For each attribute, the hypothesis will either,
  - a single required value (e.g., *Warm*) for the attribute.
  - "?" that indicates any value is acceptable for this attribute.
  - "Ø" that indicates no value is acceptable for this attribute.

Concept Learning

Table 1: 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

- To illustrate, the hypothesis that Tom 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; is represented by <?, ?, ?, ?, ?, ?>.
- The *most specific possible hypothesis* that no day is a positive example; is represented by <Ø, Ø, Ø, Ø, Ø, Ø>.
- Here, the concept learning task is to learn the set of days for which *EnjoySport* = *Yes*, i.e., describing this set by a conjunction of constraints over the instance attributes.

Notations

- The set of items over which the concept is defined is called the set of instances, which we denote by *X*. In the current example, *X* is the set of all possible days, each represented by the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*.
- In general, each hypothesis ***h*** in ***H*** represents a Boolean-valued function defined over *X*; that is, $\boldsymbol{h : X \rightarrow \{0, 1\}}$.
- If some instance *x* satisfies all the constraints of hypothesis *h*, then *h* classifies *x* as a positive ($h(x) = 1$) example (irrespective of actual target class).
- The *concept* or *function* to be learned is called the *target concept*, which we denote by *c*. In general, *c* can be any Boolean valued function defined over the instances *X*; that is, $\mathbf{c}: \boldsymbol{X} \rightarrow \mathbf{\{0, 1\}}$.

Notations

- In the current example, the *target concept* corresponds to the value of the attribute *EnjoySport* (i.e., $c(x) = 1$ if *EnjoySport* = *Yes*, and $c(x) = 0$ if *EnjoySport* = *No*).
- We use the symbol ***H*** to denote the set of *all possible hypotheses* that the learner may consider regarding the identity of the target concept. Usually *H* can be determined by the hypothesis representation.
- ***The goal*** of the learner is to find a hypothesis ***h*** such that $\boldsymbol{h(x) = c(x)}$ for all $\boldsymbol{x}$ in $\boldsymbol{X}$.
- Therefore, out of the large set *H*, we have to find out the *h* that provides same value as the target concept *c*.

Concept Learning as Search

- As per the Table 1, the attribute ***Wind*** has only one possible value, and ***Sky, AirTemp, Humidity, Water,*** and ***Forecast*** each have two possible values. The instance space *X* contains exactly 2.2.2.1.2.2 = 32 ***distinct instances***.
- There are 4.4.4.3.4.4 = 3072 ***syntactically distinct hypotheses*.**
- Notice, that every hypothesis containing one or more "Ø" symbols represents the empty set of instances; that is, it classifies every instance as negative. Therefore, there are 1 + (3.3.3.2.3.3) = 487 ***semantically distinct*** hypotheses.

- If the attribute ***Sky*** has three possible values, and that ***AirTemp, Humidity, Wind, Water,*** and ***Forecast*** each have two possible values. The instance space *X* contains exactly 3.2.2.2.2.2 = 96 ***distinct instances***. There are 5.4.4.4.4.4 = 5120 ***syntactically distinct hypotheses*** within *H* and the number of ***semantically distinct hypotheses*** is only 1 + (4.3.3.3.3.3) = 973.

Concept Learning as Search

Table 1: 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

***Exercise* 1:** Let one attribute Watercurrent is added in the Table with the values {*Light*, *Moderate*, *Light*, *Strong*} for the four instances respectively.

a) What is the number of possible *instances*?
b) What is the number of *syntactically distinct hypotheses* with the added attribute?
c) What is the number of *semantically distinct hypotheses* with the added attribute?

General-to-Specific Ordering of Hypotheses

- Consider the two hypotheses

$$h_1 = (\text{Sunny}, ?, ?, \text{Strong}, ?, ?)$$
$$h_2 = (\text{Sunny}, ?, ?, ?, ?, ?)$$

- Consider the sets of instances that are classified positive by h_1 and h_2. As h_2 imposes fewer constraints, it classifies more instances as positive.
- In fact, any instance classified positive by h_1 will also be classified positive by h_2. Therefore, we say that h_2 is more general than h_1.
- For any instance x in X and hypothesis h in H, we say that x *satisfies* h if and only if $h(x) = 1$.

General-to-Specific Ordering of Hypotheses

- The ***more_general_than_or_equal_to*** relation can be defined as follow:
- Given the hypotheses h_j and h_k :

  h_j is *more_general_than_or_equal_to* h_k

  if and only if any instance that satisfies h_k also satisfies h_j.

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 $\boldsymbol{h_j \geq_g h_k}$) if and only if

$$(\forall x \in X)[(h_k(x)=1) \rightarrow (h_j(x)=1)]$$

- We will say that h_j is (strictly) ***more_general_than*** h_k**:** **(**written $h_j >_g h_k$) if and only if $(h_j \geq_g h_k) \wedge (h_k \ngeq_g h_j)$.

- We will say that h_j is ***more_specific_than*** h_k when,

  h_k is ***more_general_than*** h_j.

FIND-S: Algorithm

- The FIND-S algorithm can find a *maximally specific hypothesis*.
- It begins with the most specific possible hypothesis in *H*, then generalize this hypothesis each time when it fails to *cover* an observed positive training example.
- We say that a hypothesis "*covers*" a positive example if it correctly classifies the example as positive.

1. Initialize *h* to the most specific hypothesis in *H*.
2. **For** each positive training instance *x*
3. **For** each attribute constraint *a*, in *h*
4. **If** the constraint *a*, is satisfied by *x*
5. Then do nothing
6. **Else** replace *a*, in *h* by the next more general constraint that is satisfied by *x*
7. Output hypothesis *h*.

FIND-S: Algorithm

- To illustrate this algorithm, assume the learner is given the sequence of training examples from the Table of the *EnjoySport* task.
- Initialize h to the most specific hypothesis in H.

$$h = \{ \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \}$$

- Upon observing the first training example from Table, which happens to be a positive example, it becomes clear that our hypothesis 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; namely, the attribute values for this training example, h ={*Sunny*, *Warm*, *Normal*, *Strong*, *Warm*, *Same*}.

	h_0 ={ Ø, Ø, Ø, Ø, Ø, Ø }
x_1 = <*Sunny Warm Normal Strong Warm Same*>, +	h_1 ={ *Sunny*, *Warm*, *Normal*, *Strong*, *Warm*, *Same*}
x_2 = < *Sunny Warm High Strong Warm Same*>, +	h_2 ={ *Sunny*, *Warm*, ?, *Strong*, *Warm*, *Same*}
x_3 = < *Rainy Cold High Strong Warm Change*>, -	h_3 ={ *Sunny*, *Warm*, ?, *Strong*, *Warm*, *Same*}
x_4 = < *Sunny Warm High Strong Cool Change*>, +	h_4 ={ *Sunny*, *Warm*, ?, *Strong*, ?, ?}

FIND-S: Algorithm

- Inconsistent training examples:
  - The training examples may contain some errors or noise.
  - Such inconsistent training examples can severely mislead FIND-S, although it ignores negative examples.
  - An algorithm that could detect when the training data is inconsistent and accommodate such errors is preferable.

- There can be several maximally specific hypotheses consistent with the data.

- It avoids negative instances.

- Although FIND-S outputs a hypothesis from *H*, that is consistent with the training examples, this is just one of many hypotheses from *H* that might fit the training data equally well.

Candidate-Elimination

- The key idea in the CANDIDATE-ELIMINATION algorithm is to output a description of the set of all hypotheses ***consistent*** with the training examples.

- A hypothesis is *consistent* with the training examples if it correctly classifies these examples.

***Definition*:** 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.

$$Consistant(h, D) = (\forall [x, c(x)] \in D)\, h(x) = c(x)$$

- Satisfy vs. Consistent: An example x is said to *satisfy* hypothesis h when $h(x) = 1$, regardless of whether x is a positive or negative example of the target concept. However, whether such an example is *consistent* with h depends on the target concept, and in particular, whether $h(x) = c(x)$.

Candidate-Elimination

- The Candidate-Elimination algorithm represents *the set of all hypotheses* consistent with the observed training examples.

- This subset of all hypotheses is called the *version space* with respect to the hypothesis space *H* and the training examples *D*, because it contains all plausible versions of the target concept.

***Definition*:** 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*.

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

Candidate-Elimination

- The Candidate-Elimination algorithm represents the version space by storing only its *most general members* (labeled *G*) and its *most specific* (labeled *S*).
- Given only these two sets *S* and *G*, it is possible to enumerate all members of the version space as needed by generating the hypotheses that lie between these two sets in the general-to-specific partial ordering over hypotheses.

Definition: The *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*.

$$G \equiv \{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

Definition: The *specific boundary S*, with respect to hypothesis space *H* and training data *D*, is the minimally general (i.e., maximally specific) members of *H* consistent with *D*.

$$S \equiv \{s \in H \mid Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

Candidate-Elimination

***Theorem* (Version space representation theorem):** Let X be an arbitrary set of instances and let H be a set of Boolean-valued hypotheses defined over X. Let $c : X \rightarrow \{0, 1\}$ be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples $\{(x, c(x))\}$. For all X, H, c, and D such that S and G are well defined,

$$VS_{H,D} = \{h \in H \mid (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$$

- It can be shown that the *version space* is precisely the set of hypotheses contained in G, plus those contained in S, plus those that lie between G and S in the partially ordered hypothesis space.

Candidate-Elimination

- It can be shown that the *version space* is precisely the set of hypotheses contained in *G*, plus those contained in *S*, plus those that lie between *G* and *S* in the partially ordered hypothesis space.
- The Candidate-Elimination algorithm begins by initializing the *G* boundary set to contain the most general hypothesis,

$$G_0 = \{(?,?,?,?,?,?)\}$$

  and initializing the *S* boundary set to contain the most specific (least general) hypothesis,

$$S_0 = \{ (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing)\}$$

- These two boundary sets delimit the entire hypothesis space, because every other hypothesis in *H* is both more general than S_0 and more specific than G_0.
- Any hypotheses found inconsistent from the version space with the new training example is eliminated.
- After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

Candidate-Elimination

$$S_0 : \{ (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing) \}$$

$$\Downarrow$$

$$S_1 : \{(Sunny, Warm, Normal, Strong, Warm, Same)\}$$

$$\Downarrow$$

$$S_2 : \{(Sunny, Warm, ?, Strong, Warm, Same)\}$$

$$G_0, G_1, G_2 : \{(?, ?, ?, ?, ?, ?)\}$$

Training examples:

1 . *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*

2 . *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*

*Trace*1: S_0 and G_0 are the initial boundary sets corresponding to the most specific and most general hypotheses. Training examples 1 and 2 force the S boundary to become more general, as in the FIND-S algorithm. More general boundary S_1 is found after trained by example 1 and S_2 is found after trained by example 2. They have no effect on the *G boundary* (thereby, G_0, G_1 and G_2 are same).

Candidate-Elimination

$$S_2, S_3 : \{(Sunny, Warm, ?, Strong, Warm, Same)\}$$

$$G_3 : \{(Sunny, ?, ?, ?, ?, ?), (?, Warm, ?, ?, ?, ?), (?, ?, ?, ?, ?, Same)\}$$

$$G_2 : \{(?, ?, ?, ?, ?, ?)\}$$

Training examples:
3 . *<Rainy, Cold, High, Strong, Warm, Change>, Enjoy Sport* = *No*

Trace 2: Training example 3 is a negative example that forces the G_2 boundary to be specialized to G_3. Note several alternative maximally general hypotheses are included in G_3.

Positive training examples may force the *S* boundary of the version space to become increasingly general. Negative training examples play the complimentary role of forcing the *G* boundary to become increasingly specific.

Candidate-Elimination

$$S_2, S_3 : \{(Sunny, Warm, ?, Strong, Warm, Same)\}$$

$$G_3 : \{(Sunny, ?, ?, ?, ?, ?), (?, Warm, ?, ?, ?, ?), (?, ?, ?, ?, ?, Same)\}$$

$$G_2 : \{(?, ?, ?, ?, ?, ?)\}$$

- Training examples: {<*Rainy*, *Cold*, *High*, *Strong*, *Warm*, *Change*>, *No*}
- There are six attributes that could be specified to specialize G_2, but there only three new hypotheses in G_3.
- For example, the hypothesis h=(?,?,*Normal*,?,?,?) is a minimal specialization of G_2 that is consistent with the negative example, but it is inconsistent with the previously encountered positive examples.
- Therefore, h=(?,?,*Normal*,?,?,?) is not included in G_3.
- The algorithm determines this simply by noting that h is not more general than the current specific boundary, S_2.

Candidate-Elimination

$$S_3 : \{(Sunny, Warm, ?, Strong, Warm, Same)\}$$

$$\downarrow$$

$$S_4 : \{(Sunny, Warm, ?, Strong, ?, ?)\}$$

$$G_4 : \{(Sunny, ?, ?, ?, ?, ?), (?, Warm, ?, ?, ?, ?)\}$$

$$\uparrow$$

$$G_3 : \{(Sunny, ?, ?, ?, ?, ?), (?, Warm, ?, ?, ?, ?), (?, ?, ?, ?, ?, Same)\}$$

Training examples:

4 . *<Sunny, Warm, High, Strong, Cool, Change>, Enjoy Sport = Yes*

***Trace* 3:** The positive training example generalizes the S boundary, from S_3 to S_4. One member of G_3 must also be deleted, because it is no longer more general than the S_4 boundary (based on training example 4).

Candidate-Elimination

$$S_4 : \{(Sunny, Warm, ?, Strong, ?, ?)\}$$

$$\{(Sunny, ?, ?, Strong, ?, ?), (Sunny, Warm, ?, ?, ?, ?), (?, Warm, ?, Strong, ?, ?)\}$$

$$G_4 : \{(Sunny, ?, ?, ?, ?, ?), (?, Warm, ?, ?, ?, ?)\}$$

- The final version space for the *EnjoySport* concept learning problem.
- This learned version space is independent of the sequence in which the training examples are presented (because at 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.

Candidate-Elimination

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

Candidate-Elimination: Inductive Bias

CANDIDATE-ELIMINATION: New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.

FIND-S: This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

- The CANDIDATE-ELIMINATION algorithm is not robust to noisy data or to situations in which the unknown target concept is not expressible in the provided hypothesis space.

Assignments

Assignment 1: Let us add one attribute *Watercurrent* in the Table 1 (*EnjoySport*) with the values {*Light*, *Moderate*, *Strong*, *Strong*} for the four instances respectively.

a) What is the number of *distinct instances*?
b) What is the number of *syntactically distinct hypotheses*?
c) What is the number of *semantically distinct hypotheses*?

Assignment 2: What is the final hypothesis after execution of FIND-S algorithm on the modified training set of Assignment 1?

Assignment 3: What is the *version space* after execution of Candidate-Elimination algorithm on the modified training set of Assignment 1?

Assignment 4: How the hypotheses grow with the addition of a new attribute *A* that takes on *k* possible values?

Books:

1. Chapter 2 of "Machine Learning" by Tom Mitchell, McGraw Hill.