Computational Learning Theory
Concept Learning and Version Spaces

Dimitris Diochnos
School of Computer Science
University of Oklahoma

Outline

Table of Contents

Version Spaces and Algorithms for Concept Learning

The material is based on the PhD thesis of Tom Mitchell [1]. It shows up as a separate chapter in Tom Mitchell's book *Machine Learning* [2, Ch. 2].

Goal 1 (Concept Learning)

Exact identification of the target concept c.

That is, given the hypothesis space $\mathcal{H}$, containing functions $h\colon \mathcal{X} \to \{0, 1\}$ our goal is to achieve:

$$h(x) = c(x), \qquad \text{for all } x \in \mathcal{X}.$$

Inductive Learning Hypothesis.
Any hypothesis h found to approximate well the target function c over a sufficiently large set of training examples will also approximate well the target function over unobserved examples.

Example: Enjoy Sport

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

- *features* or *attributes*

Representation of hypotheses. Conjunction on the instance attributes.

Attribute values:

- single values (e.g., "Sunny")
- any value (we use "?")
- no value (we use "$\emptyset$")

Most General Hypothesis: $\langle ?, ?, ?, ?, ?, ? \rangle$

Most Specific Hypothesis: $\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$

Training examples have the form $(x, c(x))$.

General-to-Specific Ordering of Hypotheses

Consider these two hypotheses:

$$\begin{aligned} h_1 &= \langle \text{Sunny}, ?, ?, \textit{Strong}, ?, ? \rangle \\ h_2 &= \langle \text{Sunny}, ?, ?, ?, ?, ? \rangle \end{aligned}$$

What can we say about the instances that are classified as positive by both h_1 and h_2?

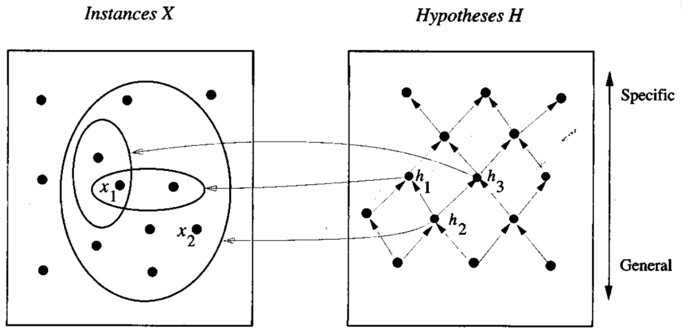
- Any instance that is classified as positive by h_1, will also be classified as positive by h_2

Definition 1

Let $h_j, h_k \in \mathcal{H}$. Then, h_j is **more-general-than-or-equal-to** h_k and write $h_j \geq_g h_k$ iff

$$(\forall x \in \mathcal{X})[(h_k = 1) \Longrightarrow (h_j = 1)]$$

General-to-Specific Ordering of Hypotheses (cont'd)



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, ?>*

- Each hypothesis corresponds to some subset of $\mathcal{X}$. Namely, the subset of instances that it classifies positive.
- The arrows connecting hypotheses in $\mathcal{H}$ correspond to the *more-general-than* relation, with the arrow pointing toward the less general hypothesis.

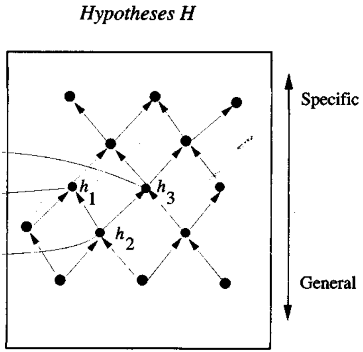
Partial Ordering on Hypotheses

Partial Ordering.

- Reflexive: $(a \leq a)$
- Antisymmetric:
  $(a \leq b) \wedge (b \leq a) \Rightarrow a = b$
- Transitive:
  $(a \leq b) \wedge (b \leq c) \Rightarrow (a \leq c)$

- Some hypotheses
  h_ℓ and h_r may be incomparable;
  e.g., if they are on the same level.

$$(h_\ell \geq_g h_r) \wedge (h_r \geq_g h_\ell)$$



Total Ordering. Also needs totality: $\forall a, b \in \mathcal{X} : (a \leq b)$ or $(b \leq a)$.

FIND-S: Finding a maximally specific hypothesis

Q: How did the algorithm for learning (monotone, or general) conjunctions work when we were using equivalence queries?

---

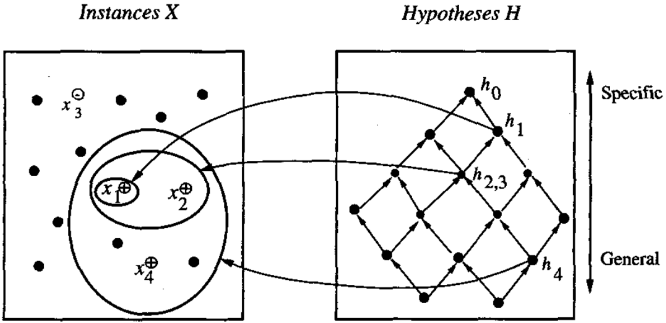
1. Initialize h to be the most specific hypothesis in $\mathcal{H}$.
2. For every positive training instance x:
   - for each attribute constraint a_i in h:
     - If a_i is satisfied by x, do nothing.
     - Otherwise replace a_i in h by the next most general constraint that is satisfied by x.
3. Output h.

Questions.

1. Have we converged to the target?
2. Why do we prefer the most specific hypothesis?
3. Are the training examples consistent?
   - Severely mislead if they have errors or noise.
4. What do we do if there are several maximally specific hypotheses?

Example on the Execution of FIND-S

- Recall the example from Slide 5.



h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

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 ? ?* >

Version Space

Definition 2 (Consistent)

A hypothesis h is **consistent** with a set of training examples D, iff $h(x) = c(x)$ for each example $(x, c(x))$ in D.
We write $\textbf{Consistent}(\boldsymbol{h}, \boldsymbol{D})$ to indicate this.

Consistent $\neq$ Satisfies.

- x satisfies $h \Rightarrow h(x) = 1$.
- h is consistent with $(x, c(x)) \Rightarrow h(x) = c(x)$.

Definition 3 (Version Space)

The **version space**, denoted $VS_{\mathcal{H},D}$, w.r.t. a hypothesis space $\mathcal{H}$ and training examples D, is the subset of hypotheses from $\mathcal{H}$ consistent with the examples in D. In other words,

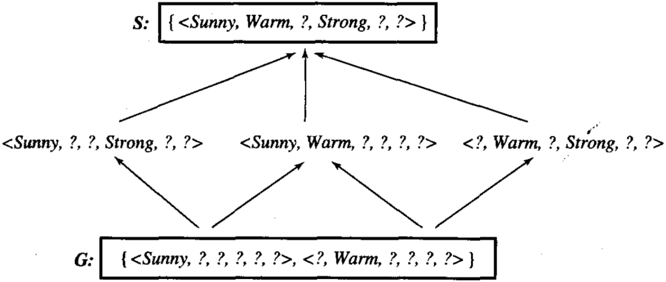
$$VS_{\mathcal{H},D} = \{h \in \mathcal{H} \mid \text{Consistent}(h, D)\}.$$

List-then-Eliminate Algorithm

List-then-Eliminate Algorithm.

1. List all members in the version space.
2. Eliminate inconsistent.
3. Output what is left.

- Apply the algorithm to the EnjoySport example from Slide 5.



General and Specific Boundary of the Version Space

Definition 4 (General Boundary)

The **general boundary** $\boldsymbol{G}$ w.r.t. the hypothesis space $\mathcal{H}$ and training data D, is the set of maximally general members of $\mathcal{H}$ consistent with D
$G \equiv \{g \in \mathcal{H} \mid \text{Consistent}(g, D) \wedge (\nexists g' \in \mathcal{H})[(g' >_g g) \wedge \text{Consistent}(g', D)]\}$.

Definition 5 (Specific Boundary)

The **specific boundary** $\boldsymbol{S}$ w.r.t. the hypothesis space $\mathcal{H}$ and training data D, is the set of minimally general members of $\mathcal{H}$ consistent with D
$S \equiv \{s \in \mathcal{H} \mid \text{Consistent}(s, D) \wedge (\nexists s' \in \mathcal{H})[(s >_g s') \wedge \text{Consistent}(s', D)]\}$.

- might be the case that G and S are not well-defined.
  - e.g., open intervals

Version Space Representation Theorem

Theorem 6 (Version Space Representation Theorem)

Let $\mathcal{X}$ be an arbitrary set of instances and let $\mathcal{H}$ be a set of Boolean-valued hypotheses defined over $\mathcal{X}$. Let $c\colon \mathcal{X} \to \{0, 1\}$ be an arbitrary target concept defined over $\mathcal{X}$ and let D be an arbitrary set of training examples $\{(x, c(x))\}$. For all $\mathcal{X}$, $\mathcal{H}$, c, and D such that S and G are well-defined, we have

$$VS_{\mathcal{H},D} \equiv \{h \in \mathcal{H} \mid (\exists s \in S)(\exists g \in G)[g \geq_g h \geq_g s]\}.$$

Proof Sketch.

($\Leftarrow$) $h \geq_g s \Rightarrow \forall (x_+, 1)$ we have $s(x_+) = 1 = h(x_+)$.
$g \geq_g h \Rightarrow \forall (x_-, 0)$ we have $g(x_-) = 0 \Rightarrow h(x_-) = 0$.
Therefore, Consistent$(h, D) \Rightarrow h \in VS_{\mathcal{H},D}$.

($\Rightarrow$) We know that $h \in VS_{\mathcal{H},D}$. Now look at all maximal chains that go through h. First look at specializations. (Similar argument for $g \in G$.)

- If none is consistent, then $h \in S$.
- If there is at least one consistent hypothesis, follow that path and do so until we can no longer specialize further to a consistent hypothesis h'. This last consistent hypothesis belongs to S. $\square$

Candidate-Elimination Algorithm

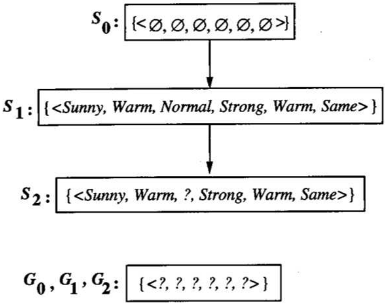
- The version space representation theorem gives rise to the following algorithm that maintains only the S and G sets.

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

Application of the CANDIDATE-ELIMINATION Algorithm

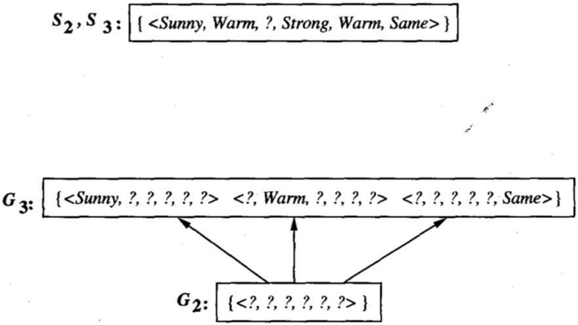
Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



- After processing the first 2 examples.

Application of the CANDIDATE-ELIMINATION Algorithm

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



- After processing the third example.

Application of the Candidate-Elimination Algorithm

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

S_3: {<*Sunny, Warm, ?, Strong, Warm, Same*>}

↓

S_4: {<*Sunny, Warm, ?, Strong, ?, ?*>}

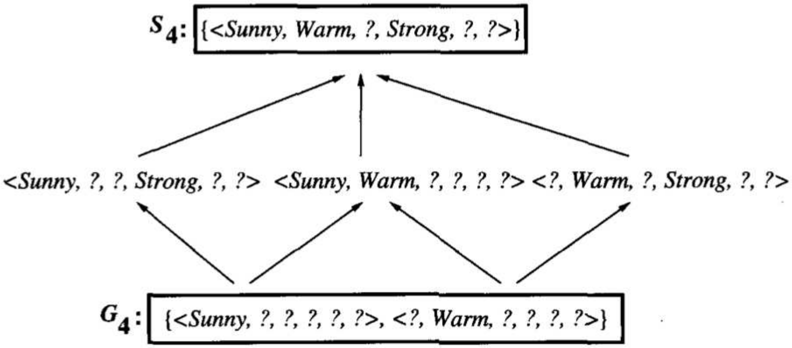
G_4: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*>}

↑

G_3: {<*Sunny, ?, ?, ?, ?, ?*> <*?, Warm, ?, ?, ?, ?*> <*?, ?, ?, ?, ?, Same*>}

- After processing the fourth example.

Application of the CANDIDATE-ELIMINATION Algorithm



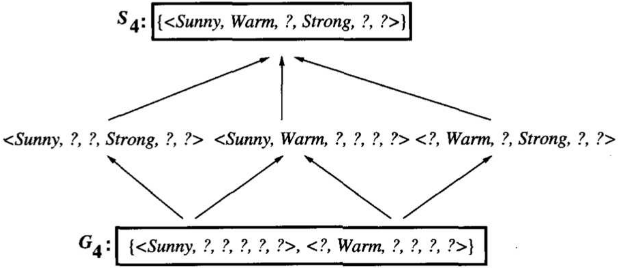
- The final version space.

Convergence of the Candidate-Elimination Algorithm

We need:

1. No errors on the training examples.
2. There exists at least one $h \in \mathcal{H}$ that correctly describes $c \in \mathcal{C}$. *(realizability assumption)*
    - Target concept c exactly learned when S and G converge to a single identical hypothesis.

What Training Example Should the Learner Request Next?



What is a good query for our version space above?

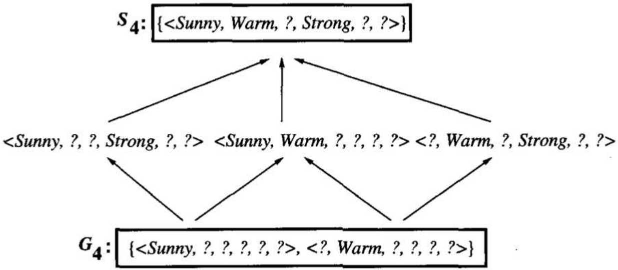
- There are some coordinates where we either have a "?", or a specific value. If we specialize such a coordinate to a different value, either the "?" is correct, or the initial specific value.

Consider the query $q = (\mathit{Rainy}, \mathit{Warm}, \mathit{Normal}, \mathit{Strong}, \mathit{Warm}, \mathit{Same})$.

- If this is positive, we eliminate 4 hypotheses.
- If this is negative, we eliminate 2 hypotheses.

Can we do better than that?

What Training Example Should the Learner Request Next?

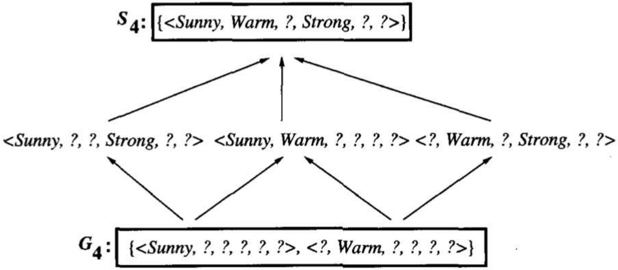


A better query for our version space above.
Consider the query $q' = (Sunny, Warm, Normal, Light, Warm, Same)$.

- If this is positive, we eliminate 3 hypotheses.
- If this is negative, again we eliminate 3 hypotheses.

In general, *what is a good strategy for creating queries?*

What Training Example Should the Learner Request Next?



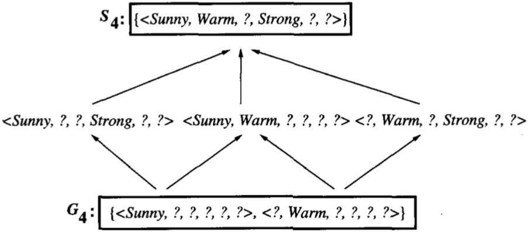
Optimal Strategy for asking queries.

- Generate instances that satisfy exactly half of the hypotheses in the version space.

This results in $\lceil \log_2 (|\mathcal{H}|) \rceil$ queries in the worst case.

- Connected to the HALVING algorithm that is used in *online learning*.

Using Partially Learned Concepts



Say we want to classify x_1 = (Sunny, Warm, Normal, Strong, Cool, Change).

- Everything that is left classifies this example as positive.

Similarly, x_2 = (Rainy, Cold, ...) is classified negative.
In other situations we might be split:

- x_3 = (Sunny, Warm, Normal, Light, Warm, Same) $\Rightarrow$ (half +, half -)
- x_4 = (Sunny, Cold, Normal, Strong, Warm, Same) $\Rightarrow$ (2 say +, 4 say -)

Abstain from predictions: We may say "I don't know".
Halving algorithm: Predict according to the majority vote.

Discussion

- Our goal is to be able to make good predictions on unseen data.
- In other words, we want to be able to generalize well to unseen data.

There are several issues that one can discuss.

- What if $c \notin \mathcal{H}$?
- How does $|\mathcal{H}|$ influence generalization?
- How does $|\mathcal{H}|$ influence the number of training examples that are needed/sufficient for our purposes? (PAC Learning)

Discussion: $c \notin \mathcal{H}$ can be an issue...

- $c \notin \mathcal{H}$ can easily be a real issue:

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

If we use conjunctions (like earlier), then the most specific hypothesis that is consistent with the first two examples is:

h = (?, Warm, Normal, Strong, Cool, Change)

- This function is already overly general for the problem at hand!
  - The third example is classified as positive, whereas it is negative.

One idea: Make the hypothesis space more expressive.

- For example, allow any (Boolean) function in the hypothesis space.
  - This solution comes with drawbacks....

Discussion: Allowing any function in $\mathcal{H}$ has issues...

Say the learner sees the positive examples x_1, x_2, x_3, and the negative examples x_4, x_5. Then,

$$S = \{(x_1 \vee x_2 \vee x_3)\}$$
$$G = \{\neg(x_4 \vee x_5)\}$$

- We can only predict correctly on instances that we have seen before.
- Trying to take the majority vote on instances that we have not seen before, gives a 50-50 score!

Bias-Free Learning is Futile

A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

The CANDIDATE-ELIMINATION algorithm was able to generalize to unseen examples based on the assumption that the target concept could be represented as a conjunction of attribute values.

More generally:

- What rules/policy does the learner follow in order to generalize beyond the training data?
  - It is useful to know the inductive bias that different learning algorithms have.
- What kind of assumptions are needed, so that we can deduce the label that an algorithm gives to an instance x, given a particular training set D?
  - That is called the *inductive bias*.

Inductive Bias

Definition 7 (Inductive Bias)

Consider a concept learning algorithm L for the set of instances $\mathcal{X}$. Let c be an arbitrary concept defined over $\mathcal{X}$, and let $D_c = \{(x, c(x))\}$ be an arbitrary set of training examples of c. Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on the data D_c. The **inductive bias** of L is any minimal set of assertions $\boldsymbol{B}$ such that for any target concept c and corresponding training examples D_c

$$(\forall x_i \in \mathcal{X})[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$

ROTE-LEARNER: No bias (Abstains to predict on previously unseen instances)

CANDIDATE-ELIMINATION: $c \in \mathcal{H}$ (All members of the version space agree on the prediction, otherwise abstains.)

FIND-S: $c \in \mathcal{H}$ and moreover all instances are negative unless the opposite is entailed by other knowledge. (*Never* abstains)

References

[1] Tom M. Mitchell. *Version spaces: An approach to concept learning.* PhD thesis, Electrical Engineering Dept., Stanford University, Stanford, CA, 1979.

[2] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.