

Intro. to machine learning (CSI 5325)

Lecture 3: concept learning

Greg Hamerly

Fall 2008

Some content from Tom Mitchell.

- 1 Concept learning
- 2 Candidate elimination algorithm (recap)
- 3 Picking new examples
- 4 The need for inductive bias
- 5 Where we are going

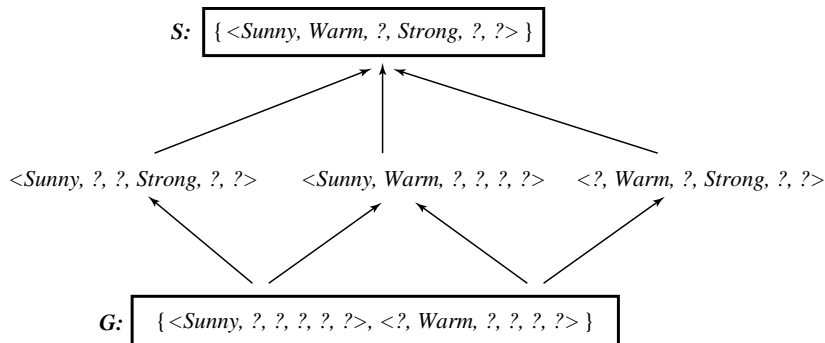
The definition of concept learning

Concept learning is learning a function which has a boolean-valued output.

$$f : X \rightarrow \{0, 1\}$$

Many machine learning approaches use this simplistic binary view of the world.

Example Version Space



Representing Version Spaces

The **General boundary**, G , of version space $VS_{H,D}$ is the set of its maximally general members

The **Specific boundary**, S , of version space $VS_{H,D}$ is the set of its maximally specific members

Every member of the version space lies between these boundaries

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

where $x \geq y$ means x is more general or equal to y

Candidate Elimination Algorithm

Initialize:

- $G \leftarrow$ maximally general hypotheses in H
- $S \leftarrow$ maximally specific hypotheses in H

For each training example d , do

- If d is a positive example, adjust sets G and S
- If d is a negative example, adjust sets G and S

Candidate Elimination Algorithm – positive example

For a positive example d :

- 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

Candidate Elimination Algorithm – negative example

For a negative example d :

- 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

Tracing the Candidate Elimination Algorithm

$$S_0: \boxed{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle}$$

$$G_0: \boxed{\langle ?, ?, ?, ?, ?, ? \rangle}$$

What about noisy data?

What would happen to the candidate elimination algorithm if it encountered an incorrectly-labeled example?

The algorithm removes every hypothesis that is inconsistent with some training example. Therefore, the true concept will be removed!

What does this show about our assumptions that

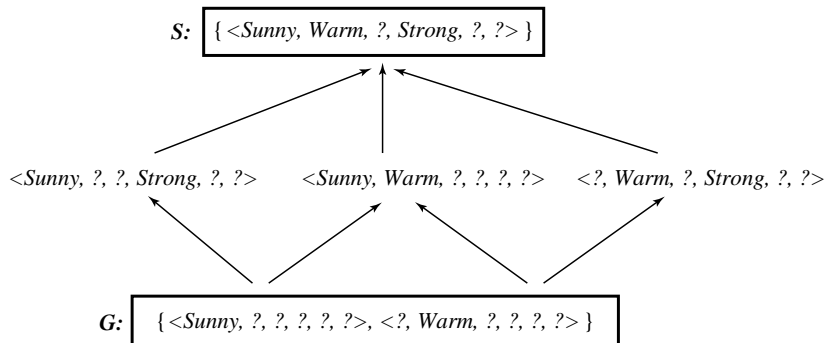
- the data has no noise?
- the hypothesis space contains the correct hypothesis?

What ordering on the training examples?

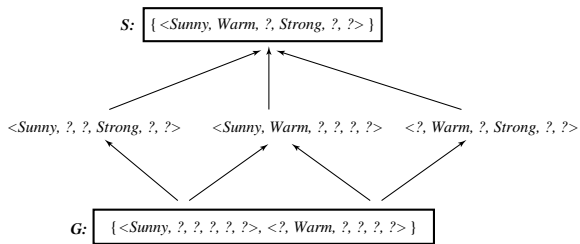
Does the order of the training data matter?

- in correctness of the algorithm?
- in efficiency of the algorithm?

What training example should be next?



How Should These Be Classified?



<Sunny Warm Normal Strong Cool Change>

<Rainy Cool Normal Light Warm Same>

<Sunny Warm Normal Light Warm Same>

What Justifies this Inductive Leap?

- + $\langle \textit{Sunny Warm Normal Strong Cool Change} \rangle$
- + $\langle \textit{Sunny Warm Normal Light Warm Same} \rangle$

$S : \langle \textit{Sunny Warm Normal ? ? ?} \rangle$

Why believe we can classify the unseen

$\langle \textit{Sunny Warm Normal Strong Warm Same} \rangle$

What is inductive bias?

In short, it's limiting our hypothesis space.

What does the word bias imply?

So far, we've limited our hypothesis space to be a conjunction of attribute constraints.

An UNbiased learner

Idea: Choose H that expresses every teachable concept (i.e., H is the power set of X)

Consider $H' =$ disjunctions, conjunctions, negations over previous H . E.g.,

$$\langle \textit{Sunny Warm Normal} \ ? \ ? \ ? \rangle \vee \neg \langle \ ? \ ? \ ? \ ? \ ? \ \textit{Change} \rangle$$

What are S , G in this case?

$S \leftarrow$

$G \leftarrow$

Inductive Bias

Consider

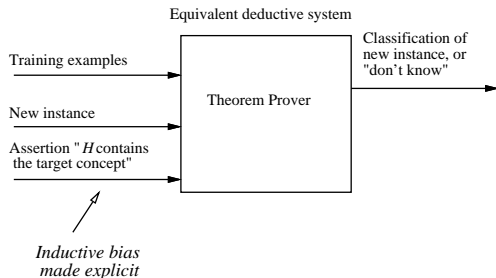
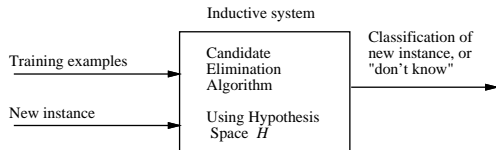
- concept learning algorithm L
- instances X , target concept c
- training examples $D_c = \{\langle x, c(x) \rangle\}$
- let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on data D_c .

Definition: The **inductive bias** of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c

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

where $A \vdash B$ means A logically entails B

Inductive Systems and Equivalent Deductive Systems



Three Learners with Different Biases

- 1 *Rote learner*: Store examples, Classify x iff it matches previously observed example.
- 2 *Version space candidate elimination algorithm*
- 3 *Find-S*

Inductive bias and tic-tac-toe

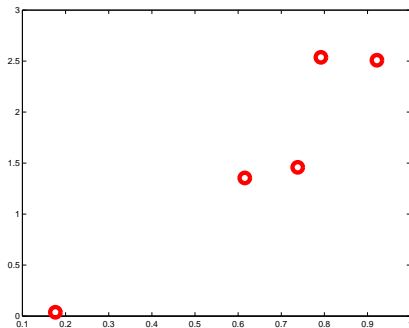
What is the inductive bias of the TTT player you're writing?

Inductive bias and curve fitting

Suppose we were trying to learn the regression function

$$f : \mathbb{R} \rightarrow \mathbb{R}$$

What is the best curve (f) through these points?



Induction leads to deduction

Notice that often we want to not just learn a concept, but use that concept to do something new.

We typically *induce* a concept/model/hypothesis from specific training examples.

We often then use that learned concept to *deduce* facts on new data.

Summary Points

- 1 Concept learning as search through H
- 2 General-to-specific ordering over H
- 3 Version space candidate elimination algorithm
- 4 S and G boundaries characterize learner's uncertainty
- 5 Learner can generate useful queries
- 6 Inductive leaps possible only if learner is biased
- 7 Inductive learners can be modelled by equivalent deductive systems

Next steps

We will look at models which are widely used for learning.

- decision trees
- neural networks
- linear models and basis/kernel expansions
- generative (probabilistic) models
- etc.

For each thing, try to keep in mind things like:

- what is the hypothesis space?
- what is the inductive bias?
- is the algorithm robust to noise?
- how well can the algorithm learn?
- how quickly can the algorithm learn?

Tools for the course

- \LaTeX – emphasis on quality writeups of experiments
- MATLAB – available on PCs and my linux servers (ask me for an account)
 - good for statistics, linear algebra, data wrangling, plotting/graphing, prototyping, and running experiments
- Weka – Java-based machine learning toolkit
- Datasets – lots available! I'll post links on the course page as needed.