Lecture 2: Concept Learning. Version Space

- Kinds of learning
- Supervised learning and concept learning
- Version space
- Candidate elimination algorithm
- The need for bias

Kinds of learning

Based on the information available:

- Supervised learning
- Reinforcement learning
- Unsupervised learning

Based on the role of the learner

- Passive learning
- Active learning

Supervised learning

- Training experience: a set of **labeled examples** of the form

$$\langle v_1\, v_2\, \ldots v_n, o \rangle,$$

  where v_i are values for *input variables* and o is the *output*
- This implies the existence of a "teacher" who knows the right answers
- What to learn: A **function** $f : V_1 \times V_2 \times \cdots \times V_n \rightarrow O$, which maps the input variables into the output domain
- Performance measure: minimize the error (loss function) on the training examples

Reinforcement learning

- Training experience: interaction with an environment; the agent receives a numerical reward signal
- E.g., a trading agent in a market; the reward signal is the profit
- What to learn: a way of behaving that leads to a lot of reward in the long run
- Performance measure: maximize the long-term reward

Unsupervised learning

- Training experience: unlabeled data
- What to learn: interesting associations in the data
- E.g., image segmentation, clustering
- Performance measure: ?
- Often there is no single correct answer

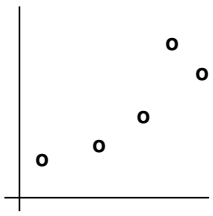
Passive and active learning

- Traditionally, learning algorithms have been *passive learners*
  Data $\rightarrow$ Learner $\rightarrow$ Model
- *Active learners* are instead allowed to query the environment
  - Ask questions
  - Perform experiments
- Open issues: how query the environment optimally? how to account for the cost of queries?

Supervised (inductive) learning

Assume somebody gives us **labeled examples** of the form $\langle v_1, v_2, \ldots v_n, o \rangle$, where:

- v_i are values for *input variables* or *attributes* or *features*
- o is the *output value*



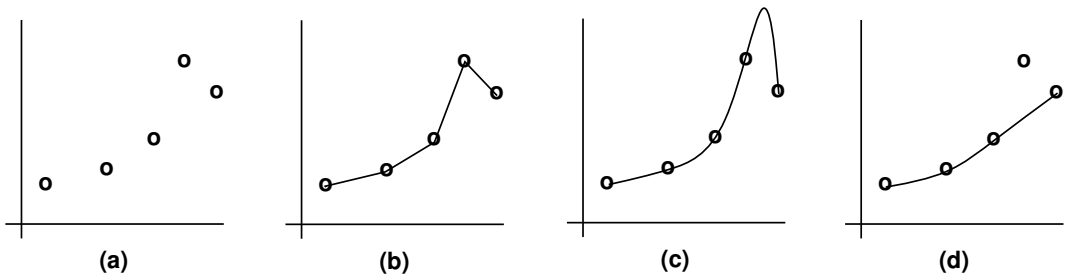
(a)

Goal: Find a *target function* $f : V_1 \times V_2 \times \cdots \times V_n \rightarrow O$, which maps the input variables into the output domain and fits well the examples

Kinds of supervised learning

- If the output domain is Boolean, this problem is called *concept learning*
- If the output domain is discrete, this problem is called *classification*
- If the output is continuous, the problem is known as *regression* or *function approximation*.

The big question



There are an infinite number of functions that can fit the training points!

How do we find a *good* one?

I.e., one that will give approximately correct values for *unseen* data.

Choosing a target function representation

- The learner usually considers a *class of hypothesis*
  E.g., decision trees, artificial neural networks, linear combinations of 3rd-degree polynomials, linear combinations of the input, etc.
- Learning methods *search* through the class of hypothesis
- If there are several hypothesis that fit the data about the same, the learner has a preference function over them
  E.g., Choose the simplest hypothesis - **Occam's razor**

Issues:

- What should the hypothesis class be?
- What search method should we use?
- What is a "good" hypothesis?

Concept learning

Inferring a Boolean function from labeled training examples

Example: "user profile" for web browsing

Domain	Platform	Browser	Day	Screen	Continent	Click?
edu	Mac	Net3	Mon	XVGA	America	Yes
com	Mac	NetCom	Tue	XVGA	America	Yes
com	PC	IE	Sat	VGA	Europe	No
org	Unix	Net2	Wed	XVGA	America	Yes

What is the general concept?

Assumption of the day: the training examples are perfect (no noise)

Representing hypotheses

Many possible representations!

Consider hypotheses h formed as a conjunction of constraints on attributes

Each constraint can be

- a specific value (e.g., *Domain* = *com*)
- don't care (e.g., *Domain* = *?*)
- no value allowed (e.g.,*Domain*=$\emptyset$)

For example,

Domain	*Platform*	*Browser*	*Day*	*Screen*	*Continent*
$\langle$ *com*	*?*	*?*	*Sun*	*?*	*America* $\rangle$

Prototypical concept learning task

Given:

- Instances X: Possible days, each described by the attributes:
  - *Domain* $\in \{$*com, edu, org*$\}$
  - *Platform* $\in \{$*Mac, PC, Unix*$\}$
  - *Browser* $\in \{$*Net2, Net3, NetCom, IE*$\}$
  - *Day* $\in \{$ *Monday, ... Sunday* $\}$
  - *Screen* $\in \{$*VGA, XVGA*$\}$
  - *Continent* $\in \{$ *America, Europe, Africa, Asia, Australia* $\}$

  How many possible instances are there?
- Target function (or concept) c: *Click*: $X \rightarrow \{0, 1\}$

  How many concepts are possible?

Given (continued)

- Hypotheses H: Conjunctions of literals.

  E.g., $\langle$*?, Cold, High, ?, ?, ?* $\rangle$

  How many syntactically distinct hypotheses are there?

  How many are semantically distinct?
- Training examples D: Positive and negative examples of the target function

$$\langle x_1, c(x_1)\rangle, \ldots \langle x_m, c(x_m)\rangle$$

Determine:

A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

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.

Why would this be true?

Sampling: statistical theory for inferring population parameters from samples

Concept learning as search in hypothesis space

The hypotheses can be partially ordered under the *more-general-than-or-equal-to* relation ($\geq_g$)

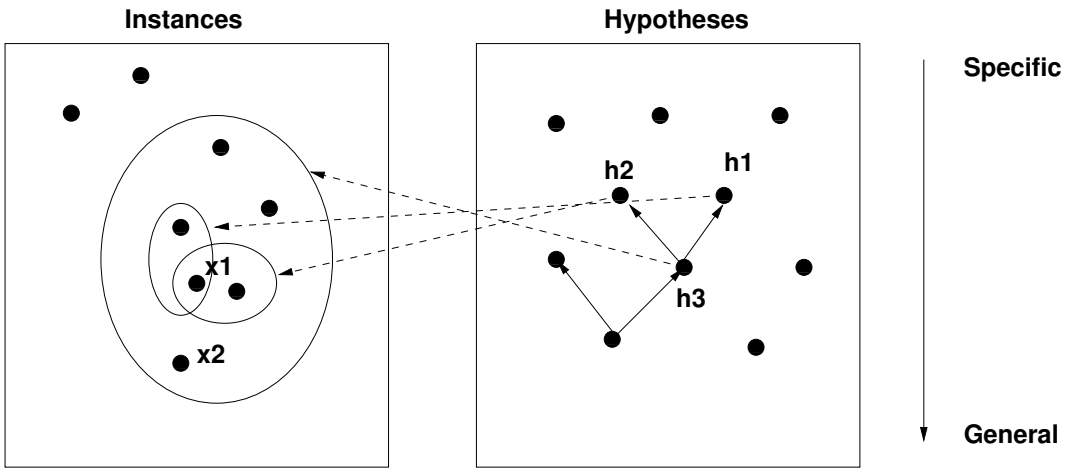
Definition: $h_1 \geq_g h_2$ if and only if
$\forall x \in X, h_2(x) = 1 \rightarrow h_1(x) = 1$

E.g.

- $h_1 = \langle edu, Mac, ?, Mon, ?, ? \rangle$
- $h_2 = \langle edu, Mac, IE, Mon, ?, Europe \rangle$

Why is this a partial ordering?

Partial ordering on hypothesis space



x1 = <edu,Mac,IE,Mon,VGA,Eur>

x2 = <edu,PC,IE,Mon,VGA,Eur>

h1 = <edu,Mac,?,?,?,Eur>

h2 = <edu,?,IE,?,?,Eur>

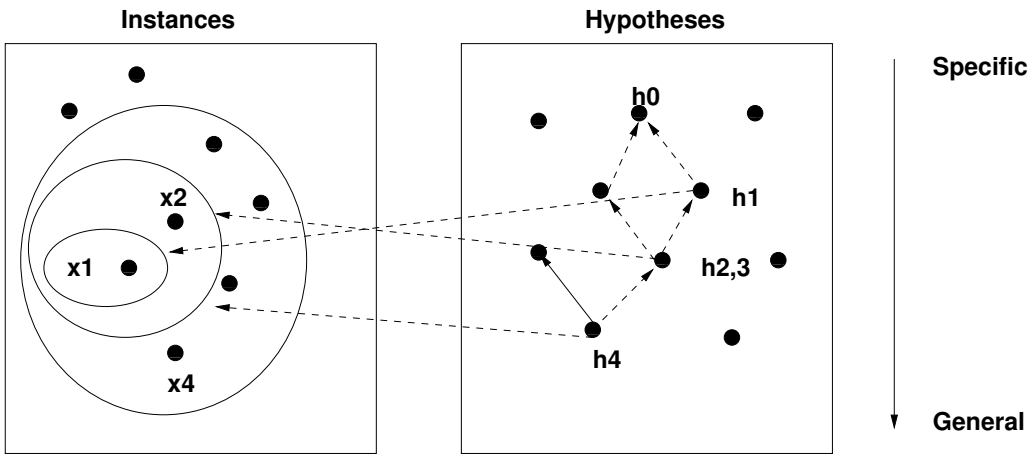
h3 = <edu,?,?,?,?,Eur>

Find-S: Finding a maximally specific hypothesis

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x, do
   - For each attribute constraint a_i in h, do

     If x is not covered by h, replace a_i by the next more general constraint that is satisfied by x
3. Output hypothesis h

Hypothesis space search by Find-S



x1 = <edu,Mac,Net3,Mon,XVGA,America>, +

x2 = <com,Mac,Net3,Tue,XVGA,America>, +

x3 = <com,PC,IE,Sat,VGA,Eur>, -

x4 = <org,Unix,Net2,Wed,XVGA,America>, +

h0 = <0,0,0,0,0,0>

h1 = <edu,Mac,Net3,Mon,XVGA,America>

h2 = <?,Mac,Net3,?,XVGA,America>

h3 = <?,Mac,Net3,?,XVGA,America>

h4 = <?,?,?,?,XVGA,America>

Complaints about Find-S

- Convergence: cannot tell whether it has learned concept
- Consistency: cannot tell when training data inconsistent
- Picks a maximally specific h (why?)
- Depending on H, there might be several consistent specific hypotheses

Version space

A hypothesis h is **consistent** with a set of training examples D of target concept c if and only if $h(x) = c(x)$ for every training example $\langle x, c(x) \rangle$ in D.

The **version space**, $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.

How can we compute the version space?

- Obvious idea: list-then-eliminate - impractical!
- Candidate elimination (Mitchell)

Representing the version space

Key idea: keep only the *boundary sets*, exploiting the partial ordering of the hypotheses space.

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

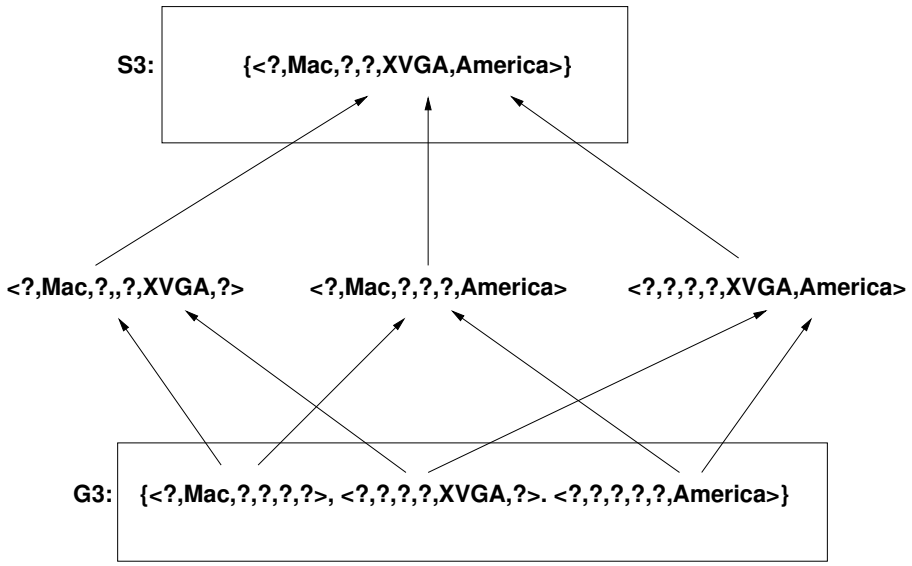
The **specific boundary**, S, of version space $VS_{H,D}$ is the set of its maximally specific members

Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H | \exists s \in S, \exists g \in G) \text{s.t.} g \geq h \geq s)\}$$

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

Example version space



The *candidate elimination algorithm* is designed to change the version space based on training examples.

Example (1)

Example: <edu,Mac,Net3,Mon,XVGA,America>, +

S0: **{<0,0,0,0,0,0>}**

G0: **{<?,?,?,?,?,?>}**

Which boundary should we adjust and how?

Example (2)

S1: {<edu,Mac,Net3,Mon,XVGA,America>}

G1: {<?,?,?,?,?,?>}

Example: <com,Mac,NetCom,Tue,XVGA,America>, +

Example (3)

S2: {<?,Mac,?,?,XVGA,America>}

G2: {<?,?,?,?,?,?>}

Candidate elimination algorithm

$G \leftarrow$ maximally general hypotheses in H

$S \leftarrow$ maximally specific hypotheses in H

For each training example d, do

- If d is a positive example
  - 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 $\exists g \in G, g \geq h$
    - $*$ Remove from S any hypothesis that is more general than another hypothesis in S

Example (4)

S2: **{<?,Mac,?,?,XVGA,America>}**

G2: **{<?,?,?,?,?,?>}**

Example: <com,PC,IE,Sat,VGA,Eur>, -

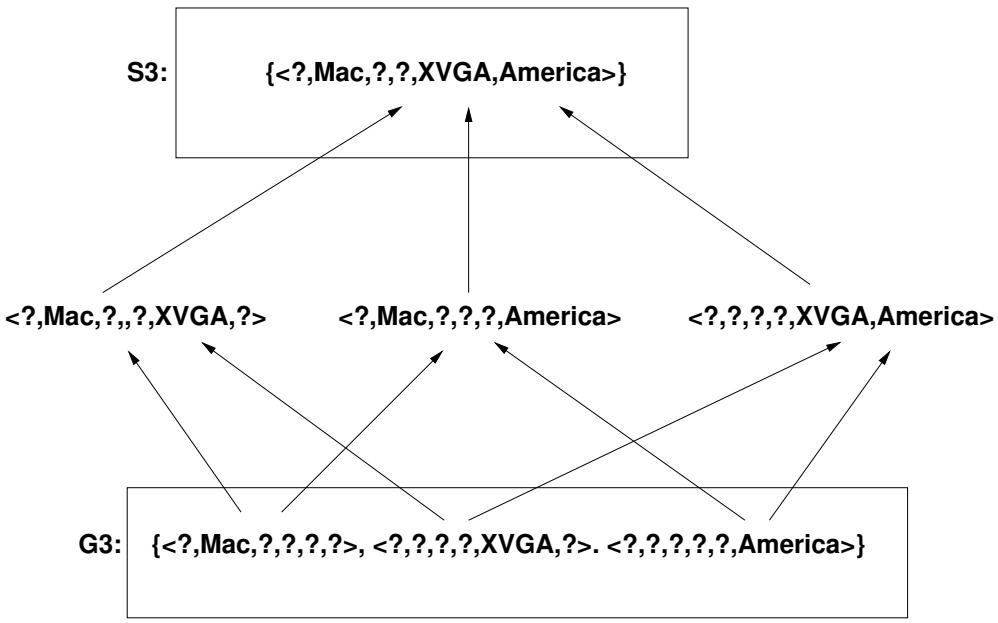
Example (5)

S3: {<?,Mac,?,?,XVGA,America>}

G3: {<?,Mac,?,?,?,?>, <?,?,?,?,XVGA,?>. <?,?,?,?,?,America>}

- 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

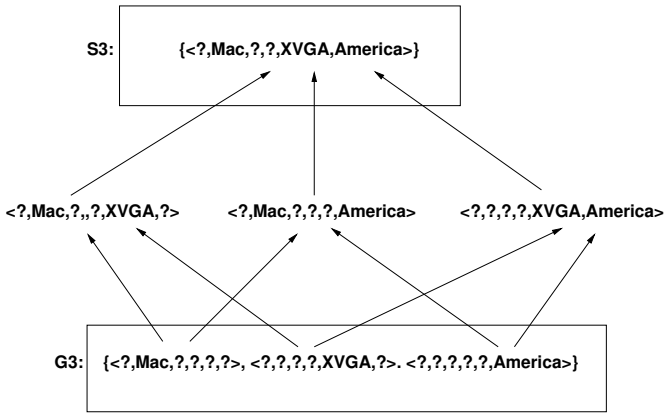
Active learning using version space



What should be the best new example?

Using partially learned concepts

Given the version space:



How should these be classified?

edu	*Mac*	*IE*	*Fri*	*XVGA*	*America*
com	*PC*	*NetCom*	*Wed*	*VGA*	*Europe*
org	*Unix*	*Net2*	*Wed*	*XVGA*	*America*

Sample complexity of candidate elimination

Let the concept be $A_1 = true$. Let the instances be described by n Boolean attributes, and consider the following sequence of examples:

- $A_1 = true \wedge A_2 = true \wedge \ldots \wedge A_{n-1} = false \wedge A_n = false$
- $A_1 = true \wedge A_2 = true \wedge \ldots \wedge A_{n-1} = false \wedge A_n = true$
- $A_1 = true \wedge A_2 = true \wedge \ldots \wedge A_{n-1} = true \wedge A_n = false$
- $A_1 = true \wedge A_2 = true \wedge \ldots \wedge A_{n-1} = false \wedge A_n = false$
- ...

How many instances are needed to find the concept?

Candidate elimination has exponential sample complexity!

Bias in concept learning

Bias is defined as any criteria (other than consistency with the training data) used to select one hypothesis over another.

Note: Inductive bias $\neq$ statistical bias!

Sources of bias

- Hypothesis language - *representational bias*
  E.g., Our hypothesis language only allows conjunctions
- Search (generalization) algorithm - *algorithmic bias* E.g., Find-S searches hypotheses from specific to general

An unbiased learner

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

In our case: consider $H' =$ disjunctions, conjunctions, negations over previous H.
E.g.,*(Platform = Unix* $\vee$ *Platform = Mac)* $\wedge\neg$ *(Platform = PC)*

Given positive instances $x_1, \ldots x_i$ and negative instances $y_1, \ldots y_j$, what are S, G in this case?

$$S \leftarrow x_1 \vee \ldots \vee x_i$$
$$G \leftarrow \neg y_1 \wedge \ldots \neg y_j$$

Bias-free learning does not allow any generalization beyond the training instances!

Bias is necessary!!!

An unbiased generalization algorithm (e.g., version space candidate elimination) that uses an unbiased hypothesis space (e.g., all Boolean functions) can never go beyond memorizing the training instances

The power of a learning system comes from the "goodness" of its biases!

Where do biases come from?

- Knowledge about the domain
- Knowledge about the source of the training data
- Intended use of the learned concept
- Simplicity and generality (e.g., Occam's razor)

Summary

- Concept learning can be viewed as search through some hypothesis space H
- Version space candidate elimination algorithm takes advantage of the partial ordering of hypotheses (general-to-specific)
- The specific and general boundaries characterize the uncertainty of the learner
- Learner can generate useful queries
- Learner can use partially learned concepts to label new examples
- Inductive leaps are possible **only if the learner is biased**
- Bias is a constant theme in machine learning!