CS578 – Interactive and Transparent Machine Learning

Topic: Concept Learning

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

MOTIVATION

- Induce a general function from specific training examples
  - Concept: spam; training examples: emails labeled as spam/~spam
  - Concept: flu; training examples: patient records labeled as flu/~flu
  - Concept: positive; training examples: reviews labeled as positive/negative
  - …
- Goal: induce a general function that fits to the training data well and generalizes well to unseen/future data

Problem Formulation

- Define a space of hypotheses / functions
- Search for a hypothesis/function that fits well to the training data
- To search efficiently, utilize the structure of the hypothesis space
  - In this chapter, we will utilize *general-to-specific ordering* of hypotheses

Concept Learning

- Given
  - A concept
  - Training data: examples that are
    - Described by attributes
    - Annotated as to whether they are a member of the given concept
- Infer
  - A classification function
  - In this lecture, a boolean-valued function

A Simple Example

- Three features
  - Weight: Light, Heavy
  - Color: Red, Green, Blue
- Target
  - Yes, No
- How many possible objects / instances?
- How many possible hypotheses (functions) are there for this domain? What are they?
- When I tell you that <Light, Red> is 'Yes', how many hypotheses left? Can you tell me whether <Light, Blue> is a 'Yes' or 'No'?

LET'S ASSUME THAT

- Our hypothesis allows only conjunction of features
- A feature
  - Can either have a one specific value, or
  - Is ignored completely
- We pick one label as our target; anything else is assumed to belong to the other label(s)
- Two special hypotheses: 1) 'No' to everything, 2) 'Yes' to everything
- Examples:
  - < ϕ, ϕ>: 'No' to everything
  - <?, ?>: 'Yes' to everything
  - <Light, Red>: 'Yes' to <Light, Red> and 'No' to everything else
  - <Light, ?>: 'Yes' to anything that is Light; ignore Color; everything else is 'No'
  - Combination of one or more of ϕ with other feature values is still equivalent < ϕ, ϕ> to 'No' to everything. E.g., < ϕ, Red> is 'No' to everything
- How many possible hypotheses can we represent?

EnjoySport?

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

TABLE 2.2
The *EnjoySport* concept learning task.

Credit: Tom Mitchell, Machine Learning

Hypothesis Representation For EnjoySport

- *h(x)* = Yes if EnjoySport is Yes and No otherwise
- A conjunction (and) of constraints on the attributes
  - <Sky, AirTemp, Humidity, Wind, Water, Forecast>
  - ? indicates any value is acceptable
  - A specific value means it has to be that value
  - ϕ means no value is acceptable
- For example <Sunny, ?, ?, Strong, ?, ?> means
  - Sky has to be Sunny, Wind has to be Strong, and other attributes can be any value

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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Credit: Tom Mitchell, Machine Learning

Why a Boolean-valued Function?

- We will learn about the core ideas of learning
  - Instance space
  - Hypothesis space
  - Version space
  - Inductive bias
- A Boolean-valued function makes it easier for us to understand these core concepts

Most-general and Most-specific

- Most-general hypothesis, i.e., the hypothesis where $h(x) = \text{Yes}\ \forall x \in X$
  - <?, ?, ?, ?, ?, ?>
- Most-specific hypothesis, i.e., the hypothesis where $h(x) = \text{No}\ \forall x \in X$
  - $<\phi, \phi, \phi, \phi, \phi, \phi>$

'More-General' Relation

- 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 has $h_j \geq h_k$) if and only if
  - $(\forall x \in X)\left[h_k(x) = Yes \Rightarrow h_j(x) = Yes\right]$
  - That is, whenever h_k says positive, h_j also says positive; h_j might say positive to other instances that h_k says negative

True Concept

- Let the true concept be $c(x)$
- We do not know what $c(x)$ is
- All we have is a training dataset D that consists of $\langle x, c(x)\rangle$ pairs
- We define a hypothesis space H, for which we hope $c \in H$, and we search for $h \in H$ such that
  - $h(x) = c(x)\ \forall x \in D$

The Instance and Hypothesis Space

- Six attributes:
  - Sky has three possible values, others have two possible values
- Total number of possible instances
  - $3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96$
- One hypothesis
  - Each attribute is ?, ϕ, or a specific value
- Total number of syntactically-different hypotheses
  - $5 \times 4 \times 4 \times 4 \times 4 \times 4 = 5120$
- Any hypothesis that contains at least one ϕ has the same meaning; i.e., it classifies all instances as negative
- Total number of semantically-different hypotheses
  - $(4 \times 3 \times 3 \times 3 \times 3 \times 3) + 1 = 973$
- How do we search this space efficiently?

ALGORITHMS

- Find-S
- List-Then-Eliminate
- Candidate-Elimination

Find-S

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

     If the constraint a_i in h is satisfied by x

     Then do nothing

     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Credit: Tom Mitchell, Machine Learning

FIND-S TRACE

$h_0 = \langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$

$x_1 = \langle$*Sunny Warm Normal Strong Warm Same*$\rangle$, + $\quad h_1 = \langle$*Sunny Warm Normal Strong Warm Same*$\rangle$

$x_2 = \langle$*Sunny Warm High Strong Warm Same*$\rangle$, + $\quad h_2 = \langle$*Sunny Warm ? Strong Warm Same*$\rangle$

$x_3 = \langle$*Rainy Cold High Strong Warm Change*$\rangle$, - $\quad h_3 = \langle$*Sunny Warm ? Strong Warm Same*$\rangle$

$x_4 = \langle$*Sunny Warm High Strong Cool Change*$\rangle$, + $\quad h_4 = \langle$*Sunny Warm ? Strong ? ?*$\rangle$

Credit: Tom Mitchell, Machine Learning

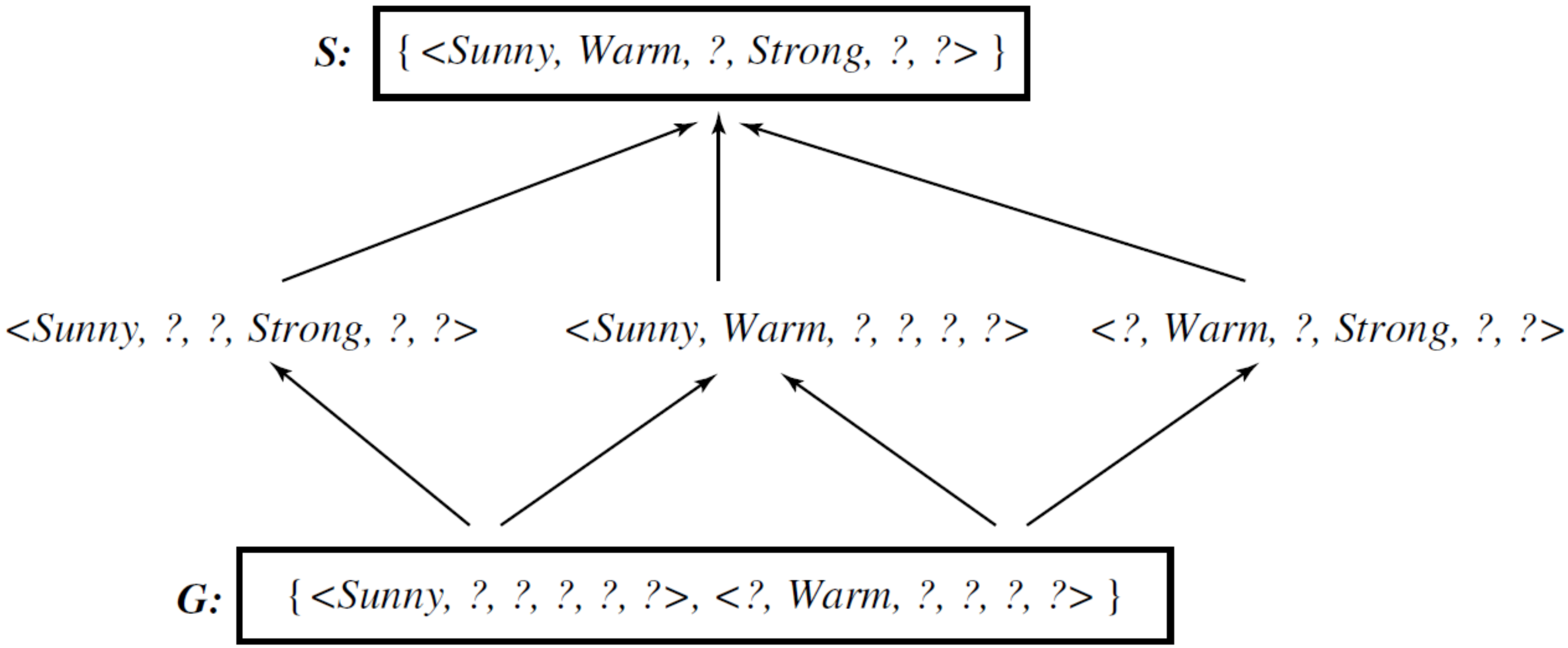
Problems with Find-S

- How do we know the $h(.)$ returned by Find-S is the actual $c(.)$?
- If there are more than one hypotheses that are consistent with data, Find-S finds only the most specific one. Why settle for the most specific one? For example, why not the most general one?
- What happens when the training data has errors?
- What if the most-specific hypothesis is not unique?

Version Space

- A hypothesis $h()$ is consistent with a set of training examples D and a target concept $c()$ if and only if $h()$ agrees with $c()$ on each training example in D
  - $Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) h(x) = c(x)$
- Version space with respect to hypothesis space H and dataset D is the set of all hypotheses in H that are consistent with all examples in D
  - $VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$

Version Space



Credit: Tom Mitchell, Machine Learning

List-Then-Eliminate

1. $VS \leftarrow$ a list of containing every hypothesis in H
2. For each training example, $\langle x, c(x) \rangle$
    1. Remove from VS any hypothesis h for which $h(x) \neq c(x)$
3. Output VS

- Advantages
    - Outputs all consistent hypotheses
- Disadvantages
    - Need to list all possible hypotheses
        - Impossible for infinite hypothesis spaces
        - Impractical for large hypothesis spaces

Candidate-Elimination

- **General boundary *G***: the set of maximally-general consistent hypotheses.
  - $G \equiv \{g \in H | Consistent(g, D) \wedge (\neg\exists g' \in H)[(g' > g) \wedge Consistent(g', D)]\}$
- **Specific boundary *S*:** the set of maximally-specific consistent hypotheses.
  - $\mathrm{S} \equiv \{s \in H | Consistent(s, D) \wedge (\neg\exists s' \in H)[(s > s') \wedge Consistent(s', D)]\}$
- **Version space representation theorem:** for every consistent hypothesis there is at least one more-general-or-equal-to hypothesis in G and there is at least one more-specific-or-equal-to hypothesis in S
  - $VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$
- **Candidate elimination algorithm**
  - Start with S that has only the most-specific hypothesis and G that has only the most-general hypothesis, and modify S and G with each training data

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 that is 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

- …
- For each training example *d*, do
  - If *d* is a positive example
    - [see previous slide]
  - *If d is a negative example*
    - Remove from *S* any hypothesis that is 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 more specific than another hypothesis in *G*

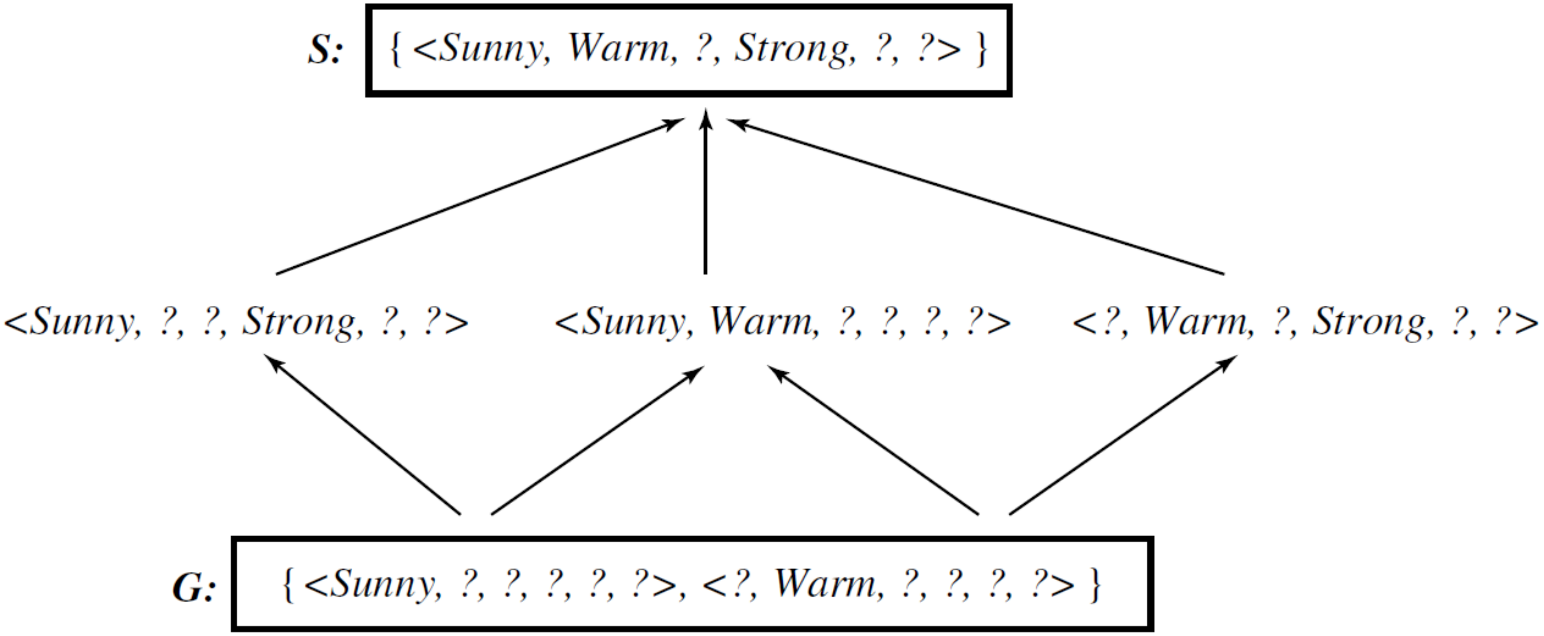
Running Example

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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Trace the Candidate-Elimination algorithm on this dataset

Solution


S: { <Sunny, Warm, ?, Strong, ?, ?> }
<Sunny, ?, ?, Strong, ?, ?>
<Sunny, Warm, ?, ?, ?, ?>
<?, Warm, ?, Strong, ?, ?>
G: { <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

CORRECT HYPOTHESIS?

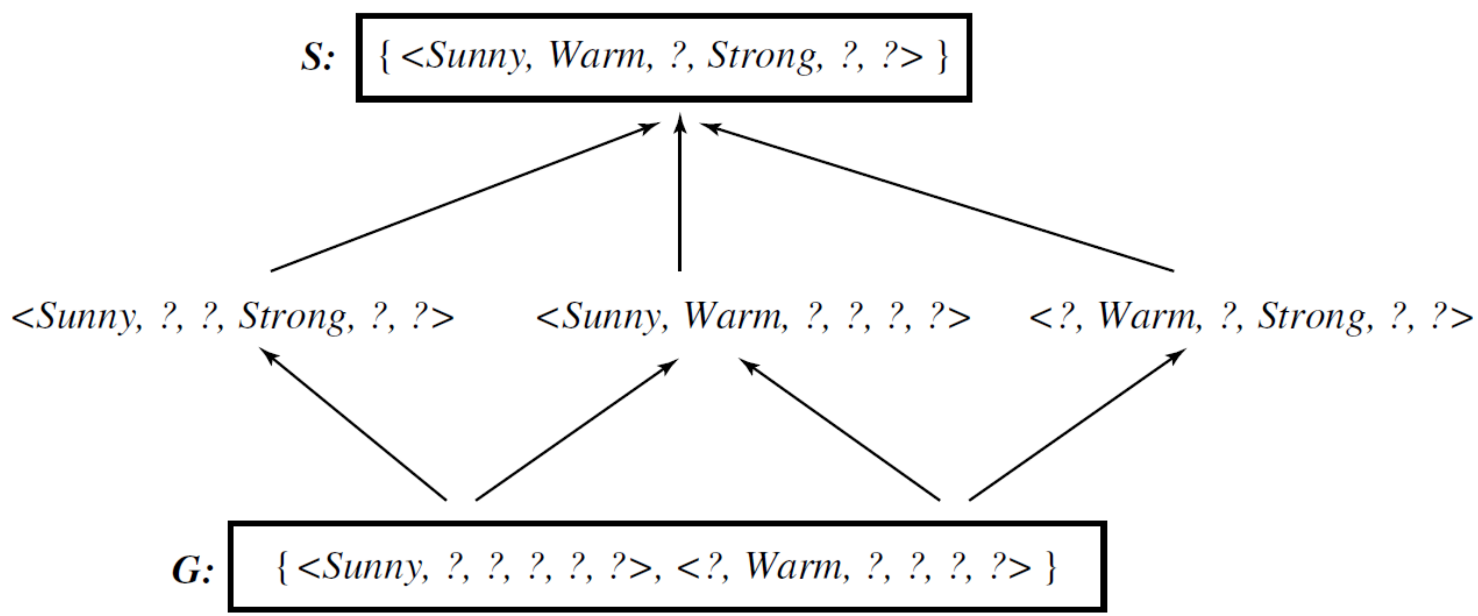
- If the training data does not contain errors, and, if the correct hypothesis is in *H*, then the candidate elimination algorithm will converge toward the correct hypothesis with each new training example
- If the training data contains errors, for example, let's say a positive example is annotated incorrectly as negative, then the correct hypothesis will surely be removed from the solution. Given enough data S and G might eventually become empty
- If the correct hypothesis is not in H, for example, if the correct hypothesis contains disjunctions whereas H contains only conjunctive hypotheses, given enough data, S and G might eventually become empty

Given S and G, how do we classify a new example?

- If the version space contain only one hypothesis, then classification of a new example is straightforward
- What if the version space contains multiple hypotheses, like the one that we just saw?

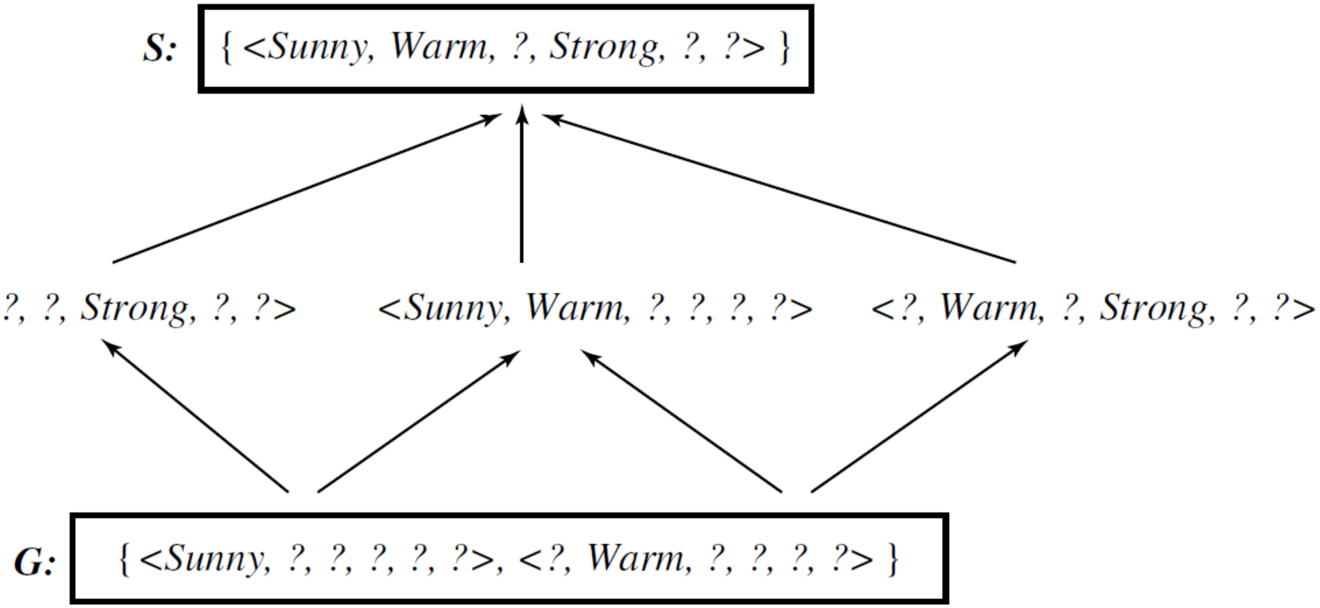
Classify the following

Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?



Active Learning

- Given the following version space, and if we give the algorithm the choice to choose the next example and ask for its label, what example should it ask about?



Inductive Bias

- We assumed that H is the conjunction of attributes. This is our inductive bias.
- What happens when the target concept is not in H?
- Can we avoid these problems by having a hypothesis space that has all possible hypotheses? That is, what if our hypothesis space is unbiased?
- First, how big is such a hypothesis space?
  - Given n Boolean attributes, there are 2^n possible examples
  - Each example can be a positive or negative example
  - Therefore, there are 2^{2^n} possible hypotheses!
- Second, how useful are such hypotheses?

Unbiased Learning

- H ≡ conjunctions, disjunctions, and negations
- Assume x_1, x_2, x_3 are positive and x_4 and x_5 are negative
- S is
  - $S = \{(x_1 \vee x_2 \vee x_3)\}$
- G is
  - $G = \{\neg(x_4 \vee x_5)\}$
- How do you classify a new/unseen example?

Biased vs Unbiased Learning

- In biased learning, we make assumptions about the hypothesis space
- In unbiased learning, no assumptions are made about the hypothesis space
- Purpose of concept learning: generalize to unseen data
- Unbiased learning simply memorizes the training data and it has no hope of generalizing to unseen data

ANOTHER EXAMPLE

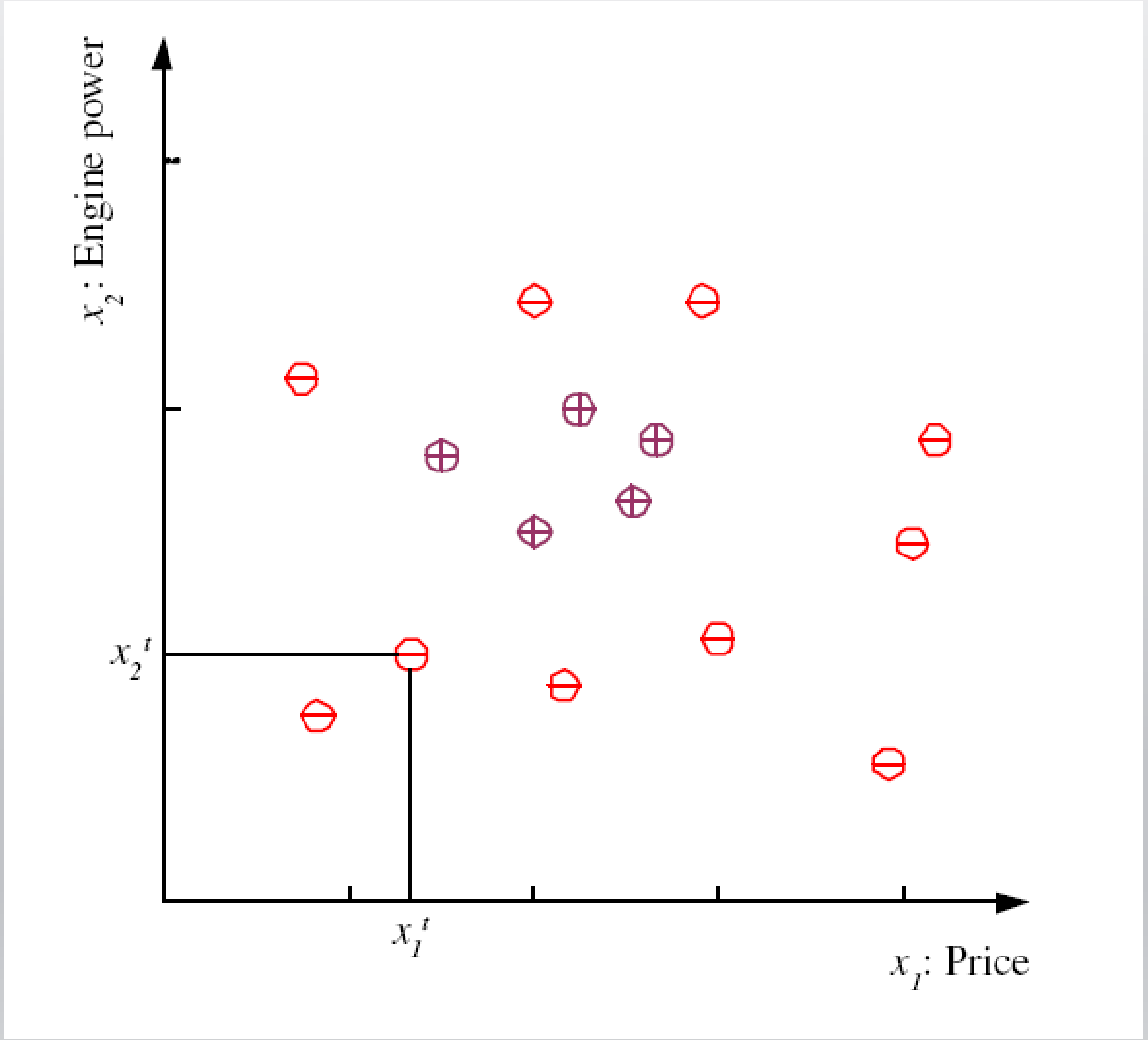
Learning a Class from Examples

- Class C of a "family car"
  - Prediction: Is car x a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Training set $\mathcal{X}$

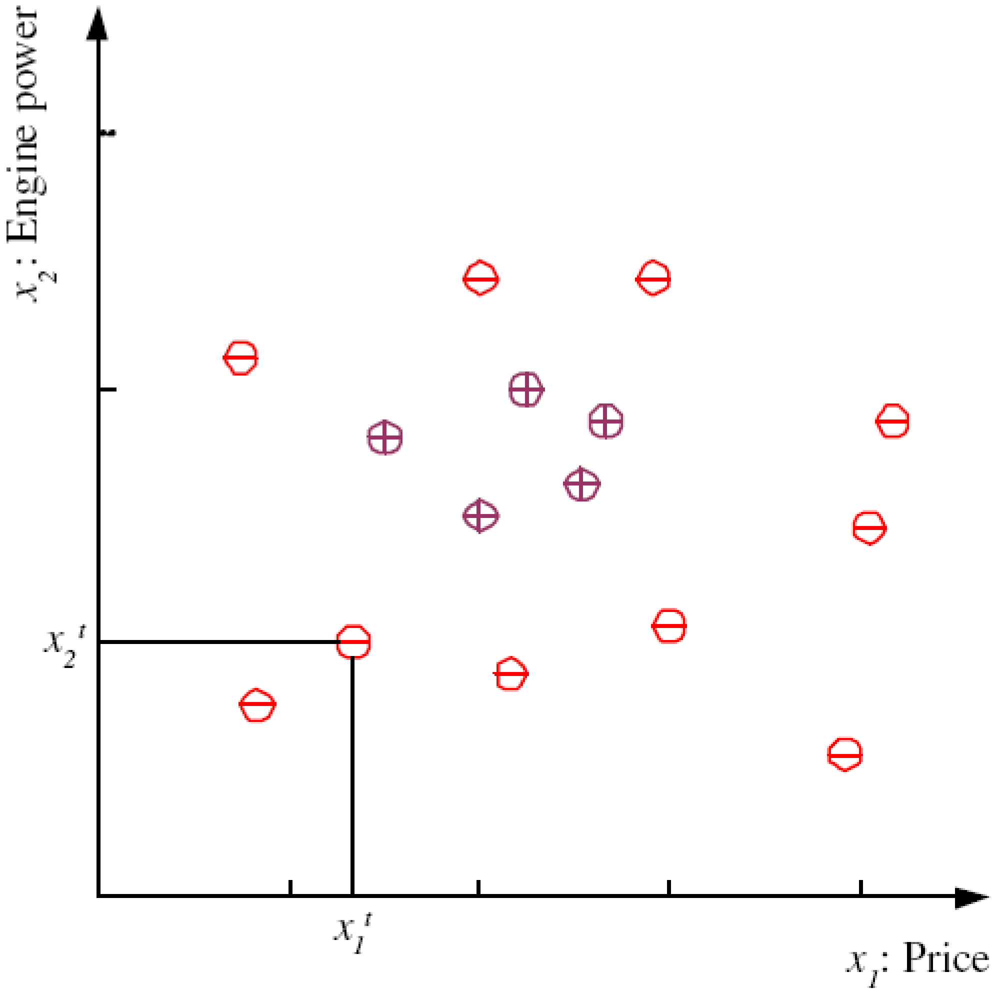


$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

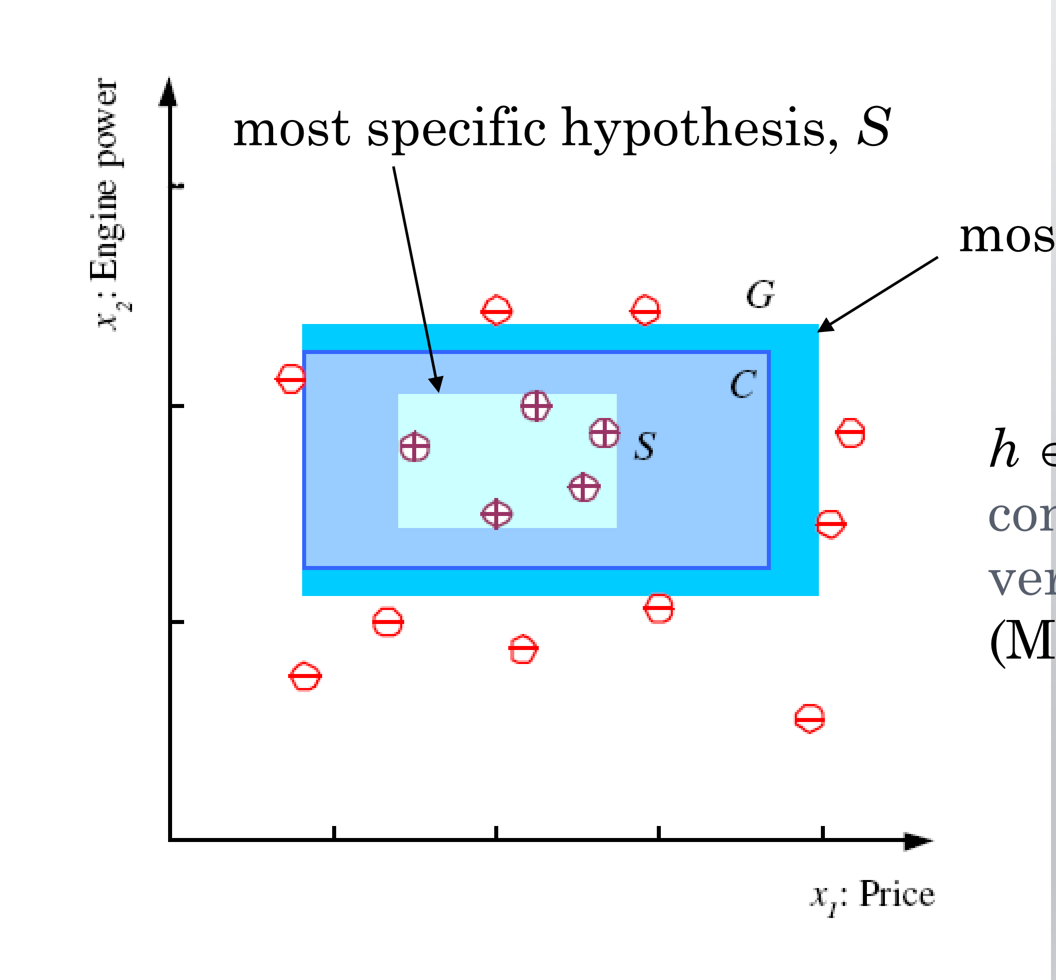
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

HYPOTHESIS SPACE



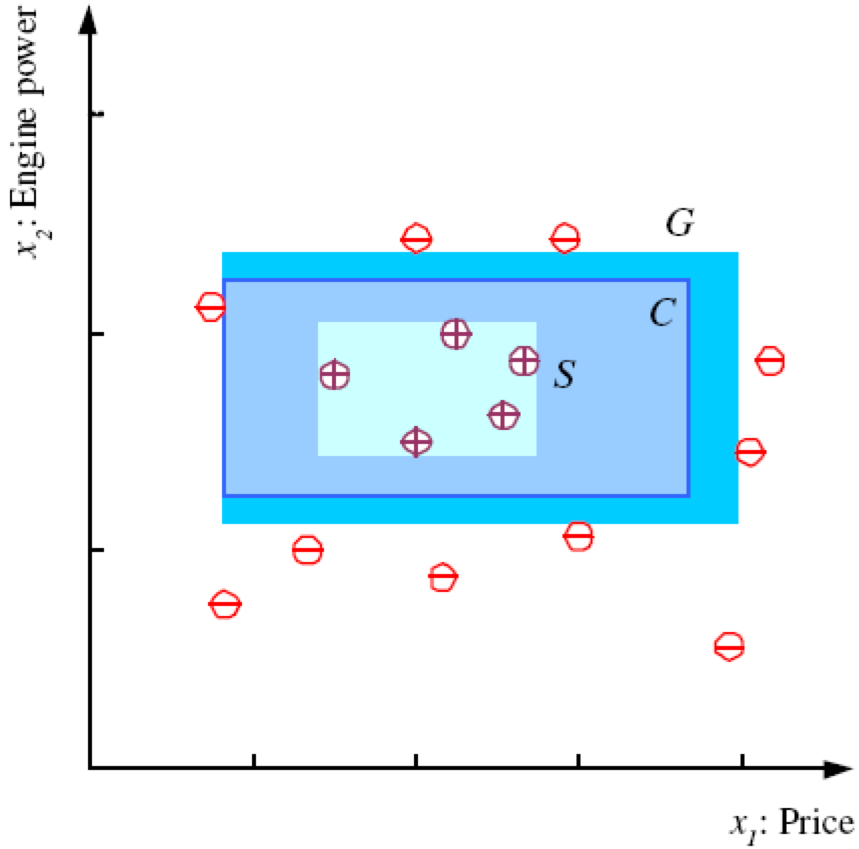
- Assume that the hypothesis space, H, consists of rectangles
- What would be S, G, and the version space?

S, G, AND THE VERSION SPACE



$h \in$ H, between S and G is consistent and make up the version space
(Mitchell, 1997)

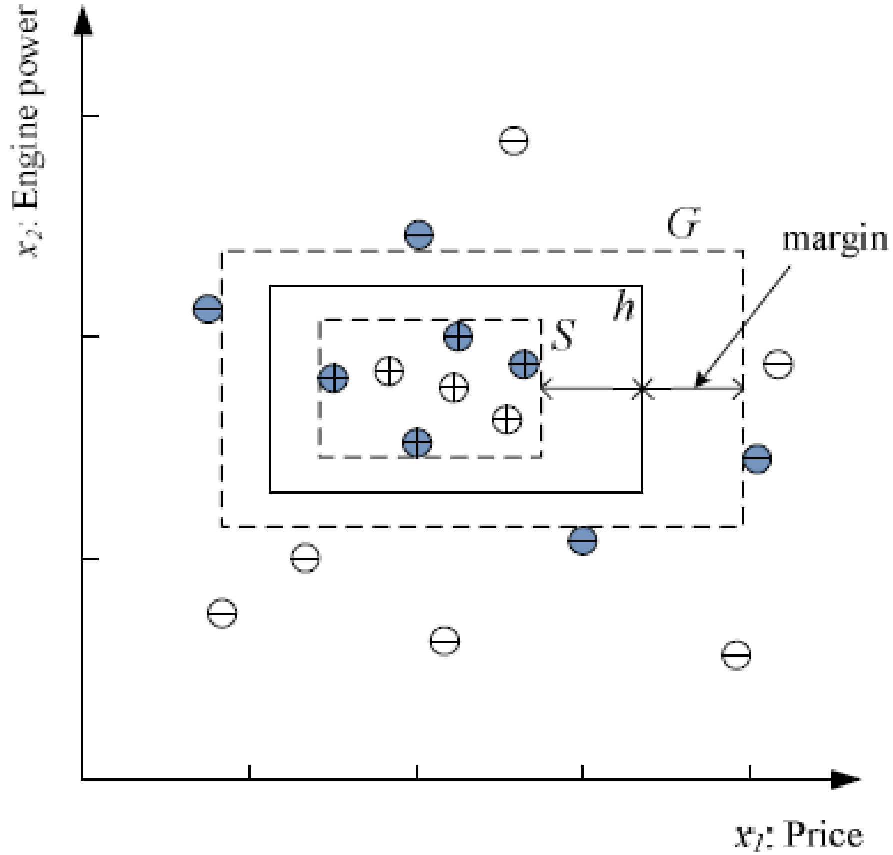
S, G, and the Version Space



- How would you classify a new example?
- In which regions would you have unanimous vote of all the hypotheses in the version space?
- In which regions, more than half would vote + and in which regions more than half would vote - ?

Margin

- Choose *h* with largest margin



Can you relate this to voting in the version space?

Exercise

- Try coming up with simple a concept learning problem
  - Define the task
    - Define the attributes
    - Define the target class / the correct hypothesis
  - Generate a few examples
  - Trace Find-S and Candidate-Elimination algorithms
  - Generate a few test examples
  - Classify the new test examples using S, G, and the full version space
  - Which object would active learning choose to label next?