Machine Learning

Chapter 10

Machine Learning

- What is learning?

Machine Learning

- Arthur Samuel (1959):

  "Field of study that gives computers the ability to learn without being explicitly programmed”.

- Tom Mitchell (1997):

  "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”.

Machine Learning

- How to construct programs that automatically improve with experience.

Machine Learning

- How to construct programs that automatically improve with experience.

- Learning problem:
  - Task T
  - Performance measure P
  - Training experience E

Machine Learning

- Chess game:
  - Task T: playing chess games
  - Performance measure P: percent of games won against opponents
  - Training experience E: playing practice games againts itself

Machine Learning

- Handwriting recognition:
  - Task T: recognizing and classifying handwritten words
  - Performance measure P: percent of words correctly classified
  - Training experience E: handwritten words with given classifications

Example

Experience

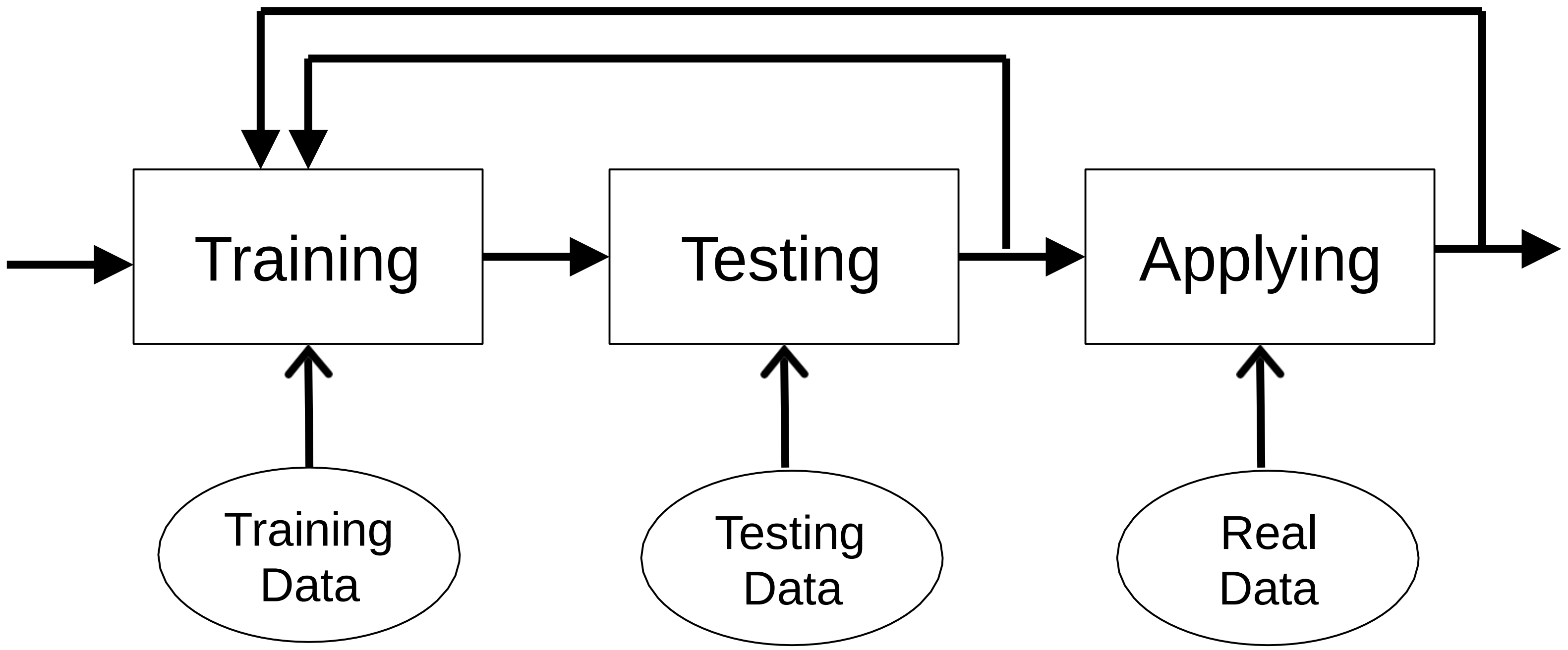
Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Low Weak

Prediction

5	Rainy	Cold	High	Strong	Warm	Change	?
6	Sunny	Warm	Normal	Strong	Warm	Same	?
7	Sunny	Warm	Low	Strong	Cool	Same	?

Machine Learning

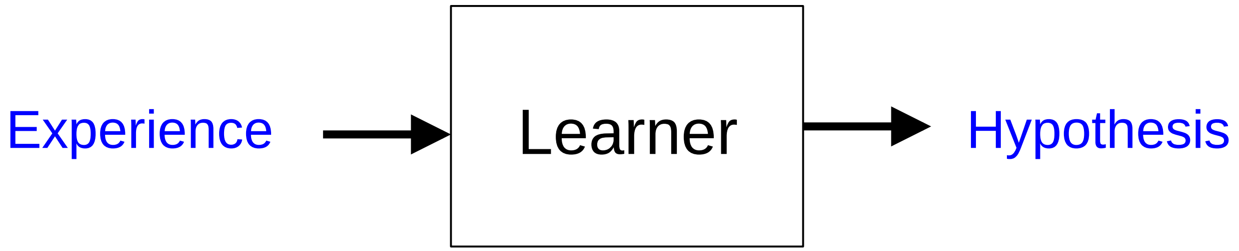


Machine Learning

- What is learning?

Machine Learning

- What is learning?



Machine Learning

- Learning is an (endless) generalization or induction process.

Machine Learning

- Supervised learning: the learner (learning algorithm) are trained on labelled examples, i.e., input where the desired output is known.

- Unsupervised learning: the learner operates on unlabelled examples, i.e., input where the desired output is unknown.

Concept Learning

- Inferring a boolean-valued function from training examples of its input (instances) and output (classifications).

Concept Learning

- Learning problem:
  - Target concept: a subset of the set of instances X

    $c: X \rightarrow \{0, 1\}$
  - Target function:

    $Sky \times AirTemp \times Humidity \times Wind \times Water \times Forecast \rightarrow \{0, 1\}$
  - Hypothesis:

    Characteristics of all instances of the concept to be learned

    $\equiv$ Constraints on instance attributes

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

Concept Learning

- Satisfaction:

  h(x) = 1 iff x satisfies all the constraints of h

  h(x) = 0 otherwsie

- Consistency:

  h(x) = c(x) for every instance x of the training examples

- Correctness:

  h(x) = c(x) for every instance x of X

Concept Learning

- How to represent a hypothesis?

Concept Learning

- Hypothesis representation (constraints on instance attributes):

  <Sky, AirTemp, Humidity, Wind, Water, Forecast>

  - ?: any value is acceptable
  - single required value
  - ∅: no value is acceptable

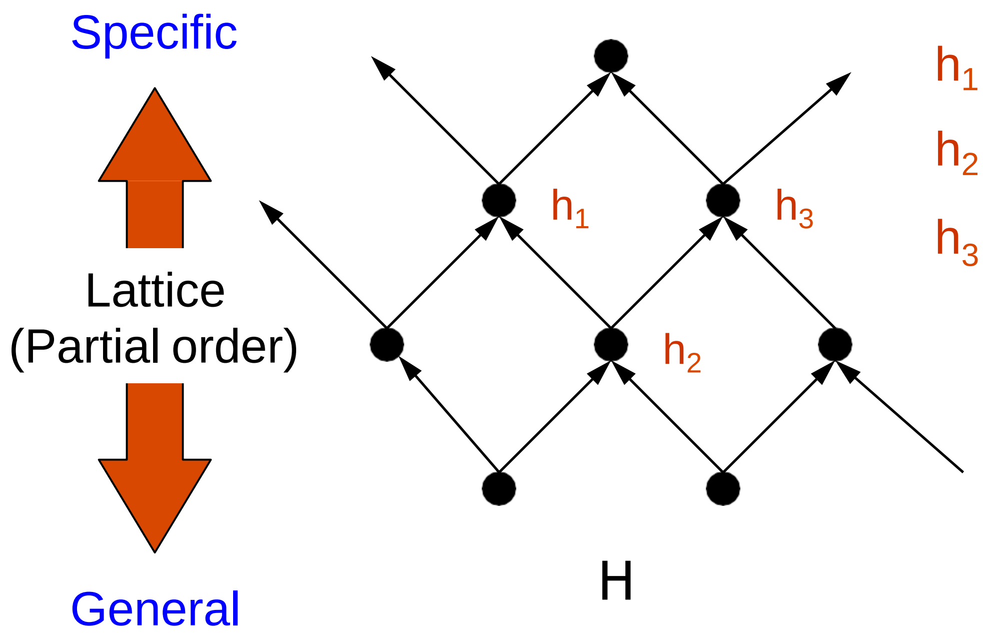
- Example:

  h1 = <Sunny, ?, ?, Strong, ? , ?>

Concept Learning

- General-to-specific ordering of hypotheses:

  $h_j \geq_g h_k$ iff $\forall x \in X: h_k(x) = 1 \Rightarrow h_j(x) = 1$



h_1 = <Sunny, ?, ?, Strong, ? , ?>

h_2 = <Sunny, ?, ?, ? , ? , ?>

h_3 = <Sunny, ?, ?, ? , Cool, ?>

Concept Learning

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 is a hypothesis that is consistent with the training examples?

h = < _ , _ , _ , _ , _ , _ >

Concept Learning

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 is the most specific hypothesis that is consistent with the training examples?

h = < _ , _ , _ , _ , _ , _ >

FIND-S

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

h = < ∅ , ∅ , ∅ , ∅ , ∅ , ∅ >

h = <Sunny, Warm, Normal, Strong, Warm, Same>

h = <Sunny, Warm, ? , Strong, Warm, Same>

h = <Sunny, Warm, ? , Strong, ? , ? >

FIND-S

- Initialize h to the most specific hypothesis in H:
- For each positive training instance x:

  For each attribute constraint a_i in h:

  If the constraint is not satisfied by x

  Then replace a_i by the next more general constraint satisfied by x
- Output hypothesis h

FIND-S

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

h = <Sunny, Warm, ? , Strong, ? , ? >

Prediction

5	Rainy	Cold	High	Strong	Warm	Change	No
6	Sunny	Warm	Normal	Strong	Warm	Same	Yes
7	Sunny	Warm	Low	Strong	Cool	Same	Yes

FIND-S

- The output hypothesis is the most specific one that satisfies all positive training examples.

FIND-S

- The result is consistent with the positive training examples.

FIND-S

- Is the result is consistent with the negative training examples?

FIND-S

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
5	**Sunny**	**Warm**	**Normal**	**Strong**	**Cool**	**Change**	**No**

h = <Sunny, Warm, ? , Strong, ? , ? >

FIND-S

- The result is consistent with the negative training examples if the target concept is contained in H (and the training examples are correct).

FIND-S

- The result is consistent with the negative training examples if the target concept is contained in H (and the training examples are correct).

- Sizes of the space:
  - Size of the instance space: $|X| = 3.2.2.2.2.2 = 96$
  - Size of the concept space $C = 2^{|X|} = 2^{96}$
  - Size of the hypothesis space $H = (4.3.3.3.3.3) + 1 = 973 << 2^{96}$

$\Rightarrow$ The target concept (in C) may not be contained in H.

FIND-S

- Questions:
  - Has the learner converged to the target concept, as there can be several consistent hypotheses (with both positive and negative training examples)?
  - Why the most specific hypothesis is preferred?
  - What if there are several maximally specific consistent hypotheses?
  - What if the training examples are not correct?

List-then-Eliminate Algorithm

- Version space: a set of all hypotheses that are consistent with the training examples.
- Algorithm:
  - Initial version space = set containing every hypothesis in H
  - For each training example $<x, c(x)>$, remove from the version space any hypothesis h for which $h(x) \neq c(x)$
  - Output the hypotheses in the version space

List-then-Eliminate Algorithm

- Requires an exhaustive enumeration of all hypotheses in H

Compact Representation of Version Space

- G (the generic boundary): set of the most generic hypotheses of H consistent with the training data D:

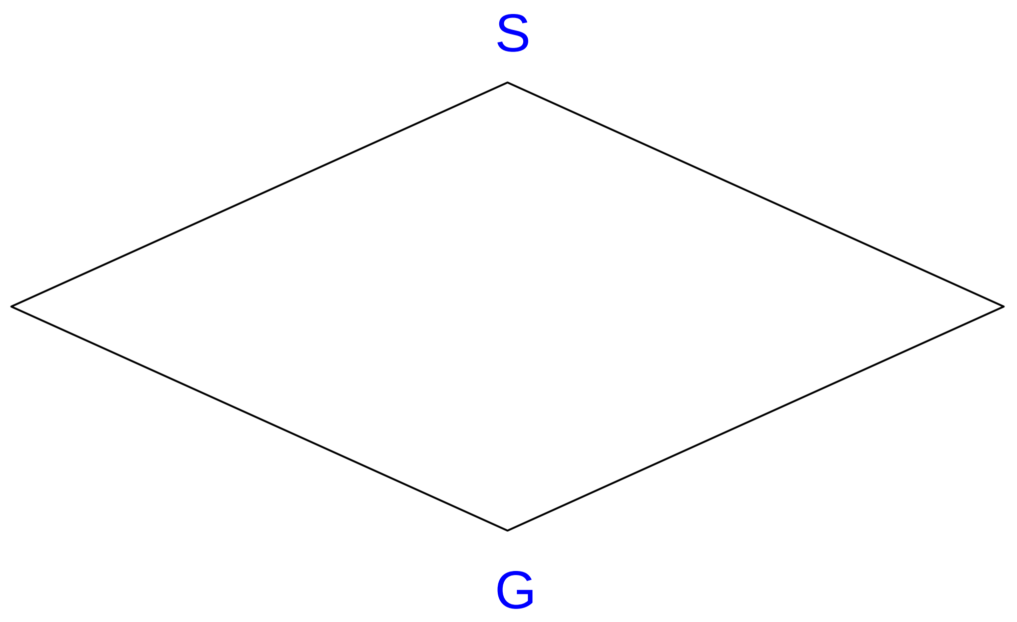
  $G = \{g \in H \mid \text{consistent}(g, D) \wedge \neg\exists g' \in H: g' >_g g \wedge \text{consistent}(g', D)\}$

- S (the specific boundary): set of the most specific hypotheses of H consistent with the training data D:

  $S = \{s \in H \mid \text{consistent}(s, D) \wedge \neg\exists s' \in H: s >_g s' \wedge \text{consistent}(s', D)\}$

Compact Representation of Version Space

- Version space = <G, S> = $\{h \in H \mid \exists g \in G\ \exists s \in S: g \geq_g h \geq_g s\}$



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

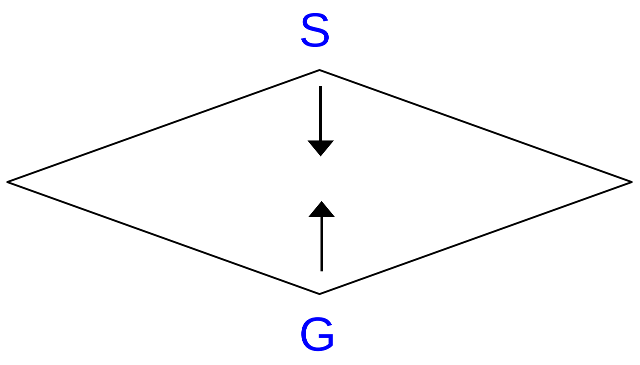
S_0 = {<∅, ∅, ∅, ∅, ∅, ∅>}
G_0 = {<?, ?, ?, ?, ?, ?>}

S_1 = {<Sunny, Warm, Normal, Strong, Warm, Same>}
G_1 = {<?, ?, ?, ?, ?, ?>}

S_2 = {<Sunny, Warm, ?, Strong, Warm, Same>}
G_2 = {<?, ?, ?, ?, ?, ?>}

S_3 = {<Sunny, Warm, ?, Strong, Warm, Same>}
G_3 = {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

S_4 = {<Sunny, Warm, ?, Strong, ?, ?>}
G_4 = {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>}



Candidate-Elimination Algorithm

$$\begin{tikzcd}
 & S4 = \{\langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ? \rangle\} & \\
\langle \text{Sunny}, ?, ?, \text{Strong}, ?, ? \rangle \arrow[ur] & \langle \text{Sunny}, \text{Warm}, ?, ?, ?, ? \rangle \arrow[u] & \langle ?, \text{Warm}, ?, \text{Strong}, ?, ? \rangle \arrow[ul] \\
G4 = \{\langle \text{Sunny}, ?, ?, ?, ?, ? \rangle, \arrow[u] \arrow[ur] & & \langle ?, \text{Warm}, ?, ?, ?, ? \rangle\} \arrow[ul] \arrow[u]
\end{tikzcd}$$

Candidate-Elimination Algorithm

- Initialize G to the set of maximally general hypotheses in H
- Initialize S to the set of maximally specific hypotheses in H

Candidate-Elimination Algorithm

- For each positive example d:
  - Remove from G any hypothesis inconsistent with d
  - For each s in S that is inconsistent with d:

    Remove s from S

    Add to S all least generalizations h of s, such that h is consistent with d and some hypothesis in G is more general than h

    Remove from S any hypothesis that is more general than another hypothesis in S

Candidate-Elimination Algorithm

- For each negative example d:
  - Remove from S any hypothesis inconsistent with d
  - For each g in G that is inconsistent with d:

    Remove g from G

    Add to G all least specializations h of g, such that h is consistent with d and some hypothesis in S is more specific than h

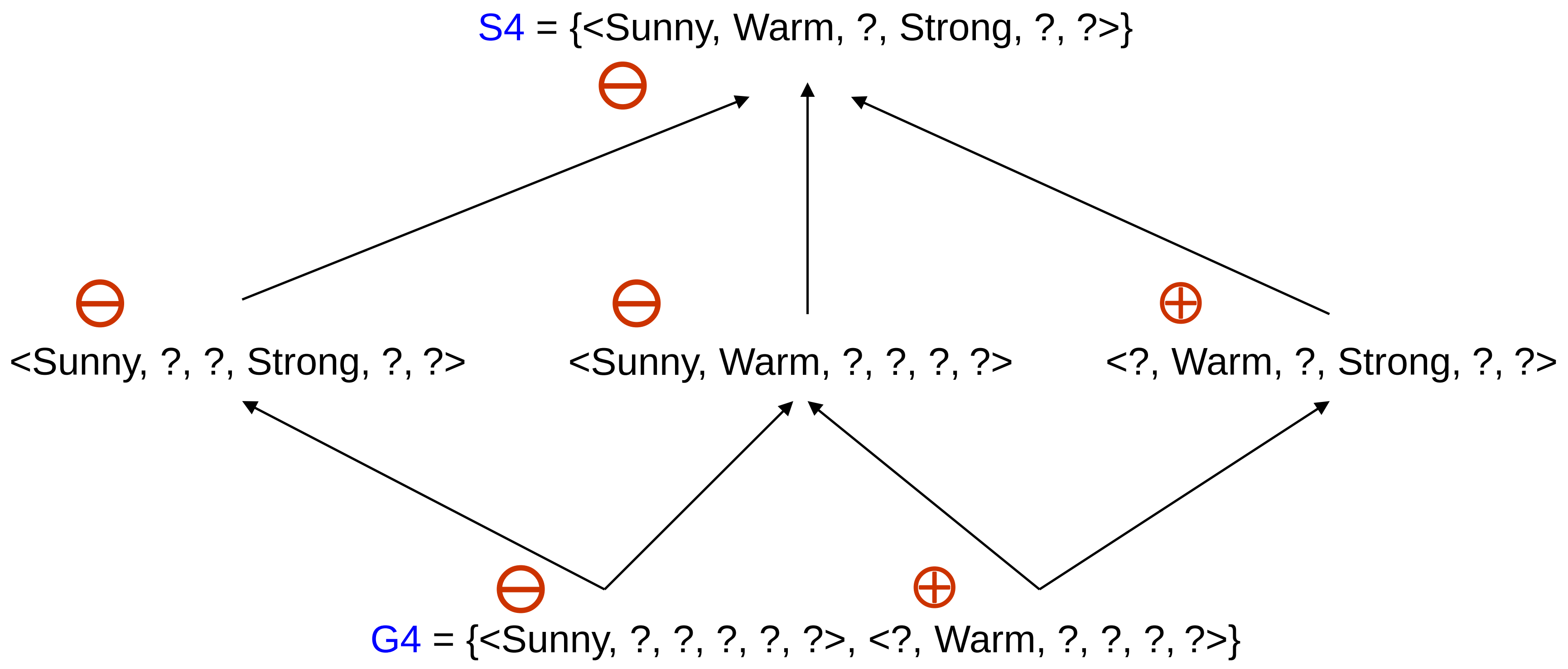
    Remove from G any hypothesis that is more specific than another hypothesis in G

Candidate-Elimination Algorithm

- The version space will converge toward the correct target concepts if:
  - H contains the correct target concept
  - There are no errors in the training examples
- A training instance to be requested next should discriminate among the alternative hypotheses in the current version space:

Candidate-Elimination Algorithm

- Partially learned concept can be used to classify new instances using the majority rule.



5	Rainy	Warm	High	Strong	Cool	Same	?

Inductive Bias

- Size of the instance space: $|X| = 3.2.2.2.2.2 = 96$
- Number of possible concepts = $2^{|X|} = 2^{96}$
- Size of $H = (4.3.3.3.3.3) + 1 = 973 << 2^{96}$

Inductive Bias

- Size of the instance space: $|X|$ = 3.2.2.2.2.2 = 96
- Number of possible concepts = $2^{|X|} = 2^{96}$
- Size of H = (4.3.3.3.3.3) + 1 = 973 << 2^{96}

$\Rightarrow$ a biased hypothesis space

Inductive Bias

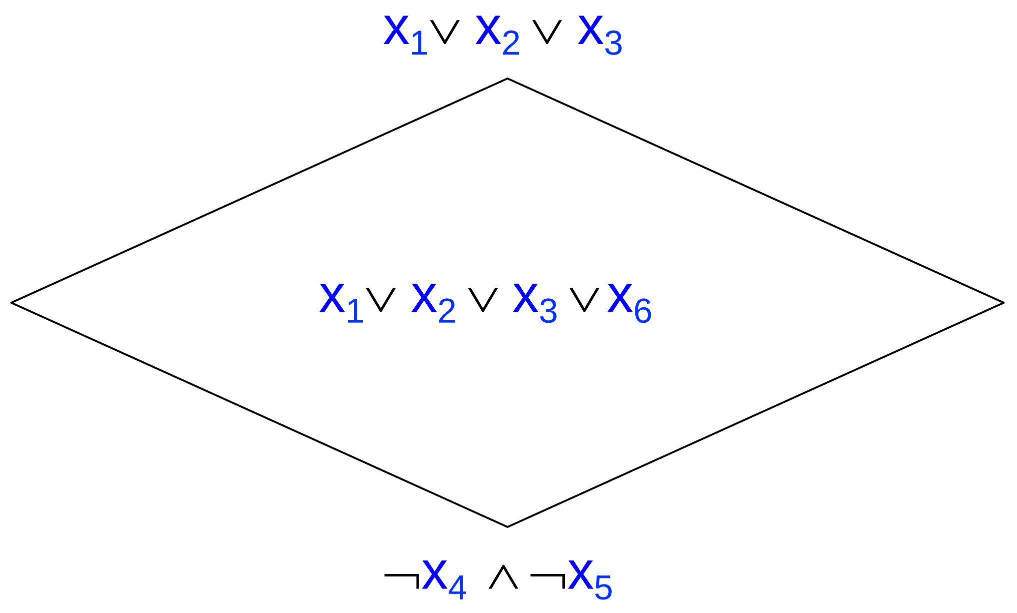
- An unbiased hypothesis space H' that can represent every subset of the instance space X: Propositional logic sentences

- Positive examples: x_1, x_2, x_3
  Negative examples: x_4, x_5

$$h(x) \equiv (x = x_1) \vee (x = x_2) \vee (x = x_3) \equiv x_1 \vee x_2 \vee x_3$$

$$h'(x) \equiv (x \neq x_4) \wedge (x \neq x_5) \equiv \neg x_4 \wedge \neg x_5$$

Inductive Bias



Any new instance x is classified positive by half of the version space, and negative by the other half

$\Rightarrow$ not classifiable

Inductive Bias

Example	Day	Actor	Price	EasyTicket
1	Mon	Famous	Expensive	Yes
2	Sat	Famous	Moderate	No
3	Sun	Infamous	Cheap	No
4	Wed	Infamous	Moderate	Yes
5	Sun	Famous	Expensive	No
6	Thu	Infamous	Cheap	Yes
7	Tue	Famous	Expensive	Yes
8	Sat	Famous	Cheap	No

9	Wed	Famous	Cheap	?
10	Sat	Infamous	Expensive	?

Inductive Bias

Example	Quality	Price	Buy
1	Good	Low	Yes
2	Bad	High	No

3	Good	High	?
4	Bad	Low	?

Inductive Bias

- A learner that makes no prior assumptions regarding the identity of the target concept cannot classify any unseen instances.

Homework

Exercises

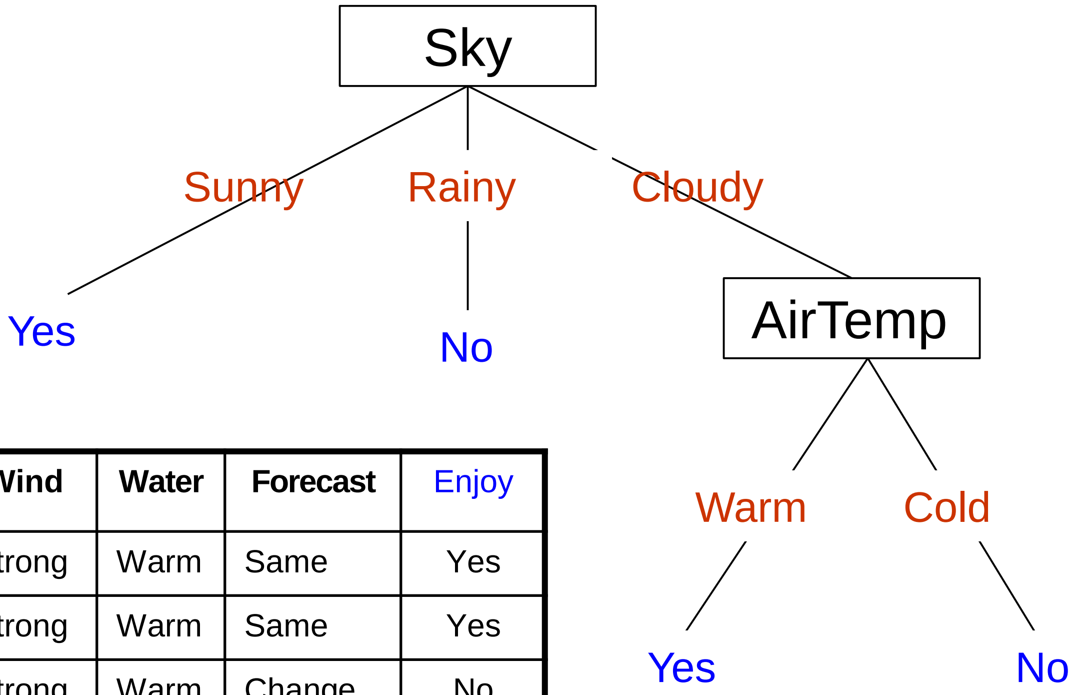
2-1 $\rightarrow$ 2.5 (Chapter 2, ML textbook)

Chapter 9 of the Vietnamese Textbook

Decision Trees

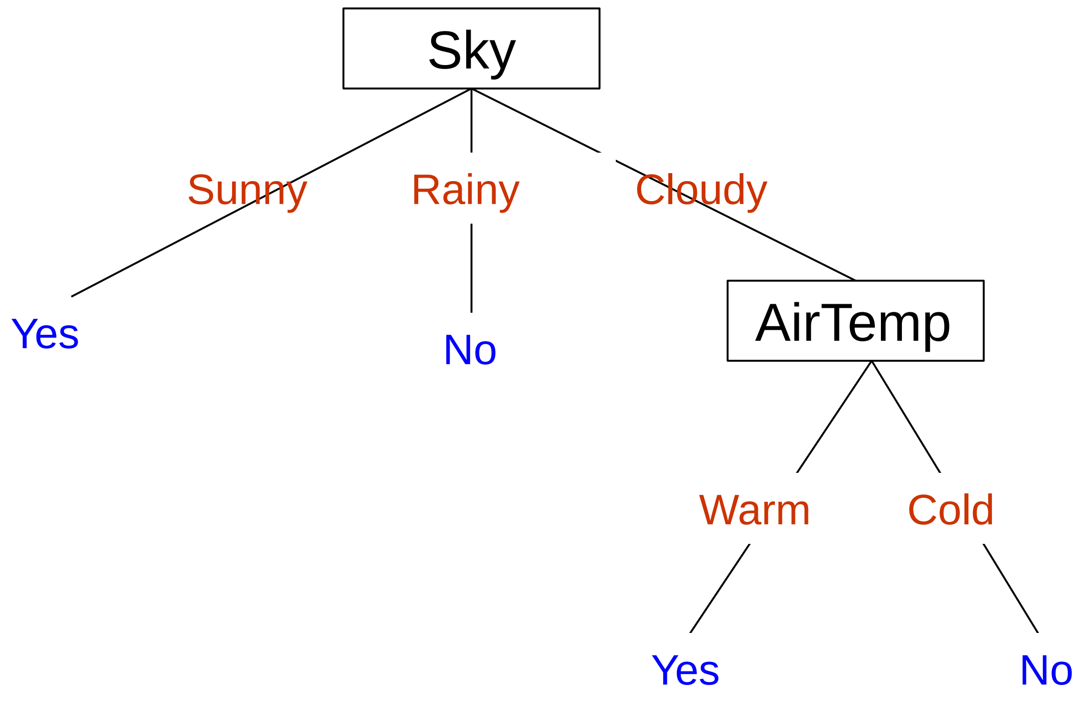
Example	**Sky**	**AirTemp**	**Humidity**	**Wind**	**Water**	**Forecast**	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes
5	Cloudy	Warm	High	Weak	Cool	Same	Yes
6	Cloudy	Cold	High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

Decision Trees



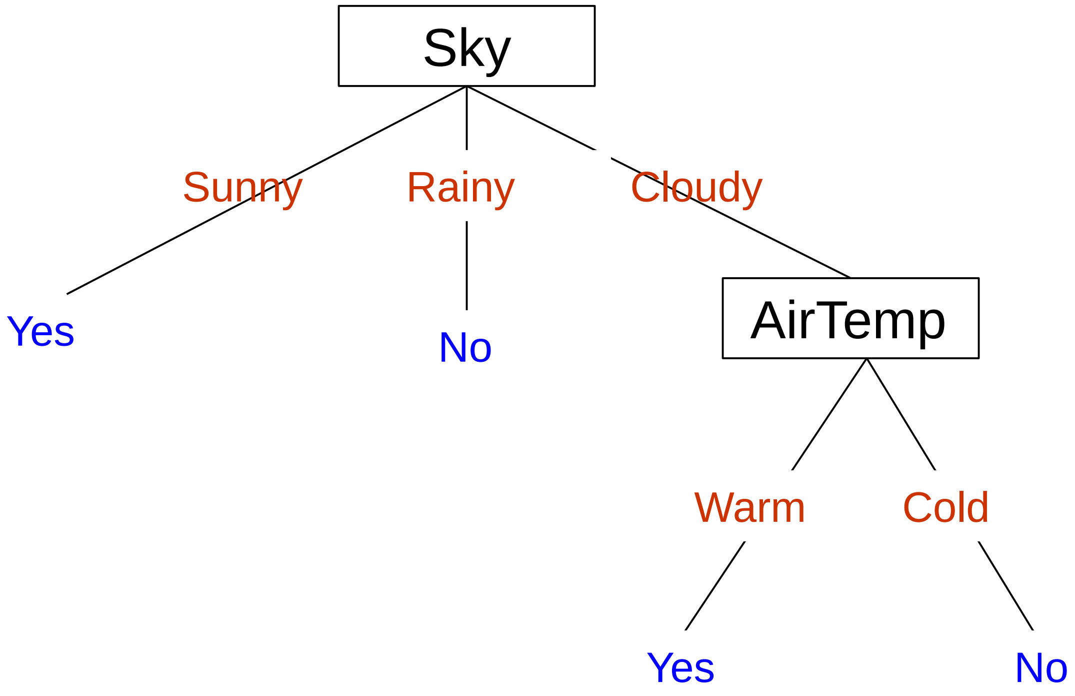
No.	Sky	AirTemp	Humidity	Wind	Water	Forecast	Enjoy
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes
5	Cloudy	Warm	High	Weak	Cool	Same	Yes
6	Cloudy	Cold	High	Weak	Cool	Same	No

Decision Trees



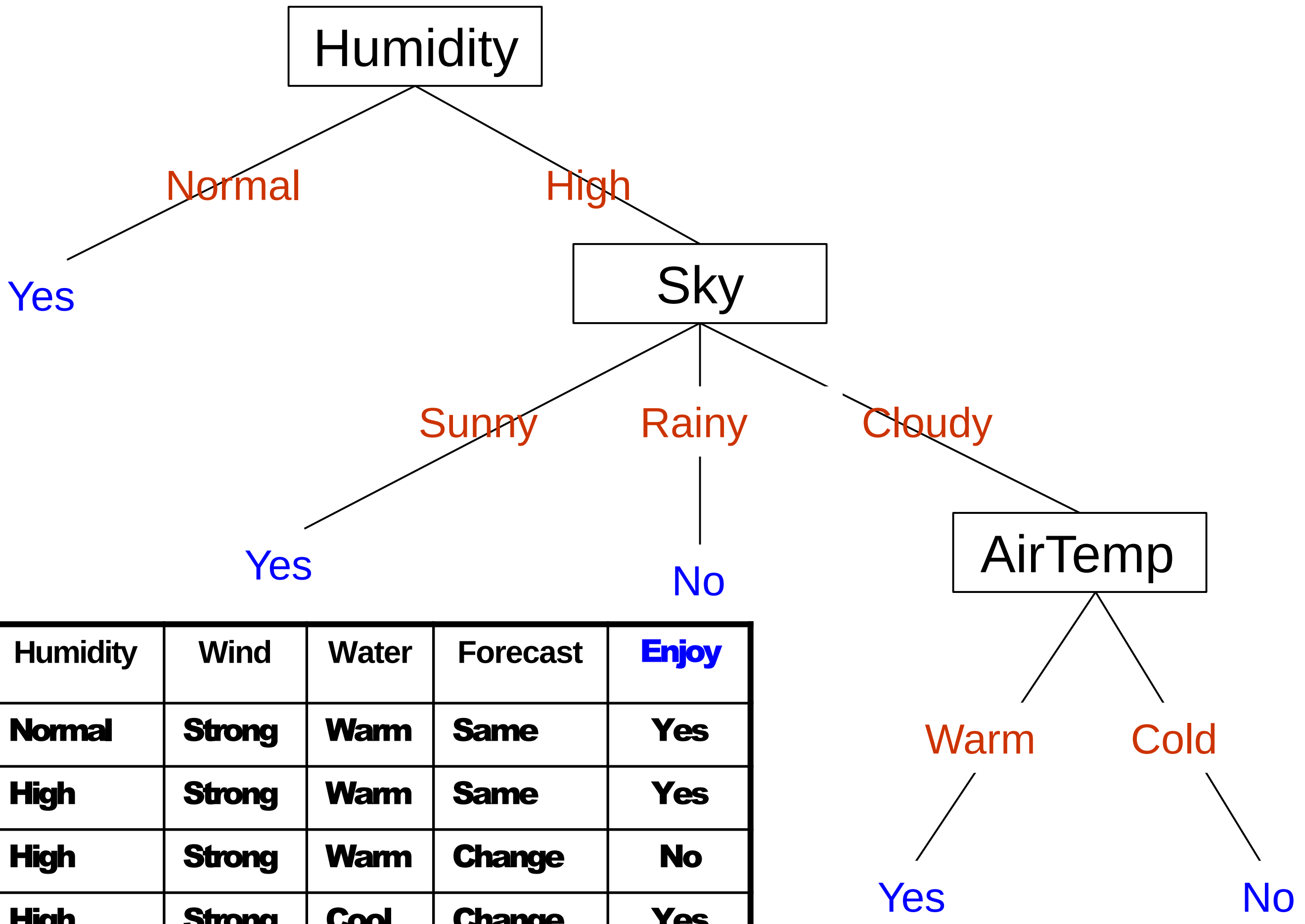
$(Sky = Sunny) \vee (Sky = Cloudy \wedge AirTemp = Warm)$

Decision Trees



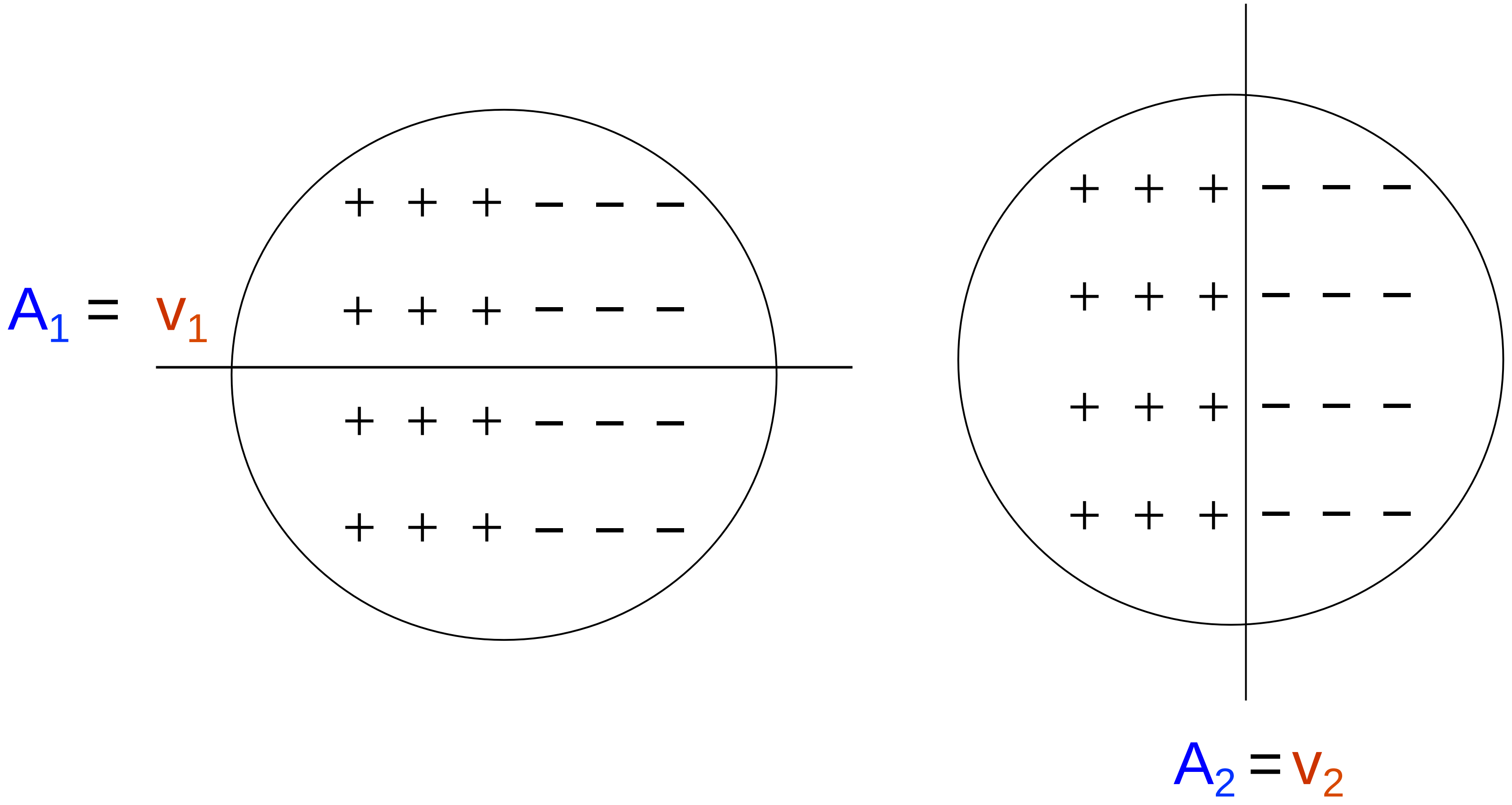
7	Rainy	Warm	Normal	Weak	Cool	Same	?
8	Cloudy	Warm	High	Strong	Cool	Change	?

Decision Trees



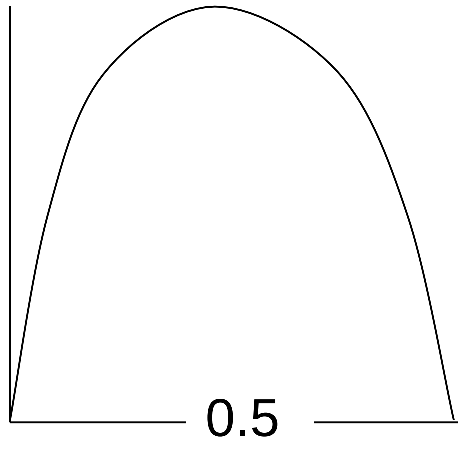
No.	Sky	AirTemp	Humidity	Wind	Water	Forecast	Enjoy
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes
5	Cloudy	Warm	High	Weak	Cool	Same	Yes
6	Cloudy	Cold	High	Weak	Cool	Same	No

Decision Trees



Homogenity of Examples

- Entropy(S) = $- p_{+}\log_{2}p_{+} - p_{-}\log_{2}p_{-}$

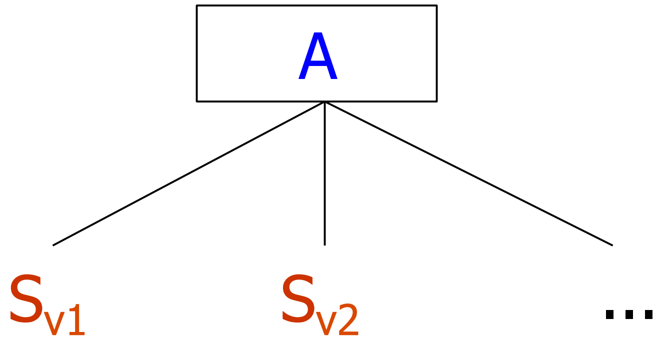


Homogenity of Examples

- Entropy(S) = $\sum_{i=1,c} - p_i \log_2 p_i$ impurity measure

Information Gain

- $\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} (|S_v|/|S|).\text{Entropy}(S_v)$



Example

- $\text{Entropy}(S) = -\ p_+\log_2 p_+ - p_-\log_2 p_- = -\ (4/6)\log_2(4/6) - (2/6)\log_2(2/6)$

  $= 0.389 + 0.528 = 0.917$

- Gain(S, Sky)

  $= \text{Entropy}(S) - \sum_{v\in\{\text{Sunny, Rainy, Cloudy}\}}(|S_v|/|S|)\text{Entropy}(S_v)$

  $= \text{Entropy}(S) - [(3/6).\text{Entropy}(S_{\text{Sunny}}) + (1/6).\text{Entropy}(S_{\text{Rainy}}) +$

  $(2/6).\text{Entropy}(S_{\text{Cloudy}})]$

  $= \text{Entropy}(S) - (2/6).\text{Entropy}(S_{\text{Cloudy}})$

  $= \text{Entropy}(S) - (2/6)[-\ (1/2)\log_2(1/2) - (1/2)\log_2(1/2)]$

  $= 0.917 - 0.333 = 0.584$

Example

- $\text{Entropy}(S) = -\ p_+\log_2 p_+ - p_-\log_2 p_- = -\ (4/6)\log_2(4/6) - (2/6)\log_2(2/6)$

  $= 0.389 + 0.528 = 0.917$

- Gain(S, Water)

  $= \text{Entropy}(S) - \sum_{v\in\{\text{Warm, Cool}\}}(|S_v|/|S|)\text{Entropy}(S_v)$

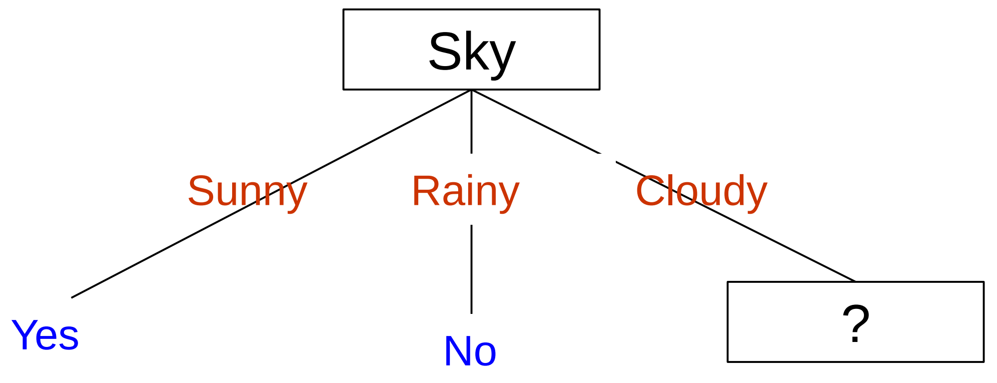
  $= \text{Entropy}(S) - [(3/6).\text{Entropy}(S_{\text{Warm}}) + (3/6).\text{Entropy}(S_{\text{Cool}})]$

  $= \text{Entropy}(S) - (3/6).2.[-\ (2/3)\log_2(2/3) - (1/3)\log_2(1/3)]$

  $= \text{Entropy}(S) -\ 0.389 - 0.528$

  $= 0$

Example



- Gain(S_{Cloudy}, AirTemp)

  $= \text{Entropy}(S_{Cloudy}) - \sum_{v\in\{\text{Warm, Cold}\}}(|S_v|/|S|)\text{Entropy}(S_v)$

  $= 1$

- Gain(S_{Cloudy}, Humidity)

  $= \text{Entropy}(S_{Cloudy}) - \sum_{v\in\{\text{Normal, High}\}}(|S_v|/|S|)\text{Entropy}(S_v)$

  $= 0$

Inductive Bias

- Hypothesis space: complete!

Inductive Bias

- Hypothesis space: complete!
- Shorter trees are preferred over larger trees
- Prefer the simplest hypothesis that fits the data

Inductive Bias

- Decision Tree algorithm: searches incompletely thru a complete hypothesis space.

  $\Rightarrow$ Preference bias

- Cadidate-Elimination searches completely thru an incomplete hypothesis space.

  $\Rightarrow$ Restriction bias

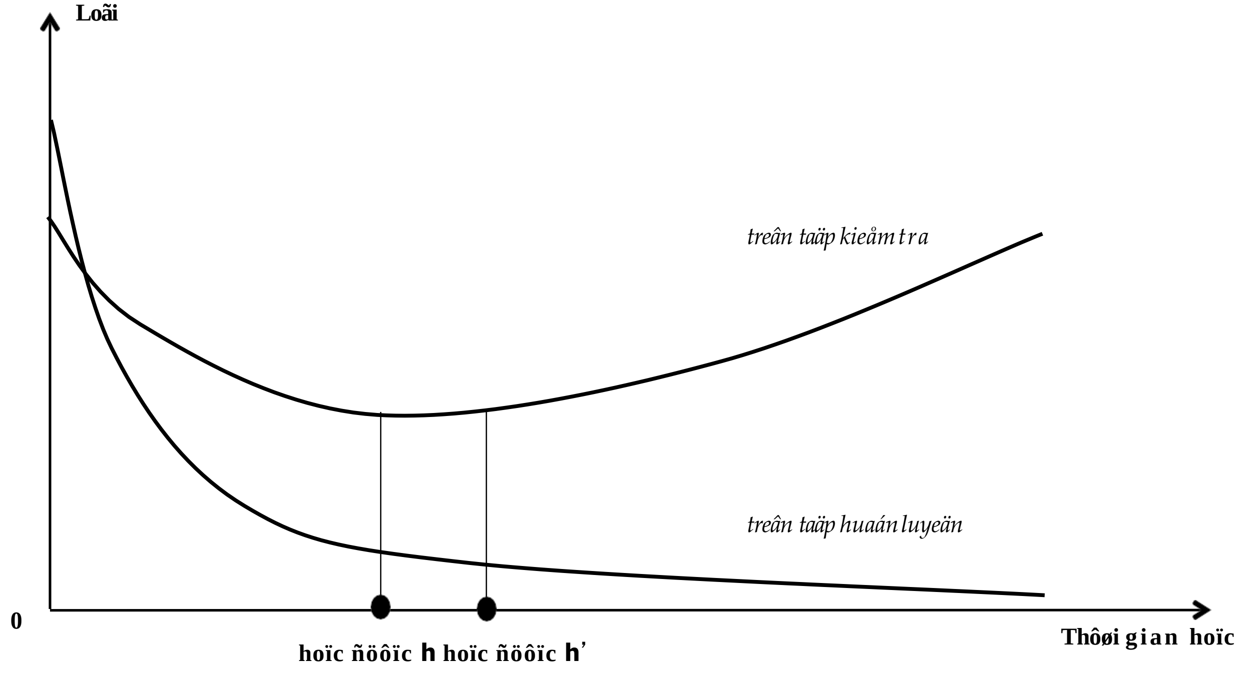
Overfitting

- $h \in H$ is said to overfit the training data if there exists $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

Overfitting

- $h \in H$ is said to overfit the training data if there exists $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances:
  - There is noise in the data
  - The number of training examples is too small to produce a representative sample of the target concept

Overfitting



Homework

Exercises 3-1$\rightarrow$3.4 (Chapter 3, ML textbook)