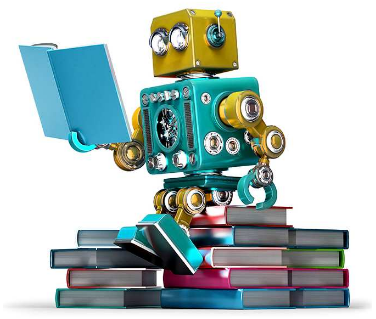
MACHINE LEARNING



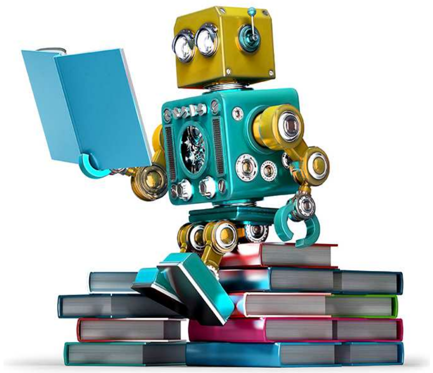
MODULE-I
INTRODUCTION TO ML,

BY
HARIVINOD N
VIVEKANANDA COLLEGE OF ENGINEERING TECHNOLOGY, PUTTUR

Module 1- Outline

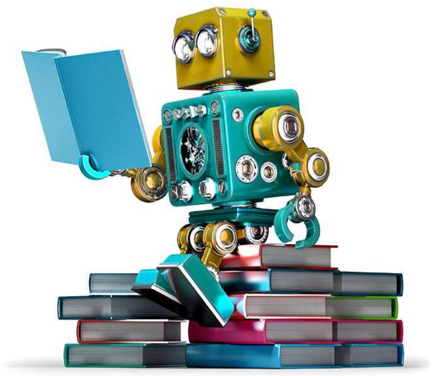
MACHINE LEARNING

- **Chapter 1: Introduction**
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Chapter 2: Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Module 1- Outline

- Chapter 1: Introduction
  - **Well posed learning problems**
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Chapter 2: Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



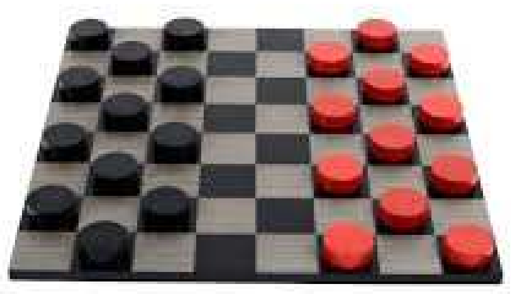
Well-Posed Learning Problems

- Learning is broadly defined as any computer program that improves its performance at some task through experience.
- **Definition: Machine Learning**

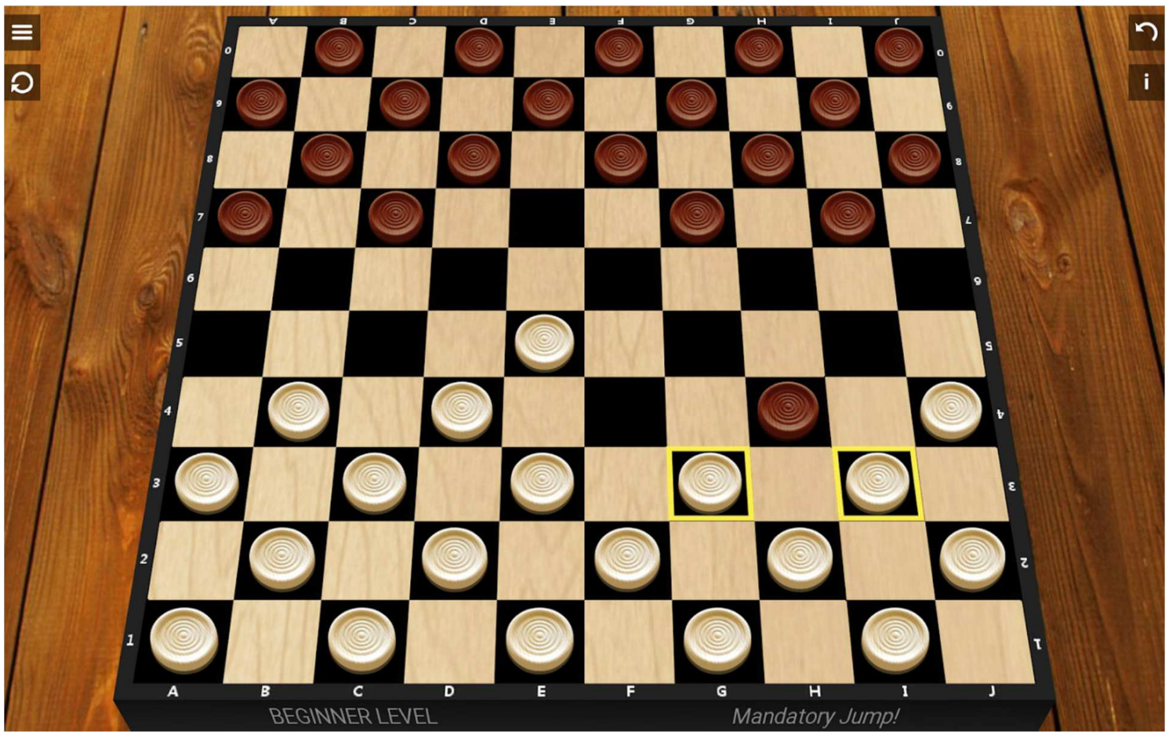
  A computer program is said to **learn** from experience *E* with respect to some class of tasks T and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

Ex1: A checkers learning problem

- Task T - Playing checkers
- Performance Measure P - Percentage of games won against opponent
- Training Experience E - Playing practice games against itself

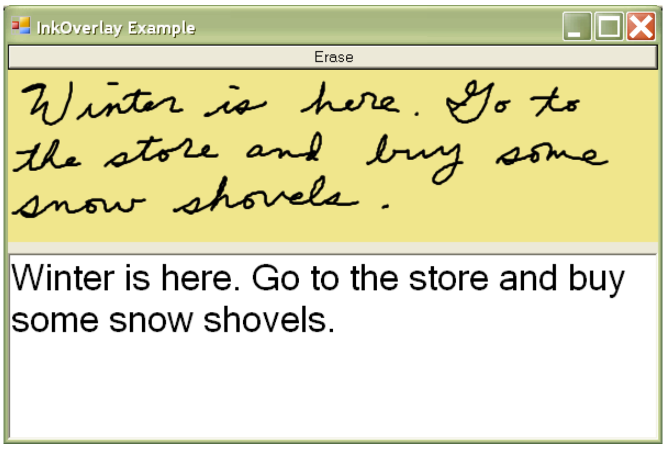


Checkers Board



Ex2: A handwriting recognition learning problem

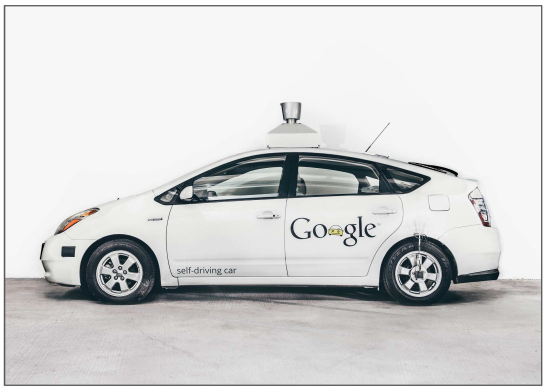
- Task T: recognizing and classifying handwritten words within images
- Performance measure P: percent of words correctly classified
- Training experience E: a database of handwritten words with given classifications



Winter is here. Go to the store and buy some snow shovels.

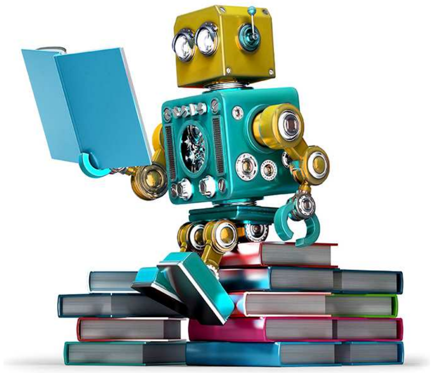
Ex3: A robot driving learning problem

- T: driving on public 4-lane highways using vision sensors
- P: average distance traveled before an error (as judged by human overseer)
- E: a sequence of images and steering commands recorded by observing a human driver



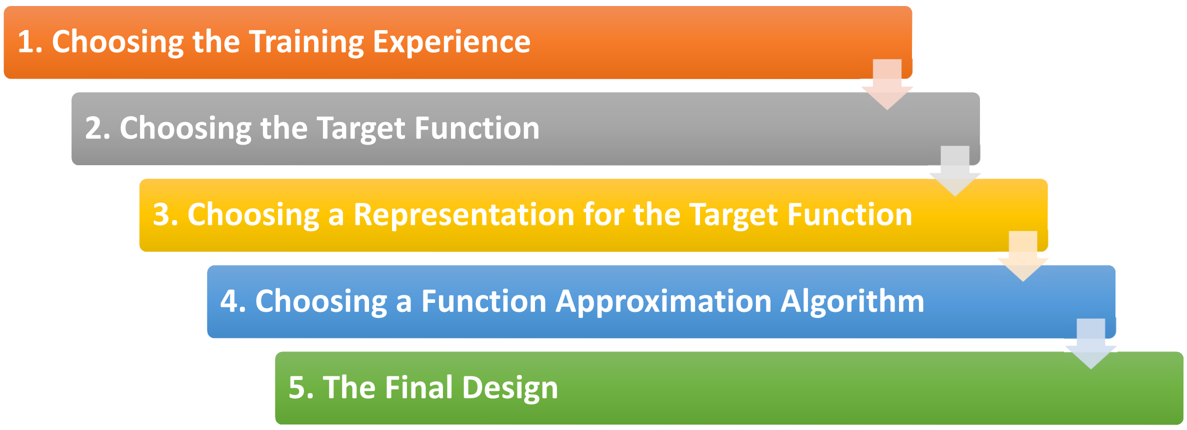
Module 1- Outline

- Chapter 1: Introduction
  - Well posed learning problems
  - **Designing a Learning system**
  - Perspective and Issues in Machine Learning
  - Summary
- Chapter 2: Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Designing a learning system

Given Problem Description, the steps to design a learning system are as follows


1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

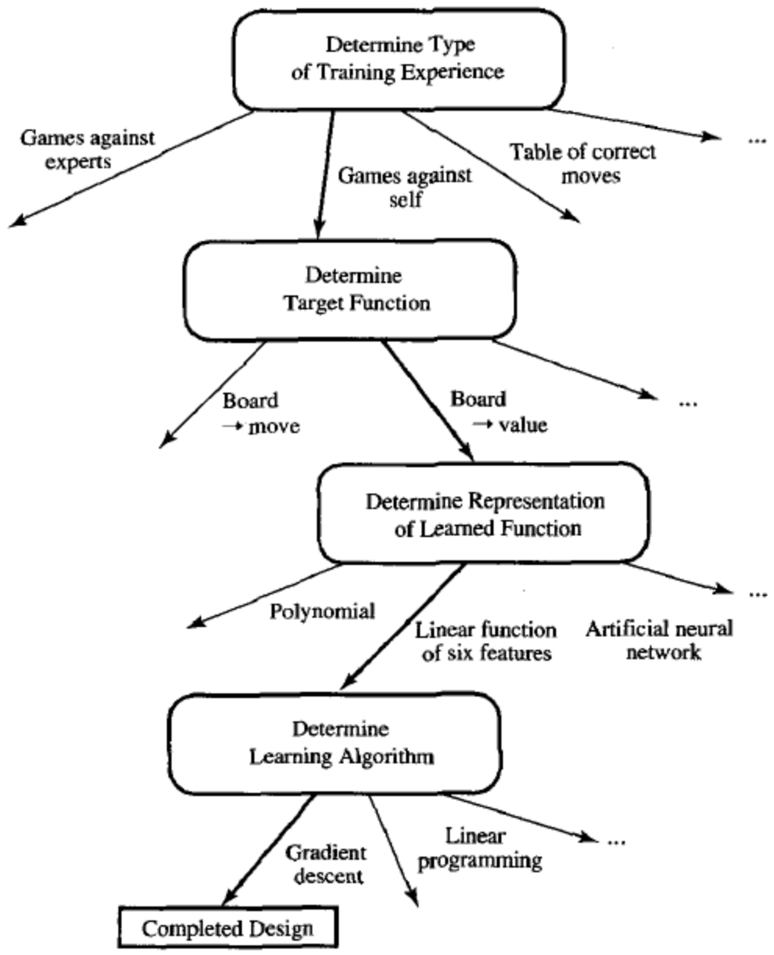
Problem Description:

A Checker Learning Problem

Let us consider designing a program to learn to play checkers, with the goal of entering it in the world checkers tournament.

We adopt the obvious performance measure: the percent of games it wins in this world tournament.

- Task T: Playing Checkers
- Performance Measure P: Percent of games won against opponents
- Training Experience E: To be selected => Games Played against itself



Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

Designing a learning system

Given Problem Description, the steps to design a learning system are as follows

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

Choosing a training experience

The type of training experience available can have a significant impact on success or failure of the learner.

1. Will the training experience provide direct or indirect feedback?

   a. **Direct Feedback:** system learns from examples of individual checkers board states and the correct move for each

   b. **Indirect Feedback:** Move sequences and final outcomes of various games played

      - Credit assignment problem: Value of early states must be inferred from the outcome

Choosing a training experience

2. Degree to which the learner controls the sequence of training examples
    a. Teacher selects informative boards and gives correct move
    b. Learner proposes board states that it finds particularly confusing. Teacher provides correct moves
    c. Learner controls board states and (indirect) training classifications
3. How well it represents the distribution of examples over which the final system performance P must be measured.

Partial Design of Checkers Learning Program

- A checkers learning problem:
    - Task T: playing checkers
    - Performance measure P: percent of games won in the world tournament
    - Training experience E: games played against itself
- Remaining choices
    - The exact type of knowledge to be learned
    - A representation for this target knowledge
    - A learning mechanism

Designing a learning system

Given Problem Description, the steps to design a learning system are as follows

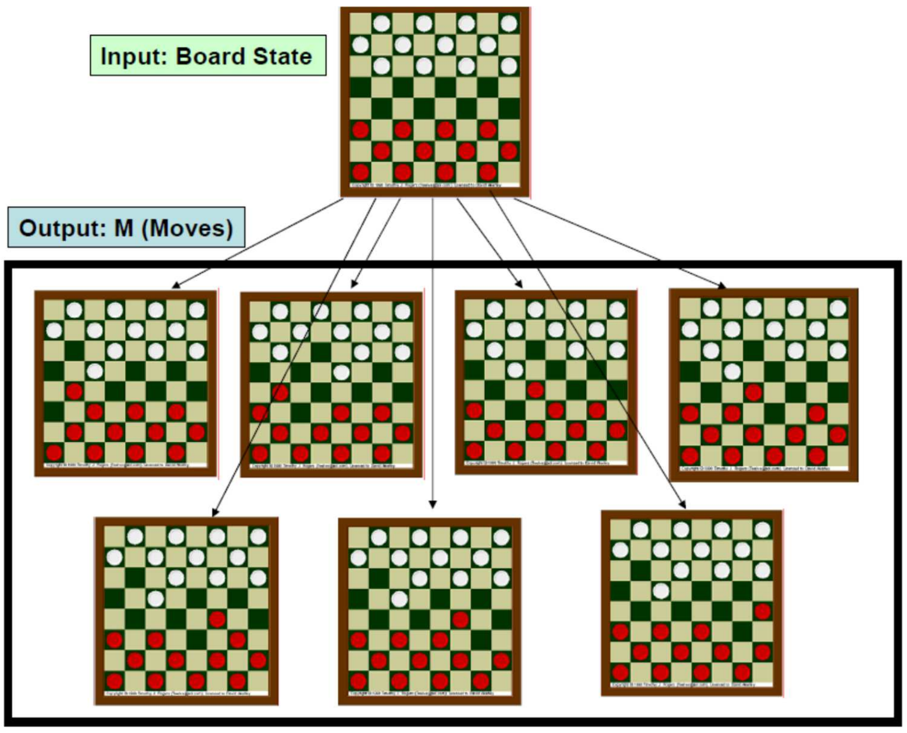
1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

Choosing the Target Function (1)

- Assume that you can determine legal moves
- Program needs to learn the best move from among legal moves
  - Defines large search space known a priori
  - target function: *ChooseMove* : B → M
- *ChooseMove* is difficult to learn given indirect training

- So we go for Alternative target function

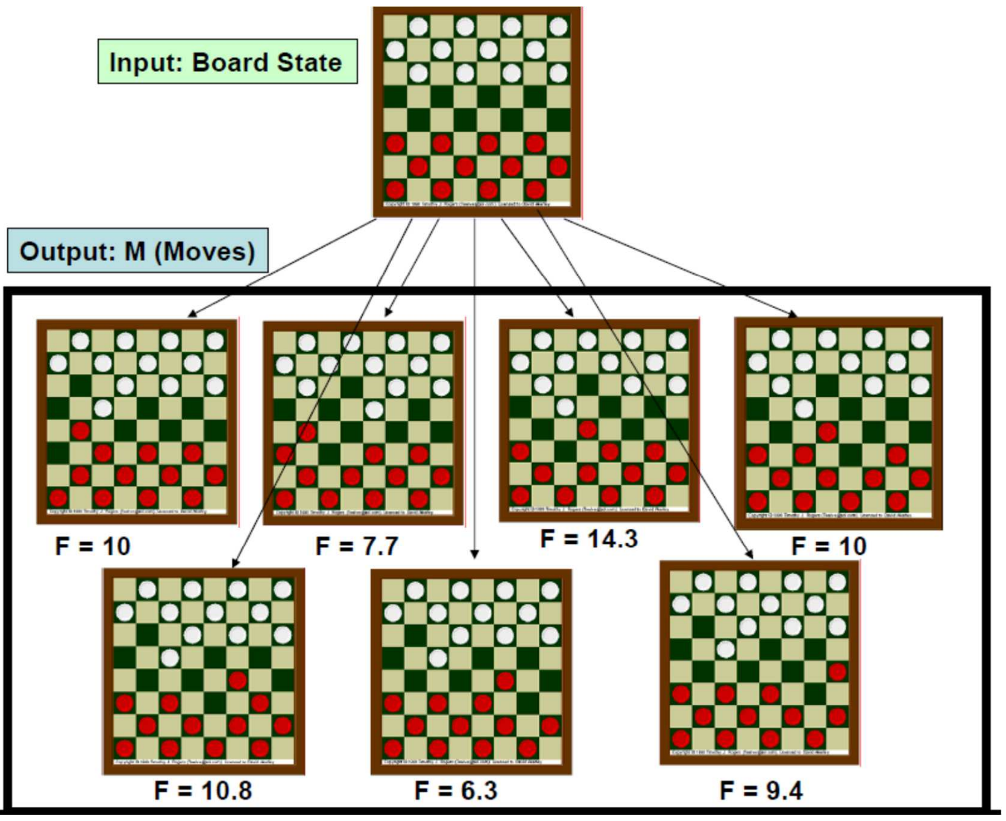
Choosing the Target Function (2)



Choosing the Target Function (3)

- *ChooseMove* is difficult to learn given indirect training
- So we go for Alternative target function
- An evaluation function that assigns a numerical score to any given board state
- $V : B \rightarrow \Re$ (where $\Re$ is the set of real numbers, B is the set of board states)

Choosing the Target Function (4)



Choosing the Target Function (5)

- V(b) for an arbitrary board state b in B
  - if b is a final board state that is won, then V(b) = 100

    …. is lost, then V(b) = -100, … is drawn, then V(b) = 0
  - if b is not a final state, then V(b) = V(b'),

    where b' is the best final board state that can be achieved
  - starting from b and playing optimally until the end of the game
- V(b) gives a recursive definition for board state b
- Not usable because not efficient to compute except is first three trivial cases
- V is a nonoperational definition

Choosing the Target Function (6)

- Goal of learning is to discover an operational description of V
- Learning the target function is often called function approximation
- Approximated function is Referred to as $\hat{V}$

Designing a learning system

Given Problem Description, the steps to design a learning system are as follows

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

Choosing a Representation for the Target function

- Choice of representations involve trade offs
  - Pick a very expressive representation to allow close approximation to the ideal target function V
  - More expressive, more training data required to choose among alternative hypotheses
- Use linear combination of the following board features:
  - x1: the number of black pieces on the board
  - x2: the number of red pieces on the board
  - x3: the number of black kings on the board
  - x4: the number of red kings on the board
  - x5: the number of black pieces threatened by red (i.e. which can be captured on red's next turn)
  - x6: the number of red pieces threatened by black

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Partial Design of Checkers Learning Program

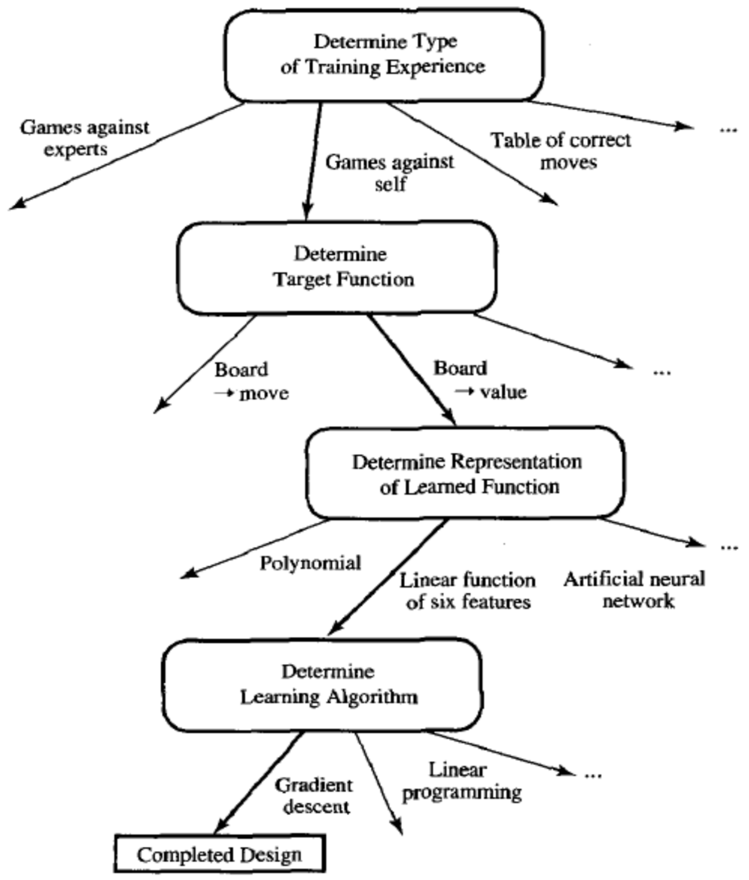
- A checkers learning problem:
  - Task T: playing checkers
  - Performance measure P: percent of games won in the world tournament
  - Training experience E: games played against itself
  - Target Function: V: Board → $\Re$
  - Target function representation

$$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5 + w_6 x_6$$

Designing a learning system

Given Problem Description, the steps to design a learning system are as follows

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design


Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

Choosing a Function Approximation Algorithm

- To learn we require a set of training examples describing the board *b* and the training value $V_{train}(b)$
- Ordered Pair $\langle b, V_{train}(b) \rangle$

$$\langle \langle x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0 \rangle , +100 \rangle$$

Choosing a Function Approximation Algorithm

1. Estimating Training Values

- The only training information available to our learner is whether the game was eventually **won or lost**.
- Despite the ambiguity inherent in estimating training values for intermediate board states, one **simple approach** has been found to be surprisingly successful.
- This approach is to assign the training value of $V_{train}(b)$ for any intermediate board state *b* to be $\hat{V}$ ***(****Successor(b))*

Rule for estimating training values:

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

Choosing a Function Approximation Algorithm

2. Adjusting the weights

One common approach is to define the best hypothesis, or set of weights, as that which minimizes the square error *E* between the training values and the values predicted by the hypothesis *V*.

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \, training \ examples} (V_{train}(b) - \hat{V}(b))^2$$

Choosing a Function Approximation Algorithm

2. Adjusting the weights

We require an algorithm that will incrementally refine the weights as new training examples become available and that will be robust to errors in these estimated training values.

One such algorithm is called the **least mean squares (LMS) training rule**.

LMS Weight update rule

For each training example $\langle b, V_{train}(b)\rangle$

- Use the current weights to calculate $\hat{V}(b)$
- For each weight w_i, update it as

$$w_i \leftarrow w_i + \eta \ (V_{train}(b) - \hat{V}(b)) \ x_i$$

Designing a learning system

Given Problem Description, the steps to design a learning system are as follows

1. Choosing the Training Experience
2. Choosing the Target Function
3. Choosing a Representation for the Target Function
4. Choosing a Function Approximation Algorithm
5. The Final Design

The final design

- The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems.

1. The **Performance System** is the module that must solve the given performance task,
   - in this case playing checkers, by using the learned target function(s).
   - It takes an instance of a new problem (new game) as input and produces a trace of its solution (game history) as output.
2. The **Critic** takes as
   - input - the history or trace of the game and
   - produces as output - a set of training examples of the target function.

The final design

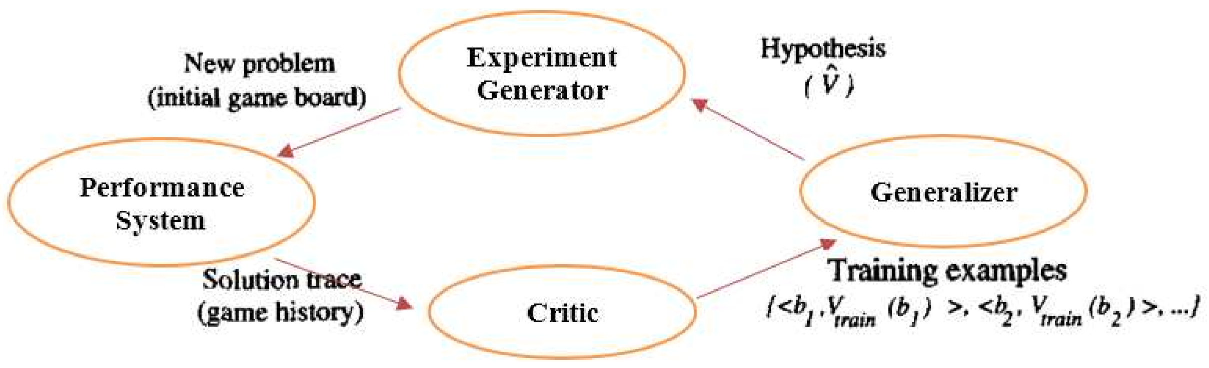
3. The **Generalizer** takes as input the training examples and produces an output hypothesis that is its estimate of the target function.

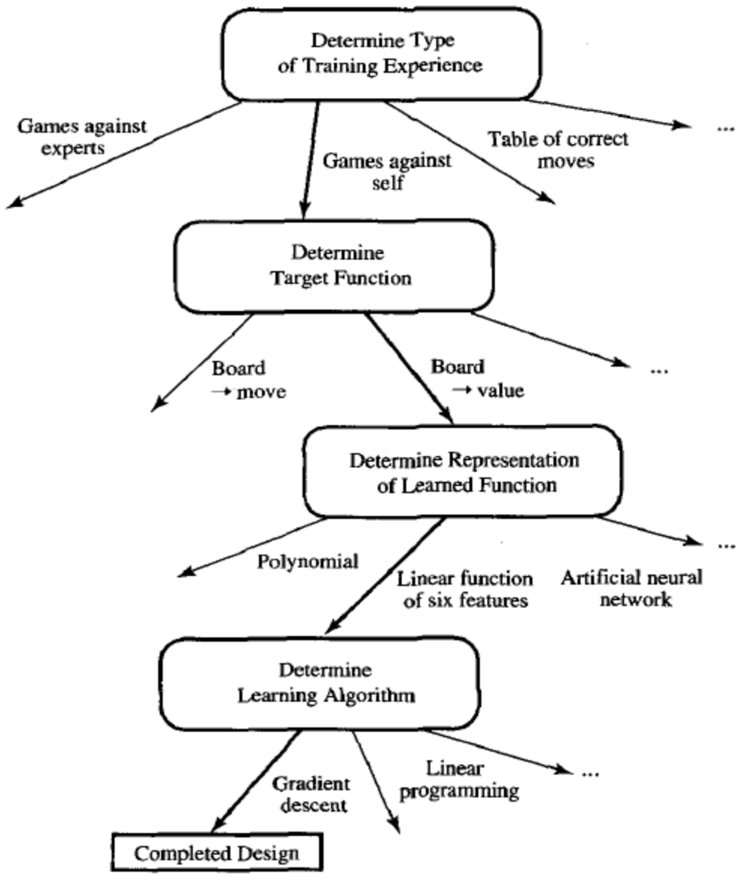
   It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples.

4. The **Experiment Generator** takes as input the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore.

   Its role is to pick new practice problems that will maximize the learning rate of the overall system

Summary of final design

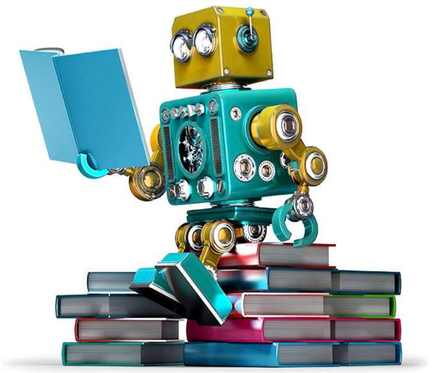

Experiment Generator
New problem (initial game board)
Hypothesis ($\hat{V}$)
Performance System
Generalizer
Solution trace (game history)
Critic
Training examples
$\{<b_1, V_{train}(b_1)>, <b_2, V_{train}(b_2)>, ...\}$


Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

Summary of choices in designing the checkers learning program

Module 1- Outline

- Chapter 1: Introduction
  - Well posed learning problems
  - Designing a Learning system
  - **Perspective and Issues in Machine Learning**
  - Summary
- Chapter 2: Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Perspective in ML

- One useful perspective on machine learning is that
    - it involves searching a very large space of possible hypotheses to determine
    - one that best fits the observed data
    - and any prior knowledge held by the learner.
- For example, consider the space of hypotheses that could in principle be output by the above checkers learner.
- This hypothesis space consists of
    - all evaluation functions
    - that can be represented by
    - some choice of values for the weights w_0 through w_6.

Perspective in ML

- The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with the available training examples.
- The LMS algorithm for fitting weights achieves this goal by iteratively tuning the weights, adding a correction to each weight each time the hypothesized evaluation function predicts a value that differs from the training value.
- This algorithm works well when the hypothesis representation considered by the learner defines a continuously parameterized space of potential hypotheses.

Issues in ML

Our checkers example raises a number of generic questions about ML. It is concerned with answering questions like;

- What **algorithms** exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations?
- **How much training data** is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?

Issues in ML

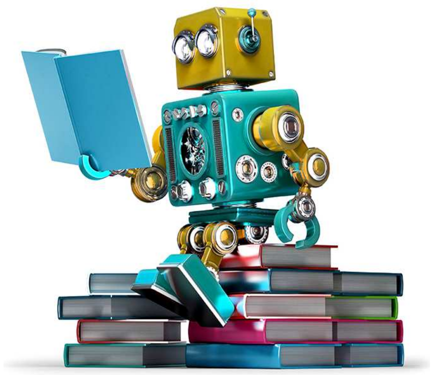
- When and how can **prior knowledge** held by the learner guide the process of generalizing from examples?

  Can prior knowledge be helpful even when it is only approximately correct?
- What is the **best strategy** for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
- What is the best way to **reduce the learning task** to one or more function approximation problems?

  Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?
- How can the learner **automatically alter its representation** to improve its ability to represent and learn the target function?

Module 1- Outline

- Chapter 1: Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - **Summary**
- Chapter 2: Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Summary

- ML have proven to be of great value in a variety of applications.
  1. Data mining problems containing large databases
     a. To analyze outcomes of medical treatments from patient databases
     b. To learn general rules for credit worthiness from financial databases
  2. Poorly understood domains where humans might not have the knowledge needed to develop effective algorithms
     e.g., human face recognition from images
  3. Domains where the program must dynamically adapt to changing conditions
     a. controlling manufacturing processes under changing supply stocks
     b. adapting to the changing reading interests of individuals.

Summary

- Machine learning draws on ideas from a diverse set of disciplines, including AI, probability and statistics, computational complexity, information theory, psychology and neurobiology, control theory, and philosophy.
- A well-defined learning problem requires a well-specified task, performance metric, and source of training experience.
- Designing a machine learning approach involves a number of design choices, including
  - choosing the type of training experience,
  - the target function to be learned,
  - a representation for this target function, and
  - an algorithm for learning the target function from training examples.

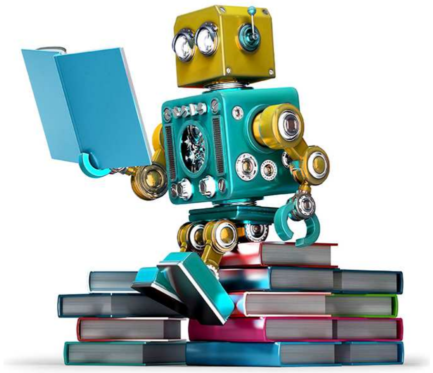
Summary

- Learning involves search:

  searching through a space of possible hypotheses to find the hypothesis that best fits the available training examples and other prior constraints or knowledge.

Module 1- Outline

- Chapter 1: Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- **Chapter 2: Concept Learning**
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary

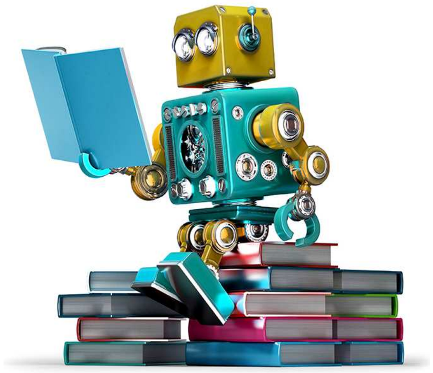


Concept Learning

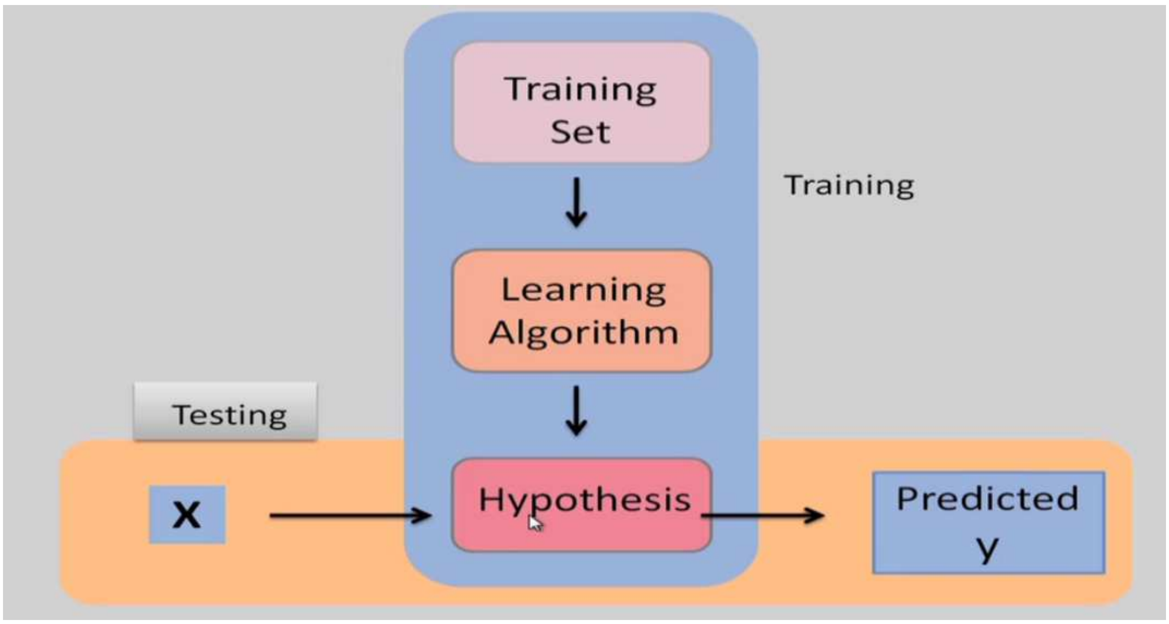
- Inducing general functions from specific training examples is a main issue of machine learning.
- A task of acquiring a potential hypothesis (solution) that best fits the training examples
- It is the process of acquiring the definition of a general category from given sample positive and negative training examples of the category.
- Concept Learning can seen as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples.
- **Concept learning**: Inferring a boolean-valued function from training examples of its input and output.

Module 1- Outline

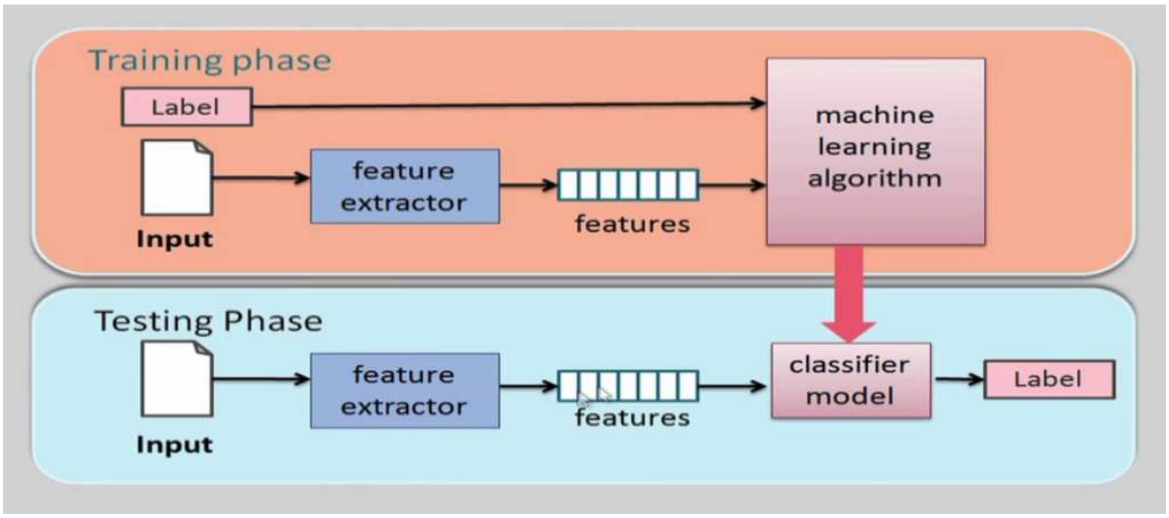
- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - **Concept learning task**
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Hypothesis



Classification

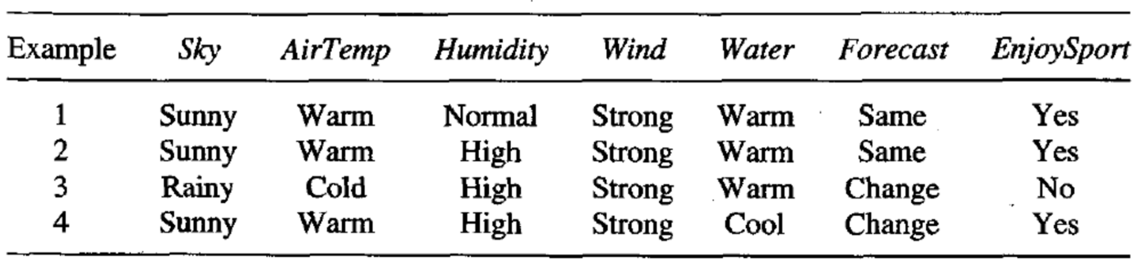


Concept Learning Task

- Problem is to learning the target concept

  "Days on which my friend Sachin enjoys his favorite water sport".

- Given



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*.

Concept Learning Task

- What hypothesis representation shall we provide to the learner in this case?
- For each attribute, the hypothesis will either
  - indicate by a “ ? ” that any value is acceptable for this attribute,
  - specify a single required value (e.g., Warm) for the attribute, or
  - indicate by a "Φ" that no value is acceptable.
- If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (h(x) = 1).
- hypothesis that Sachin enjoys his favorite sport only on cold days with high humidity : (?, Cold, High, ?, ?, ?)
- Most general hypothesis - (?, ?, ?, ?, ?, ?)
- Most specific possible hypothesis- (Φ, Φ, Φ, Φ, Φ, Φ)

“EnjoySport” Concept Learning Task

- **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.

Inductive learning hypothesis

- Our assumption is that the Fundamental assumption of inductive learning.

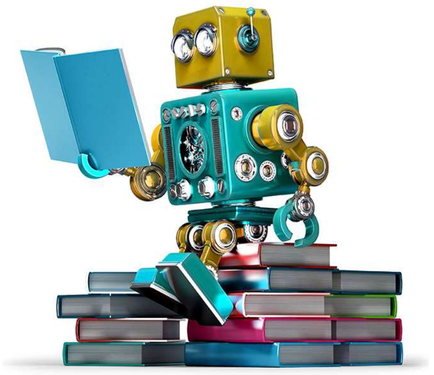
  "Best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data."

- **The 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.

Module 1- Outline

- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - Concept learning task
  - **Concept learning as search**
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Concept learning as search

- Concept learning can be viewed as the task of searching through a large space of hypotheses
- The goal is to find the hypothesis that best fits training examples.
- **General-to-Specific Ordering of Hypotheses**

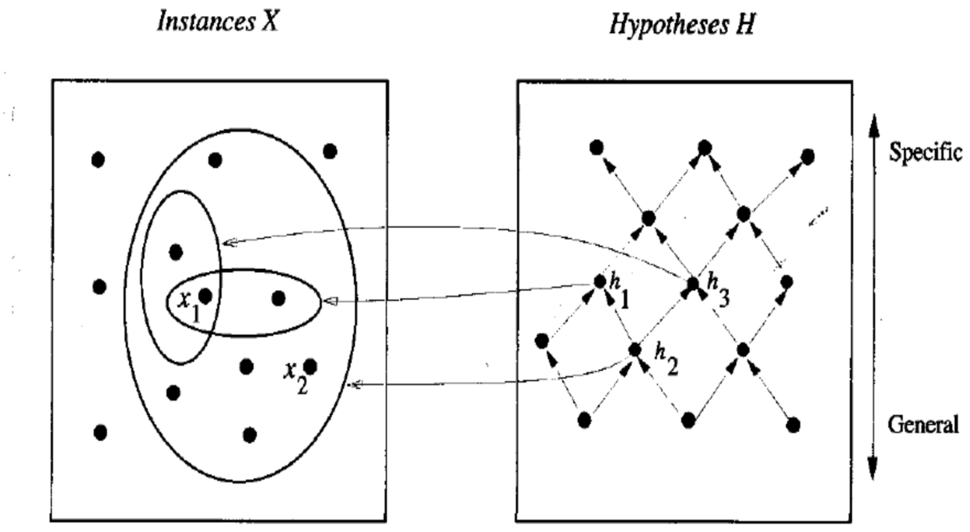
$$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$$
$$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$$

h_2 is more general than h_1.

***Definition*:** 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 $h_j \geq_g h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

Concept learning as search



x_1 = <Sunny, Warm, High, Strong, Cool, Same>

x_2 = <Sunny, Warm, High, Light, Warm, Same>

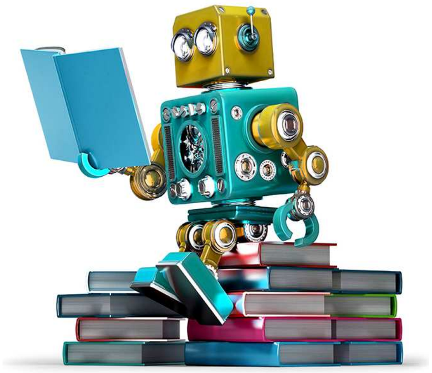
h_1 = <Sunny, ?, ?, Strong, ?, ?>

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

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

Module 1- Outline

- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - Concept learning task
  - Concept learning as search
  - **Find-S algorithm**
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Find-S: Finding A Maximally Specific Hypothesis

- How can we use the ***more-general-than*** partial ordering to organize the search for a hypothesis consistent with the observed training examples?
- One way is to begin with the most specific possible hypothesis in H, then generalize this hypothesis each time it fails to cover an observed positive training example.
- FIND-S algorithm is used for this purpose.

Find-S Algorithm

To Find Maximally Specific Hypothesis

1. Initialize *h* to the most specific hypothesis in *H*
2. For each positive training instance *x*

   For each attribute constrain a_i in *h*

   - If the constraint a_i 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*

Illustration

x_1 = <Sunny Warm Normal Strong Warm Same>, +

x_2 = <Sunny Warm High Strong Warm Same>, +

x_3 = <Rainy Cold High Strong Warm Change>, -

x_4 = <Sunny Warm High Strong Cool Change>, +

$h_0 = <\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>$

h_1 = <Sunny Warm Normal Strong Warm Same>

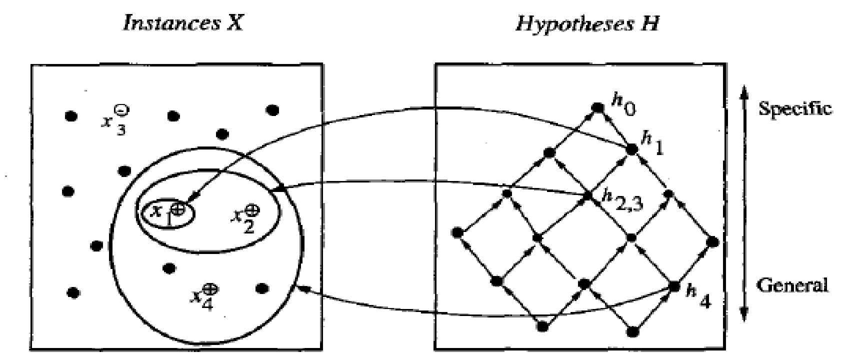
h_2 = <Sunny Warm ? Strong Warm Same>

h_3 = <Sunny Warm ? Strong Warm Same>

h_4 = <Sunny Warm ? Strong ? ? >

Key Property

- The key property of the Find-S algorithm is that for hypothesis spaces described by conjunctions of attribute constraints
- Find-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.



x_1 = <Sunny Warm Normal Strong Warm Same>, +
x_2 = <Sunny Warm High Strong Warm Same>, +
x_3 = <Rainy Cold High Strong Warm Change>, -
x_4 = <Sunny Warm High Strong Cool Change>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
h_1 = <Sunny Warm Normal Strong Warm Same>
h_2 = <Sunny Warm ? Strong Warm Same>
h_3 = <Sunny Warm ? Strong Warm Same>
h_4 = <Sunny Warm ? Strong ? ? >

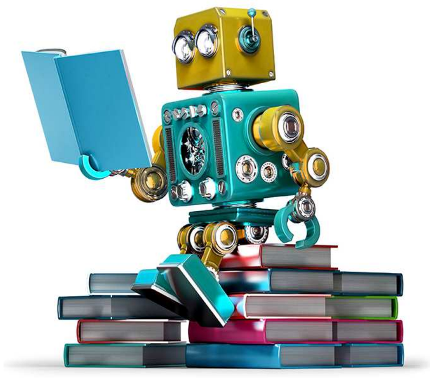
Find S - Drawback

Questions still left unanswered

- Has the learner converged to the correct target concept?
- Why prefer the most specific hypothesis?
  If multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis
- Are the training examples consistent?
  Training examples may contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, since it ignores negative examples.
- What if there are several maximally specific consistent hypotheses?
  There can be several maximally specific hypotheses consistent with the data. Find S finds only one.

Module 1- Outline

- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - **Version space**
  - Candidate Elimination algorithm
  - Inductive Bias
  - Summary



Definition: Consistent

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

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x)$$

- Notice the key difference between this definition of ***consistent*** and our earlier definition of ***satisfies***.
- An example x is said to satisfy hypothesis h when *h(x) = 1*, regardless of whether x is a positive or negative example of the target concept.
- However, whether such an example is consistent with h depends on the target concept, and in particular, whether *h(x) = c(x).*

Definition: Version Space

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

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$

- This subset of all hypotheses is called the **version space** with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.

The List-Then-Eliminate algorithm

- One obvious way to represent the version space is simply to list all of its members.
- This leads to a simple learning algorithm, which we might call the List-Then-Eliminate algorithm.

The LIST-THEN-ELIMINATE Algorithm

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

Few definition

Definition: The **general boundary** G, with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D.

$$G \equiv \{g \in H | Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

Definition: The **specific boundary** S, with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D.

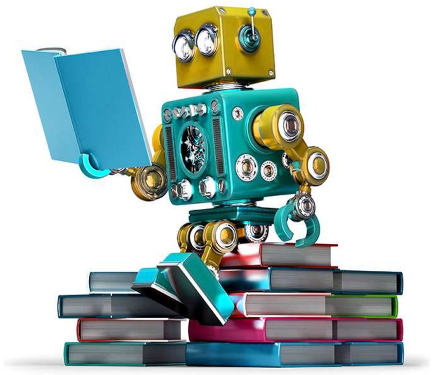
$$S \equiv \{s \in H | Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

Theorem 2.1. Version space representation theorem. Let X be an arbitrary set of instances and let H be a set of boolean-valued hypotheses defined over X. Let $c : X \rightarrow \{0, 1\}$ be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples $\{\langle x, c(x)\rangle\}$. For all X, H, c, and D such that S and G are well defined,

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

Module 1- Outline

- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - **Candidate Elimination algorithm**
  - Inductive Bias
  - Summary



Candidate Elimination Algorithm

- The CEA computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.
- It begins by
  - initializing the version space to the set of all hypotheses in H; that is, G-most general hypothesis in H

$$G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\}$$

  - and initializing the ***S*** boundary set to contain the most specific (least general) hypothesis

$$S_0 \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$$

Candidate Elimination Algorithm using Version Spaces

1. Initialize G to the set of maximally general hypotheses in H
2. Initialize S to the set of maximally specific hypotheses in H
3. For each training example d, do
   a. 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, hypothesis that is more general than another hypothesis in S

   b. 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 in G

Illustration

Training examples:

1 . *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*

2 . *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*

3. *<Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No*

4. *<Sunny, Warm, High, Strong, Cool, Change>, EnjoySport = Yes*

$$S_0: \boxed{\{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}}$$

$$G_0 \quad \boxed{\{<?, ?, ?, ?, ?, ?>\}}$$

Illustration

Training examples:

1 . *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*

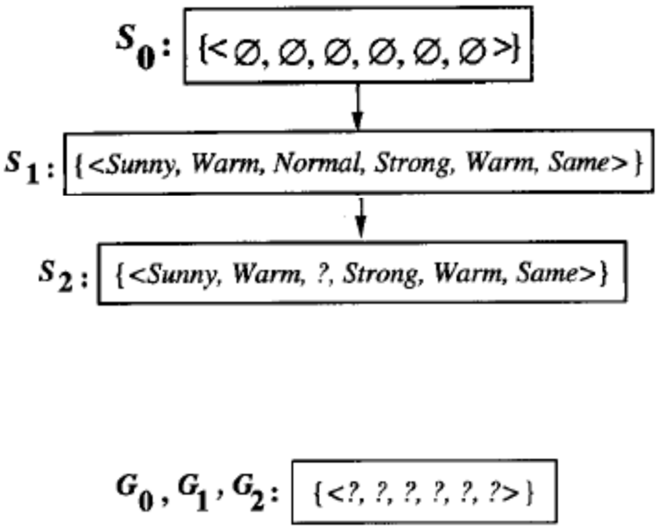
$$\begin{tikzcd} S_0: \boxed{\{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}} \arrow[d] \\ S_1: \boxed{\{<Sunny, Warm, Normal, Strong, Warm, Same>\}} \end{tikzcd}$$

$$G_0\ , G_1 \quad \boxed{\{<?, ?, ?, ?, ?, ?>\}}$$

Illustration

Training examples:

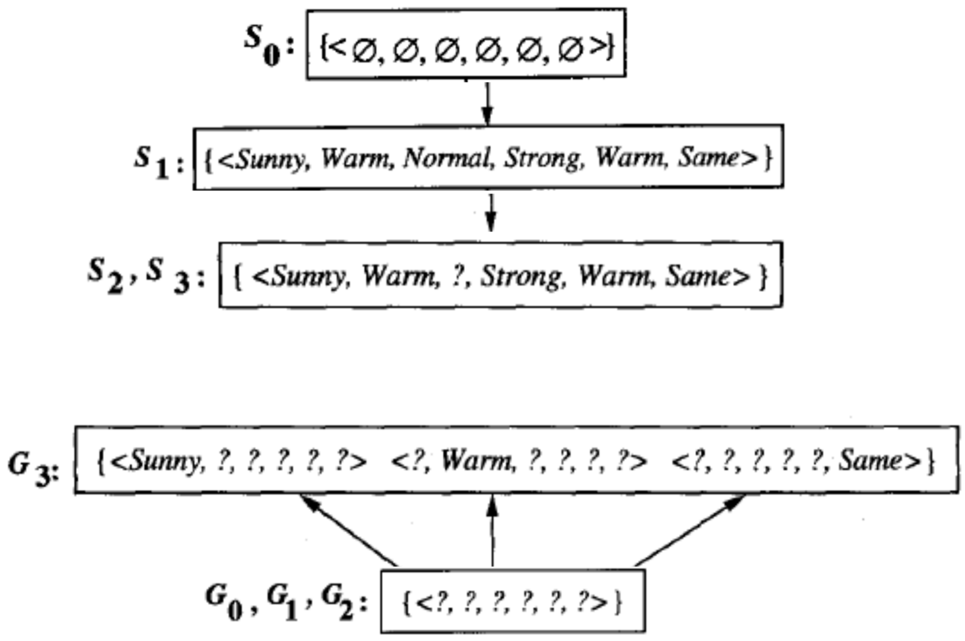
1. *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*
2. *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*



S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}
S_1: {<Sunny, Warm, Normal, Strong, Warm, Same>}
S_2: {<Sunny, Warm, ?, Strong, Warm, Same>}
G_0, G_1, G_2: {<?, ?, ?, ?, ?, ?>}

Illustration

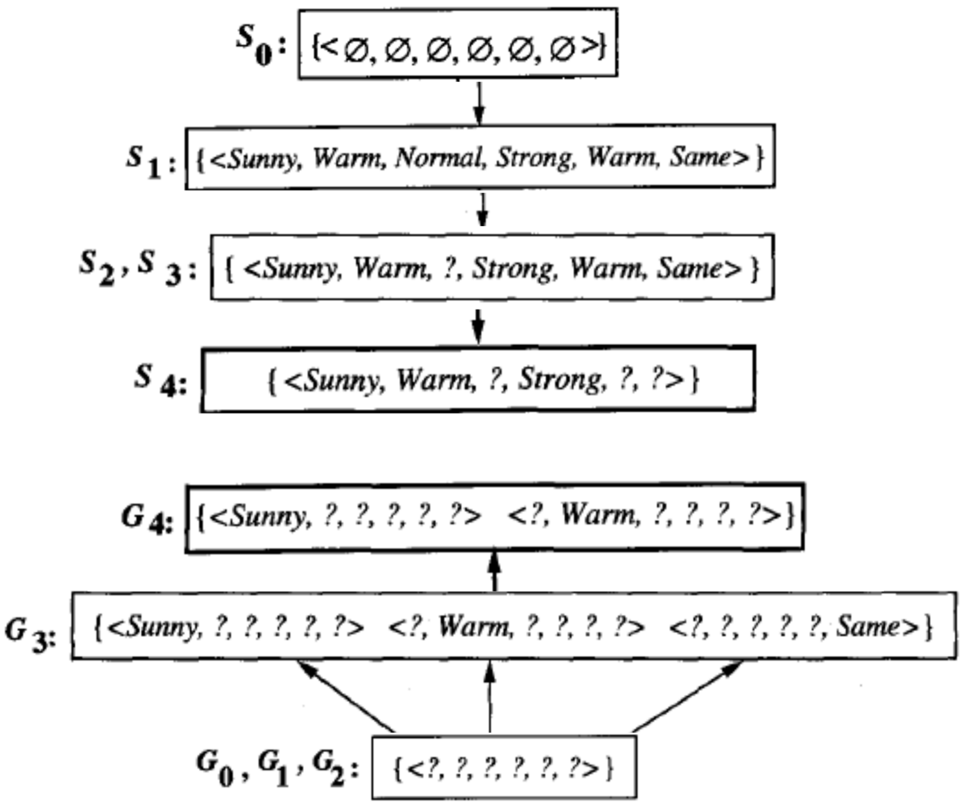
1. *<Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes*
2. *<Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes*
3. *<Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No*



S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}
S_1: {<Sunny, Warm, Normal, Strong, Warm, Same>}
S_2, S_3: { <Sunny, Warm, ?, Strong, Warm, Same> }
G_3: {<Sunny, ?, ?, ?, ?, ?> <?, Warm, ?, ?, ?, ?> <?, ?, ?, ?, ?, Same>}
G_0, G_1, G_2: {<?, ?, ?, ?, ?, ?>}

Illustration

4.<Sunny, Warm, High, Strong, Cool, Change>, EnjoySport = Yes



S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}
S_1: {<Sunny, Warm, Normal, Strong, Warm, Same>}
S_2, S_3: { <Sunny, Warm, ?, Strong, Warm, Same> }
S_4: { <Sunny, Warm, ?, Strong, ?, ?> }
G_4: {<Sunny, ?, ?, ?, ?, ?> <?, Warm, ?, ?, ?, ?>}
G_3: {<Sunny, ?, ?, ?, ?, ?> <?, Warm, ?, ?, ?, ?> <?, ?, ?, ?, ?, Same>}
G_0, G_1, G_2: {<?, ?, ?, ?, ?, ?>}

Illustration

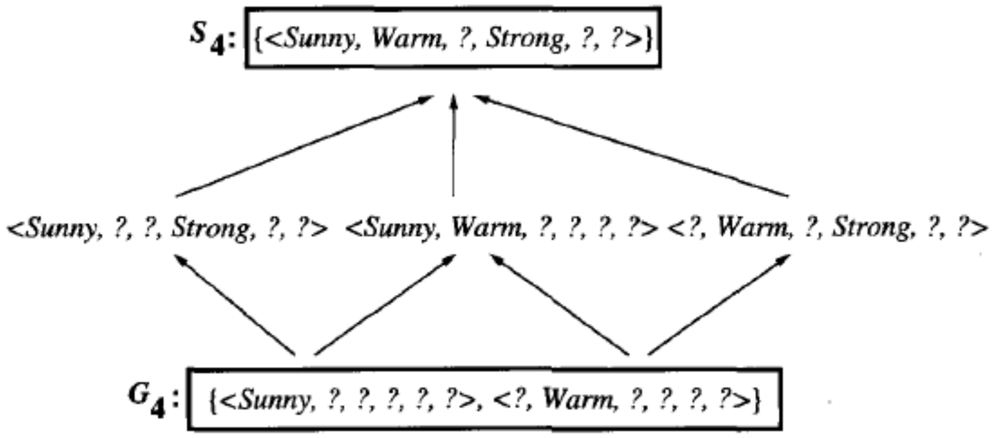
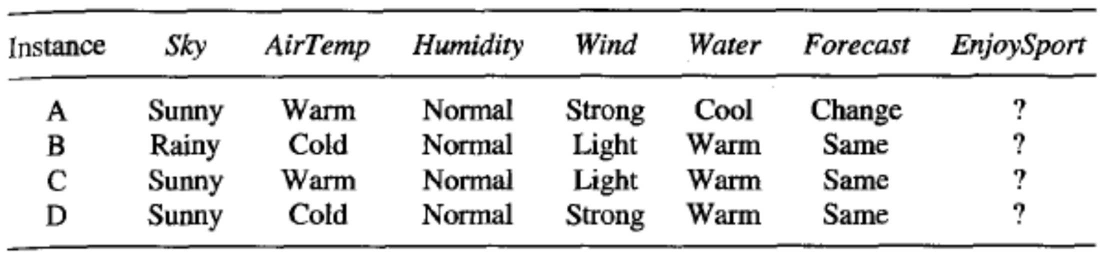


FIGURE 2.7
The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

Illustration

The final version space for the ***EnjoySport*** concept learning problem

<Sunny, ?, ?, Strong, ?, ?> <Sunny, Warm, ?, ?, ?, ?> <?, Warm, ?, Strong, ?, ?>



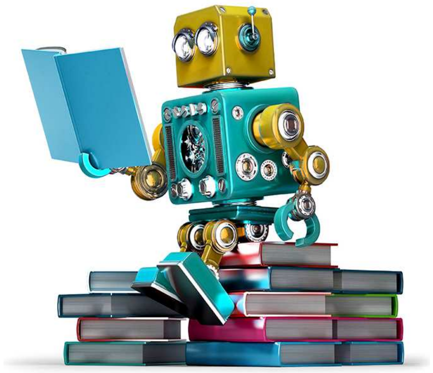
Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

TABLE 2.6
New instances to be classified.

Illustration

- After processing these four examples, the boundary sets S_4 and G_4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples.
- The entire version space, including those hypotheses bounded by S_4 and G_4.
- This learned version space is independent of the sequence in which the training examples are presented
- As further training data is encountered, the S and G boundaries will move monotonically closer to each other

Module 1- Outline



Inductive Bias

- The CEA will converge toward the true target concept provided it is given accurate training examples
- The fundamental questions for inductive inference in general.
  - What if the target concept is not contained in the hypothesis space?
  - How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
  - How does the size of the hypothesis space influence the number of training examples that must be observed?
- Here we examine them in the context of the CEA.
- The conclusions can be extended to any concept learning system.

Inductive Bias

1. A Biased Hypothesis Space

- Suppose we wish to assure that the hypothesis space contains the unknown target concept.
- The obvious solution is to enrich the hypothesis space to include every possible hypothesis.
- Consider *EnjoySport* example in which we restricted the hypothesis space to include only conjunctions of attribute values.

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

Inductive Bias

1. A Biased Hypothesis Space (Continued)

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

- Most specific hypothesis consistent with the first two examples

$$S_2 : \langle ?, Warm, Normal, Strong, Cool, Change \rangle$$

- It incorrectly covers the third (negative) training example
- The problem is that we have biased the learner to consider only conjunctive hypotheses.
- In this case we require a more expressive hypothesis space.

Inductive Bias

2. An unbiased learner

- The obvious solution to be a unbiased learner– design hypothesis space H to represent ***every teachable concept***;
- It should capable of representing every possible subset of the instances X. In general, the set of all subsets of a set X is called the ***power-set*** of X.
- In general, number of distinct subsets is $2^{|X|}$.
- Thus, there are 2^{96}, or approximately distinct target concepts that could be defined over this instance space and that our learner might be called upon to learn.
- Our conjunctive hypothesis space is able to represent only 973 of these-a very biased hypothesis space indeed!

Inductive Bias

2. An unbiased learner (Continued)

- Let us reformulate the *Enjoysport* learning task
- Let H’ represent every subset of instances; that is, let H' correspond to the power set of X.
- One way to define such an H' is to allow arbitrary disjunctions, conjunctions, and negations of our earlier hypotheses.
- For instance, the target concept "Sky = Sunny or Sky = Cloudy" could then be described as

$$\langle Sunny, ?, ?, ?, ?, ?\rangle \vee \langle Cloudy, ?, ?, ?, ?, ?\rangle$$

Inductive Bias

2. An unbiased learner (Continued)

- Now new problem: we are completely unable to generalize beyond the observed examples!
- To see why, suppose we present three positive examples (x_l, x_2, x_3) and two negative examples (x_4, x_5) to the learner.
- At this point, the S and G boundary of the version space will be

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

- Here in order to converge to a single, final target concept, we will have to present every single instance in X as a training example!

Inductive Bias

3. The Futility of Bias-Free Learning

- The fundamental property of inductive inference:

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

- CEA generalizes observed training examples because it was biased by the implicit assumption that the target concept could be represented by a conjunction of attribute values.
  - If this assumption is correct (and the training examples are error-free), its classification of new sample will also be correct.
  - If this assumption is incorrect, however, it is certain that the CEA will mis-classify at least some instances from X.

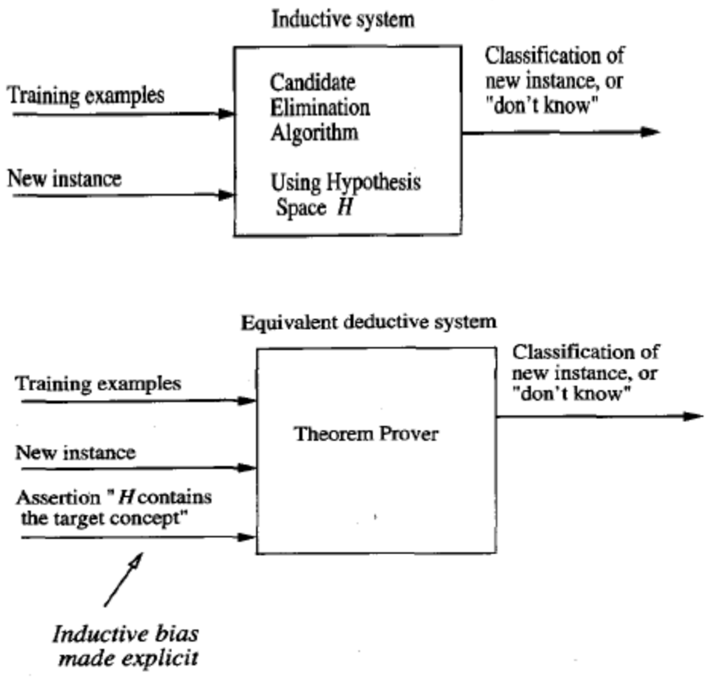
Inductive bias

- Consider a concept learning algorithm L for the set of instances X.
- Let c be an arbitrary concept defined over X, and let D_c = $\{\langle x, c(x) \rangle\}$ 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 B such that for any target concept c and corresponding training examples D_c

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

Inductive bias of CEA

"The target concept c is contained in the given hypothesis space H".

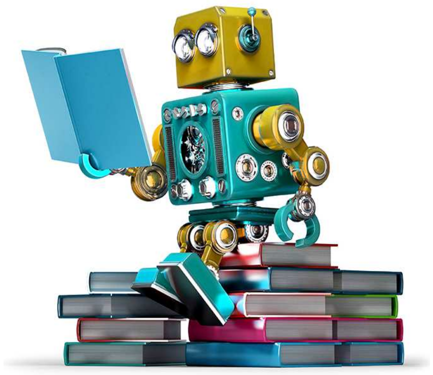

Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space H
Classification of new instance, or "don't know"
Equivalent deductive system
Training examples
New instance
Assertion "H contains the target concept"
Theorem Prover
Classification of new instance, or "don't know"
Inductive bias made explicit

Algorithms listed from weakest to strongest bias

- Rote-Learner: Learning corresponds simply to storing each observed training example in memory.
- CEA: New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.
- FIND-S: This algorithm, described earlier, finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

Module 1- Outline

- Introduction
  - Well posed learning problems
  - Designing a Learning system
  - Perspective and Issues in Machine Learning
  - Summary
- Concept Learning
  - Concept learning task
  - Concept learning as search
  - Find-S algorithm
  - Version space
  - Candidate Elimination algorithm
  - Inductive Bias
  - **Summary**



Summary …(1)

- Concept learning can be cast as a problem of searching through a large predefined space of potential hypotheses.
- The general-to-specific partial ordering of hypotheses, provides a useful structure for organizing the search through the hypothesis space.
- The Find-S algorithm
  utilizes general-to-specific ordering,
  performing a specific-to-general search
  through the hypothesis space
  along one branch of the partial ordering,
  to find the most specific hypothesis
  consistent with the training examples.

Summary …(2)

- The Candidate Elimination Algorithm
  utilizes this general-to-specific ordering
  to compute the version space
  (the set of all hypotheses consistent with the training data)
  by incrementally computing the sets of maximally specific (S)
  and maximally general (G) hypotheses.
- Because the S and G sets delimit the entire set of hypotheses consistent with the data, they provide the learner with a description of its uncertainty regarding the exact identity of the target concept.

Summary ...(3)

- Version spaces and the CEA provide a useful conceptual framework for studying concept learning.
- However, CEA is not robust to noisy data or to situations in which the unknown target concept is not expressible in the provided hypothesis space.
- Inductive learning algorithms
  are able to classify unseen examples
  only because of their implicit inductive bias
  for selecting one consistent hypothesis over another.

THANK YOU!