Machine Learning

Introduction

Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Well-Posed Learning Problems

- *Definition*:

  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*.

Well-Posed Learning Problems : Examples

- A checkers learning problem
  - Task *T* : playing checkers
  - Performance measure *P* : percent of games won against opponents
  - Training experience *E* : playing practice games against itself
- 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

Well-Posed Learning Problems : Examples (cont.)

- A robot driving learning problem
  - Task *T* : driving on public four-lane highways using vision sensors
  - Performance measure *P* : average distance traveled before an error (as judged by human overseer)
  - Training experience *E* : a sequence of images and steering commands recorded while observing a human driver

Designing a Learning System

- Choosing the Training Experience
- Choosing the Target Function
- Choosing a Representation for the Target Function
- Choosing a Function Approximation Algorithm
- The Final Design

Choosing the Training Experience

- Whether the training experience provides direct or indirect feedback regarding the choices made by the performance system:
- Example:
  - Direct training examples in learning to play checkers consist of individual checkers board states and the correct move for each.
  - Indirect training examples in the same game consist of the move sequences and final outcomes of various games played in which information about the correctness of specific moves early in the game must be inferred indirectly from the fact that the game was eventually won or lost – *credit assignment problem.*

Choosing the Training Experience (cont.)

- The degree to which the learner controls the sequence of training examples:
- Example:
  - The learner might rely on the teacher to select informative board states and to provide the correct move for each
  - The learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move. Or the learner may have complete control over the board states and (indirect) classifications, as it does when it learns by playing against itself with no teacher present.

Choosing the Training Experience (cont.)

- How well it represents the distribution of examples over which the final system performance P must be measured: In general learning is most reliable when the training examples follow a distribution similar to that of future test examples.
- Example:
  - If the training experience in play checkers consists only of games played against itself, the learner might never encounter certain crucial board states that are very likely to be played by the human checkers champion. (Note however that the most current theory of machine learning rests on the crucial assumption that the distribution of training examples is identical to the distribution of test examples)

Choosing the Target Function

- To determine what type of knowledge will be learned and how this will be used by the performance program:
- Example:
  - In play checkers, it needs to learn to choose the best move among those legal moves: *ChooseMove*: $B \rightarrow M$, which accepts as input any board from the set of legal board states *B* and produces as output some move from the set of legal moves *M*.

Choosing the Target Function (cont.)

- Since the target function such as *ChooseMove* turns out to be very difficult to learn given the kind of indirect training experience available to the system, an alternative target function is then an evaluation function that assigns a numerical score to any given board state, $V: B \rightarrow R$.

Choosing a Representation for the Target Function

- Given the ideal target function *V*, we choose a representation that the learning system will use to describe *V'* that it will learn:
- Example:
  - In play checkers,

$$V'(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

  - where w_i is the numerical coefficient or weight to determine the relative importance of the various board features and x_i is the number of *i*-th objects on the board.

Choosing a Function Approximation Algorithm

- Each training example is given by $<b, V_{train}(b)>$ where $V_{train}(b)$ is the training value for a board b.
- Estimating Training Values:

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

- Adjusting the weights: To specify the learning algorithm for choosing the weights w_i to best fit the set of training examples $\{<b, V_{train}(b)>\}$, which minimizes the squared error E between the training values and the values predicted by the hypothesis V‘

- $$E = \sum_{<b,V\text{train}(b)>\in \text{training examples}} (V_{\text{train}}(b) - V'(b))^2$$

Choosing a Function Approximation Algorithm (cont.)

- $$E = \sum_{<b, V_{train}(b)> \in \text{ training examples}} (V_{train}(b) - V'(b))^2$$
- To minimize *E*, the following rule is used:

LMS weight update rule

```
For each training example <b, V_train(b)>
    Use the current weights to calculate V'(b)
    For each weight w_i , update it as
        $w_i \leftarrow w_i + \eta (V_{train}(b) - V'(b)) x_i$
```

The Final Design

- Performance System: To solve the given performance task by using the learned target function(s). It takes an instance of a new problem (new game) as input and a trace of its solution (game history) as output.

- Critic: To take as input the history or trace of the game and produce as output a set of training examples of the target function.

The Final Design (cont.)

- Generalizer: To take as input the training examples and produce 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.

The Final Design (cont.)

- Experiment Generator: To take as input the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for Performance System to explore. Its role is to pick new practice problems that will maximize the learning rate of the overall system.

The Final Design (cont.)

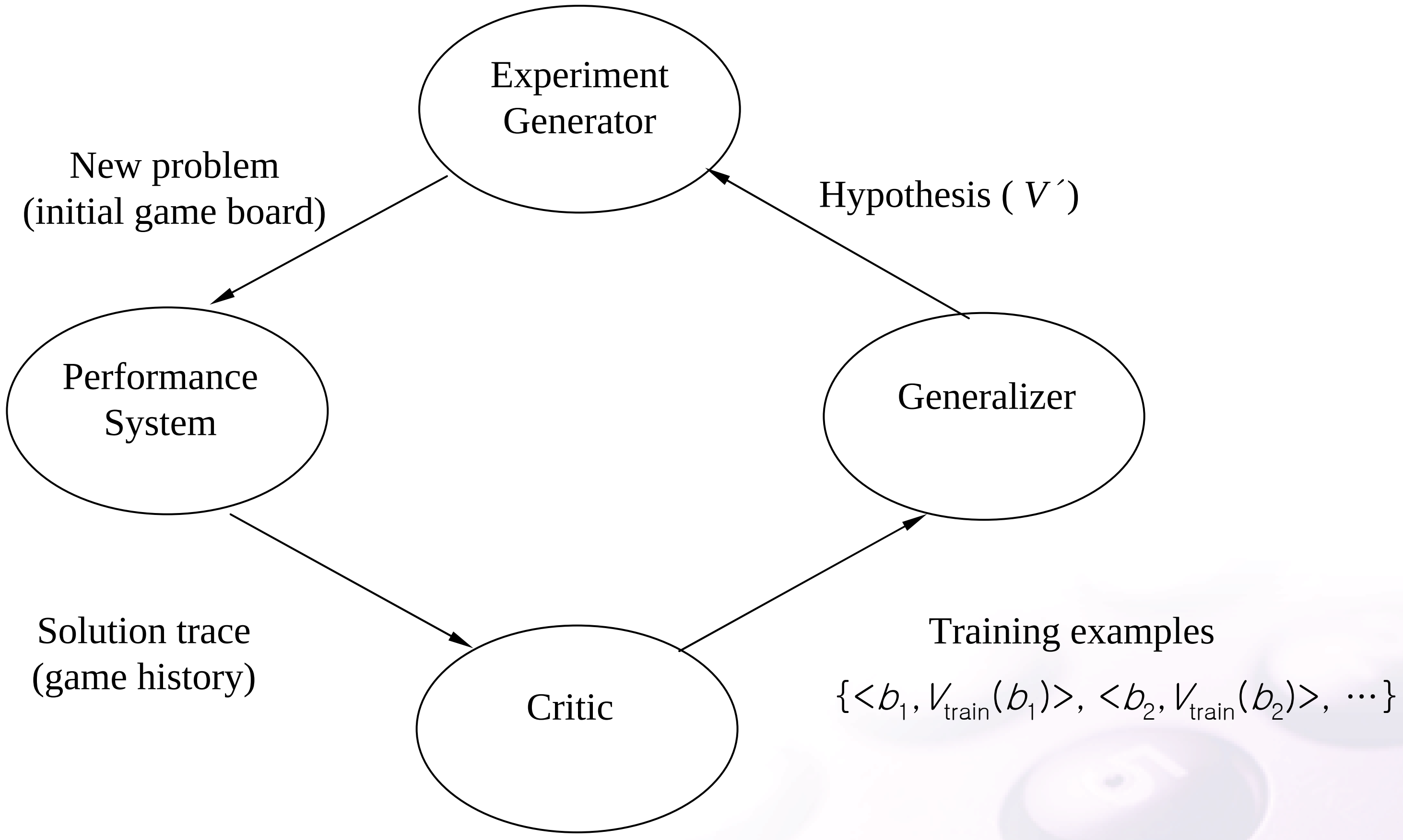
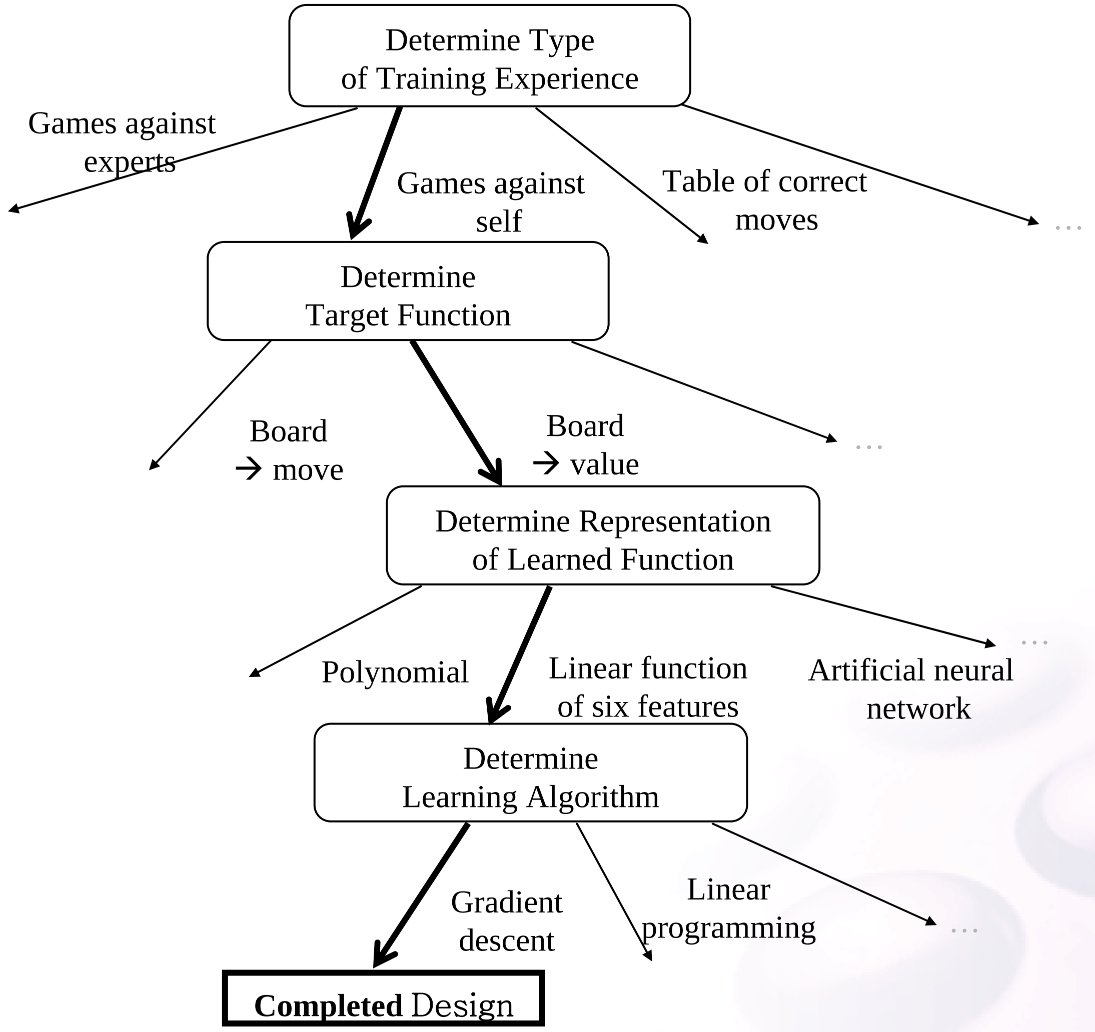

Experiment Generator
New problem (initial game board)
Hypothesis (V')
Performance System
Generalizer
Solution trace (game history)
Critic
Training examples
$\{\langle b_1, V_{train}(b_1)\rangle, \langle b_2, V_{train}(b_2)\rangle, \cdots\}$

Figure 1.1 Final design of the checkers learning program

Choices in Designing the Checkers Learning Problem


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

Issues in Machine Learning

- What algorithms exist for learning general target functions from specific training examples ?
- How does the number of training examples influence accuracy ?
- When and how can prior knowledge held by the learner guide the process of generalizing from examples ?

Issues in Machine Learning (cont.)

- 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 ?

- How can the learner automatically alter its representation to improve its ability to represent and learn the target function ?