Module--1

INTRODUCTION

Introduction (1a)

Introduction

Learning Problems

Designing Learning systems

Perspectives and Issues

Introduction

Machine learning is inherently **multidisciplinary** field.

It **derives** from Artificial intelligence, probability & statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology and other fields.

Machine learning will play an **increasingly central role** in computer science and computer technology.

Few Scenarios for ML

Computers learning from medical records like which treatments are most effective for new diseases.

Personal software assistants learning the evolving interests of their users in order to highlight especially relevant stories from the online morning newspaper.

Well Posed Learning Problems

To include any computer program that improves its performance at some task through experience.

“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”

Contd….

Well defined learning problem should have following features:

- The class of **tasks** (T)
- The measure of **performance** to be improved (P)
- The source of **experience** (E)

Example1

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

Example2

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

Example3

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

Training experience E: a sequence of images and steering commands recorded while observing human driver.

Perspectives and Issues in ML

- What algorithms are exist for learning general target functions from specific training examples?
- How much training data is sufficient to learn a concept with high confidence?
- When and how can prior knowledge held by the learner guide the process of generalizing from examples?
- 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 problem?
- What is the best way for a system to represent its knowledge?

Designing A Learning System

The following steps are involved in designing a learning system:

- Choosing the training experience
- Choosing the target function
- Choosing a representation for the target function
- Choosing a function approximation algorithm
    - Estimating training values
    - Adjusting the weights
- The final design

Checkers game

Choosing the training experience (Data)

- Choose the type of training experience from which our system will learn
- The type of training experience available can have a significant impact on success or failure of the learner
- This has three attributes
  - Choosing the type of training experience: Direct or indirect training examples
  - Control the Sequence of training examples
  - How well the distribution of training examples covers future test scenarios

Choosing the target function

- To determine exactly what type of knowledge will be learned and,
- how this will be used by the performance program

Case 1: **Direct** training experience,

In this case we will have board states and corresponding legal moves, and we have to choose ***the best move*** by using the below function.

ChooseMove : B -> M

Case 2: **Indirect** training experience

consisting of the move sequences and final outcomes of various games played

Case 2: **Indirect** training experience

- We will have current board state and its corresponding numerical value.
- Numerical value is based on the chances of winning from that move.
- We need a function **V: B -> R**, to denote that V maps any legal board state from the set B to some real value.
- We intend for this target function V to assign higher scores to better board states.
- If the system can successfully learn such a target function V,then it can easily use it to select the best move from any current board position.

Cont...

Let us define the target value V(b) for an **arbitrary board state b** in B, as follows:

1. if b is a final board state that is won, then V(b) = 100
2. if b is a final board state that is lost, then V(b) = -100
3. if b is a final board state that is drawn, then V(b) = 0

4. if b is a not a final state in the game, 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

The Fourth Case

- In this case the function V will neither give 100 nor -100.
- So in this case the numerical value has to be approximated.
- This approximation function is represented as **V’**

Choosing a Representation for the Target Function

- Now, we have specified the ideal target function V.
- we must choose a representation that the learning program will use to describe the function V'
- The possible representation can be:
  - a large table with a distinct entry specifying the value for each distinct board state
  - a collection of rules that match against features of the board state
  - a quadratic polynomial function of predefined board features, or an artificial neural network.

Simple representation...

For any given board state, the function **V'** will be calculated as a linear combination of the following board features:

xl: 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

cont...

Thus, our learning program will represent $\hat{V}(b)$ as a linear function of the form

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

where ***wo*** through ***W6*** are numerical coefficients, or weights, to be chosen by the learning algorithm

Here we have reduced the problem of learning a checkers strategy to

the problem of learning values for the coefficients ***Wo*** through ***W6*** in the target function representation.

Choosing a Function Approximation Algorithm

In order to learn the target function V'
we require a set of training examples, each describing a specific board state b and the training value Vtrain(b) for b.

So each training example is an ordered pair of the form (b, $V_{train}(b)$).

For instance, the following training example describes a board state b in which black has won the game (note ***x2*** = 0 indicates that **red** has no remaining pieces) and
for which the target function value Vtrain(b)is therefore **+100.**

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

Cont….

The procedure has two steps:

- Estimating the training values
- Adjust the weights that best fit the training examples

Estimating the training values

- **The only training information available to our learner is whether the game was eventually won or lost**
- **we require training examples that assign specific scores to specific board states**
- **Now, we need to know how to assign training values to the more *intermediate* board states**

Rule for estimating training values.

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

-- assign the training value of $V_{train}(b)$ for any intermediate board state ***b*** to be ***V'(Successor(b))***

-- where ***Successor(b)*** denotes the next board state

Adjusting the weights

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

- It computes the error value between expected value(Vtrain(b)) and actual value(V'(b))
- If the difference value is large then we need adjust ensure that the error is minimized by adjusting the weights
- Based on the error value we adjust the weights to minimize the errors to reach expected results.

LMS weight update rule

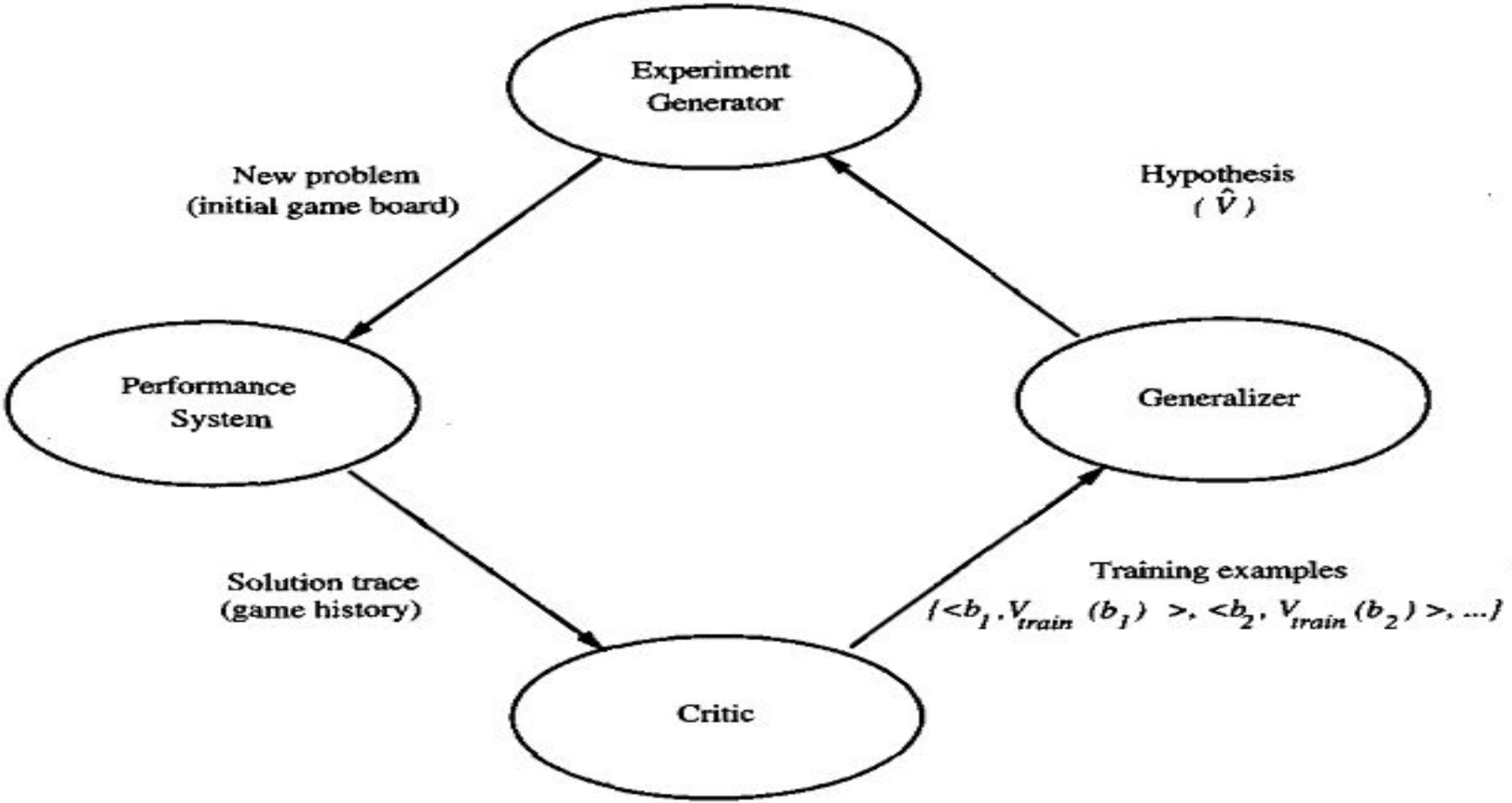
For each training example $\langle b, V_{train}(b) \rangle$

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

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

- **Here n is learning rate is a fractional value to ensure that the weight does not change drastically**
- **Its value is 0.01**

The final design



The **Performance System**

- It is the module that must solve the given performance task, in this case playing checkers, by using the learned target function(s)
- Strategy used by the Performance System to select its next move at each step is determined by the learned **V'** evaluation function

Critic

- Takes as input the history of the game and
- Produces as output a set of training examples of the target function.
- Each training example in this case is V_{train}

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

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

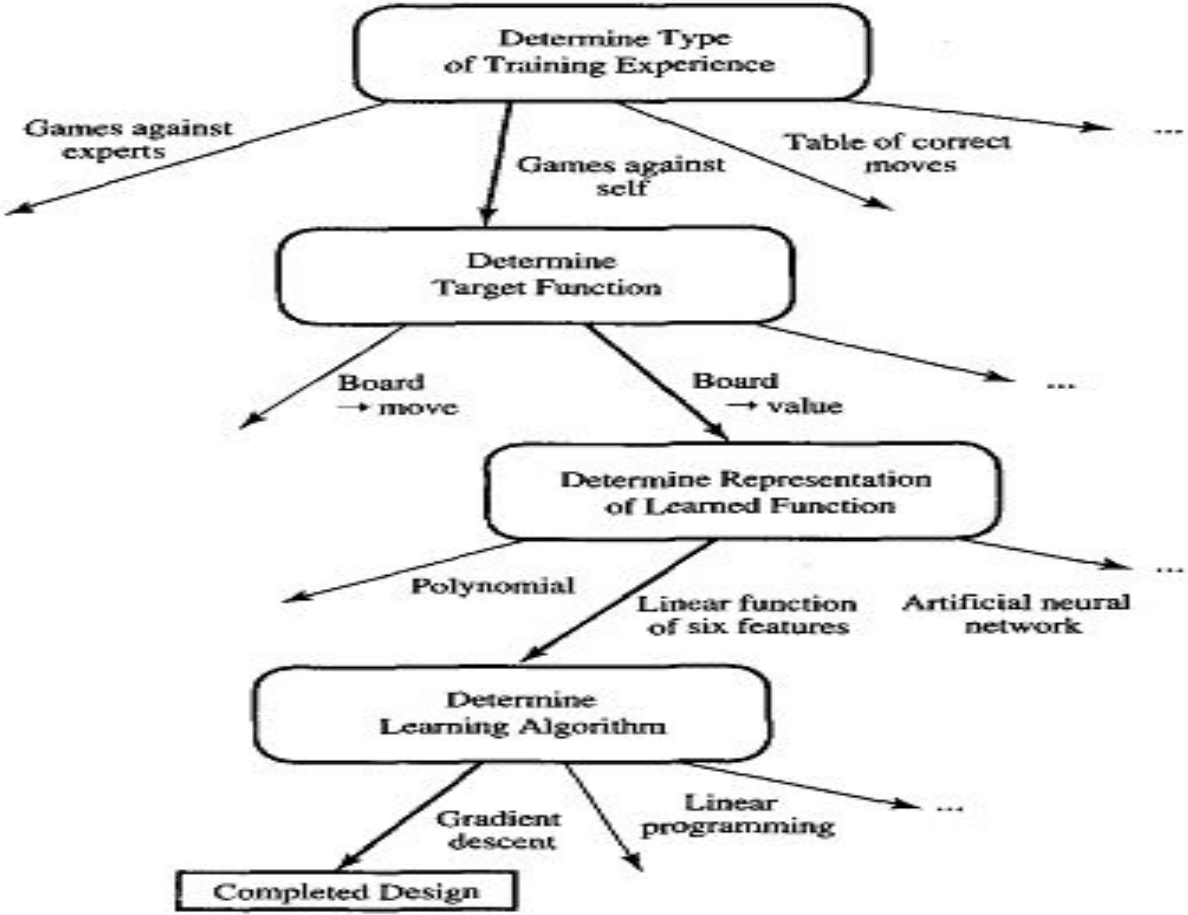


FIGURE 1.2
Summary of choices in designing the checkers learning program.

Thank You