A CHECKERS LEARNING PROBLEM

"Machine Learning" By Tom Mitchell

PROBLEM

- Task T: playing checkers
- Performance measure ***P:*** percent of games won in the world tournament
- Training experience E: games played against itself

APPROACH

1. The exact type of knowledge to be learned
2. A representation for this target knowledge
3. A learning mechanism

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

TARGET FUNCTION

- to reduce the problem of improving performance P at task T to the problem of learning some particular ***target function***

Legal Moves	Won or Lost	← Indirect Training Experience
Given Broad	Moves required to win	← Direct Training Experience

TARGET FUNCTION

- Evaluation function that assigns a numerical score to any given board state.
- ***V*** : B →R to denote that ***V*** maps any legal board state from the set B to some real value (we use R to denote the set of real numbers).

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 (assuming the opponent plays optimally, as well).

TARGET FUNCTION

- ***operational description of the ideal target function*** V is required.
- Learning algorithms is expected to acquire only some ***approximation*** to the target function, and for this reason the process of learning the target function is often called ***function approximation***

On one hand, we wish to pick a very expressive representation to allow representing as close an approximation as possible to the ideal target function V. On the other hand, the more expressive the representation, the more training data the program will require in order to choose among the alternative hypotheses it can represent.

Problem Representation

A simple representation: for any given board state, the function $\hat{V}$ 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

TARGET FUNCTION

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

$$\hat{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

where w_0 through w_6 are numerical coefficients, or weights, to be chosen by the learning algorithm. Learned values for the weights w_1 through w_6 will determine the relative importance of the various board features in determining the value of the board, whereas the weight w_0 will provide an additive constant to the board value.

ESTIMATING TRAINING VALUES

- In order to learn the target function $\hat{V}$ we require a set of training examples, each describing a specific board state b and the training value $V_{train}(b)$ for b. In other words, each training example is an ordered pair of the form $\langle b, V_{train}(b) \rangle$.
- **Rule for estimating training values.**

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

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 $\hat{V}$.
- $E = \sum_{\langle b, V_{train}(b) \rangle \,\epsilon\, training\ sample} (V_{train}(b) - \hat{V}(b))^2$

Thus, we seek the weights, or equivalently the $\hat{V}$, that minimize E for the observed training examples.

LMS Training

Least mean squares or **LMS** training rule is one of several algorithms to incrementally refine the weights.

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**

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

The Final Design