Introduction to Machine Learning

Well-Posed Learning Problems

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 E.

For example, for a handwriting recognition problem:

1. Task T : Recognizing and classifying handwritten words within images
2. Performance measure P : Percent of words correctly classified
3. Training Experience E : Database of handwritten words with correct classification

Designing a Learning System

1. **Choosing the Training Experience**
    a. Type of training experience to be chosen
        i. Does it provide direct feedback or indirect feedback?
        ii. How will the credit assignment happen?
    b. Choose Degree to which the learner controls the sequence of training examples
    c. Examine how well the training experience represents the distribution of examples
        i. Useful experience only when training examples follow the same distribution as future examples
2. **Choosing the Target Function**
    a. What type of knowledge needs to be learnt
    b. How will the program use this knowledge
    c. Choosing from legal moves in the search space
    d. Function approximation is used to find an operational description of the ideal target function
3. **Choosing Representation for the Target Function**
    a. Rule-based representation or Table-based representation, for example
    b. Tradeoff between excessive representation and approximation to variable data needs to be considered. More expressive the representation, the more data you need to fit that representation.

4. **Choosing a Function Approximation Algorithm**
    a. **Estimating Training Values**
        i. Assign values to intermediate states that the model visits
        ii. Rule for estimating training values is summarized as

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

    b. **Adjusting the Weights**
        i. The best hypothesis is defined as the hypothesis that best fits the data. This usually is selected as the hypothesis has the least squared error E between the training values and the values predicted.

$$E \equiv \sum_{<b,\, V_{train}(b)> \in \, training\ examples} \left(V_{train}(b) - \hat{V}(b) \right)^2$$

        ii. The Least Mean Square rule allows the update on the errors to minimize the least squared error.
        iii. LMS Weight Update Rule

        For each training example,

            1. Use current weights to calculate V'(b)
            2. For each weight w, update w as

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

5. **Final Design**
    a. Performance System
        i. Module that solves the performance task using the target function
        ii. Takes a new problem as input and provides a trace of its solution as output
    b. Critic
        i. Takes an input the history or trace of the game and provides output as a set of training examples of the target function
    c. Generaliser
        i. Takes training examples as input and produces output that is its estimate of the target function.
        ii. Corresponds to the LMS algorithm and produces the learned weights
    d. Experiment Generator
        i. Takes the current hypothesis as input and outputs a new problem
        ii. Chooses problems that maximises the learning rate of the overall system

Perspective in Machine Learning

Involves searching a very large space of possible hypotheses to determine one that fits the observed data the best.

The LMS algorithm achieves this by correcting the weights each time the predicted value by a hypothesis differs from the actual value. This works well when the hypothesis representation considered by the learner defines a continuously parameterised space of potential hypotheses.