Chapter 1
Introduction

1.1 WELL-POSED LEARNING PROBLEMS

- **Definition:** A computer is said to learn from experience "E" with respect to some class of tasks "T" and performance measure "P", if its performance at tasks T, as measured by P, improves with experience E.
- **Ex.**: Handwriting recognition problem:
  - **Task T:** Recognizing and classifying handwritten words within images.
  - **Performance measure P:** Percentage of words correctly classified.
  - **Training experience E:** A database of handwritten words with given classifications.

1.2 DESIGNING A LEARNING SYSTEM

- **Choosing the Training Experience:**
  - One key attribute is whether the training experience provides direct or indirect regarding the choices made by the performance system. For example, if we're building an autonomous vehicle to drive through a specific type of roads, let's say highways, the system can learn from:
    1. ***Direct*** training examples consisting of individual video frames accompanied with the suitable steering angles for each frame.
    2. Alternatively, we made have only ***indirect*** information consisting of the steering angles sequence and the final outcome of several trips through the road (Finished/didn't finish the trip). Here the learning algorithm must determine the degree to which each move in the given sequence deserves credit or blame for the final outcome (credit assignment).

    NOTE: Learning from *direct* training feedback is typically easier than learning from *indirect* feedback.

- Another attribute is the degree to which the learning algorithm controls the sequence of training examples.
  1. The learning algorithm might rely on the designer (teacher) to select informative information about the environment surrounding the vehicle and to provide the correct move for each situation.
  2. The learning algorithm might itself propose specific environment settings that it finds difficult and ask the designer for the correct moves.
  3. The learning algorithm might have complete control over the track map and *indirect* classifications, and then it credits or discredits each move it takes depending on its output.
- A third attribute of the training experience is how well it represents the distribution of examples over which the final system performance "P" must be measured. In general, the learning algorithm will be most reliable if the training examples follow a distribution like the future test example. For instance, if our training examples only include steering angles for straight roads (without any curves), there's a high probability that the learning algorithm will fail to "generalize" to roads with curves. Simply put, it doesn't know how deal with curves.
- Now our autonomous driving problem might be described as:
  1. Task "T": Driving through highways.
  2. Performance measure "P": Percentage of "Finished trips" in a 50km highway.
  3. Training experience "E": Several sequences of steering angles for several trips (Finished successfully + not finished).
- In order to complete the design, we must choose:
  1. The exact type of knowledge to be learned.
  2. A representation for this target knowledge.
  3. A learning mechanism.

- **Choosing the Target Function:**
  - We need to determine exactly what type of knowledge will be learned and how this will be used by the algorithm.
  - Sticking to our autonomous vehicle example, the learning algorithm needs only to learn how to choose the best steering wheel angle for every point on the road from among the range of possible angles, depending on the sequence of angles that brought us to this point.
  - Let's call this function *ChooseAngle* and use the notation:

    $$ChooseAngle: S \rightarrow A$$

    To indicate that the function accepts as input a sequence of angles (S) produces an output angle from the set of possible angles (A).

    NOTE: It's always useful to reduce the problem of improving performance "P" at task "T" to the problem of learning some particular target function such as *ChooseAngle*.
  - Our target function will assign a numerical score to any given sequence of steering angles. Let's call this target function *V*. We intend for the target function *V* to assign higher scores for better sequences of steering angles.
  - If the algorithm can successfully learn such function, it can select the best angle from any current sequence of angles. This can be accomplished by generating each possible steering angle for next point on the road, then using *V* to choose the best steering sequence and therefore the best angle.

- Let's define the value of the target function *V* for an arbitrary point s in S, as follows:
  1. If *s* is the final sequence of angles and the road is finished, then *V(s) = +1000*.
  2. If *s* is the final sequence of angles the road is not finished, then *V(s) = 0*.
  3. If *s* is the final sequence of angles and the vehicle steered off the road, then *V(s) = -1000*.
  4. If *s* is not the final sequence of angles, then *V(s) = V(s')*, where *s'* is the best sequence of steering angles starting from *s* and continuing optimally till the end of the road.
- While this recursive definition specifies a value of *V(s)* for every sequence *s*, it's not efficiently computable, because determining the value of *V(s)* in case no. 4 requires searching for the best sequence of steering angles all the way to the end of the road. This is called a *nonoperational definition*.
- The goal of learning in this case is to discover an operational description of *V*, one that can be used to select angles within realistic time bounds. Now, we have reduced our task to the problem of selecting an *operational description of the target function V*.
- The learning algorithms often acquires only an approximation for this ideal target function (*V'*), therefore the learning process is called *function approximation*.

- **Choosing a Representation for the Target Function:**
  - The selection of a representation function (*V'*) involves a crucial trade-off:
    1. On one hand, picking a very expressive representation will allow producing a very close approximation to the ideal target function *V*.
    2. On the other hand, the more expressive the representation, the more training data it will require.
  - We'll chose a simple representation, for any given sequence of steering angles, *V'* will be calculated as a linear combination of the following features:
    1. x_1: The number of times steering wheel angle was between 10° and 180° to the right.
    2. x_2: The number of times steering wheel angle was between 180° and 360° to the right.
    3. x_3: The number of times steering wheel angle was between 10° and 180° to the left.
    4. x_4: The number of times steering wheel angle was between 180° and 360° to the left.
    5. x_5: The number of times the vehicle had to stop completely to adjust its path to the right one.
    6. x_6: The total distance finished till now.
  - Our learning algorithm will represent *V'(s)* as a linear function of the form:

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

    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 each of these factors in determining the value of the sequence of angles. And w_0 will provide an additive constant to the value of the sequence of angles.
- **Choosing a Function Approximation Algorithm:**
  - Each training example should be in an ordered pair of the form $(s, V_{train}(s))$. For example:

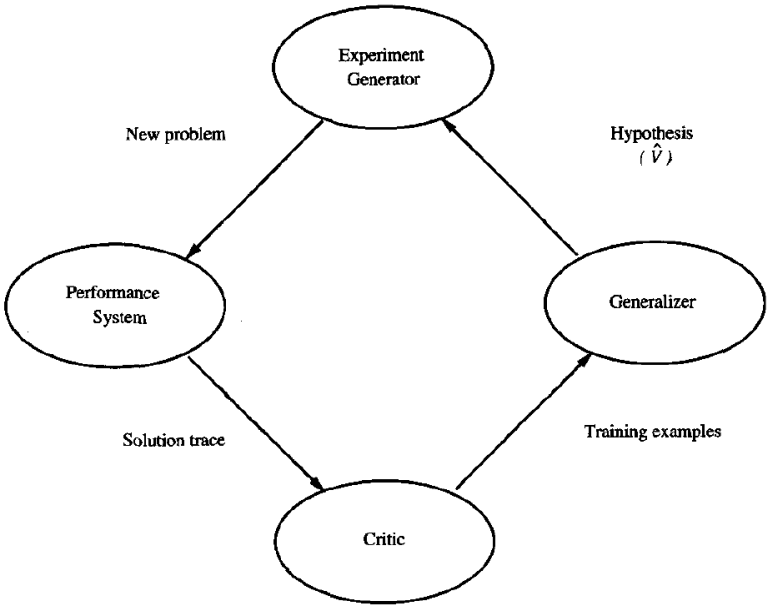
    $$((x1 = 50, x2 = 22, x3 = 52, x4 = 17, x5 = 2, x6 = 50), +1000)$$

- Estimating Training Values:
  a) According to our definition of the autonomous driving problem, the only training information we have is whether the vehicle finished the road or not using the given sequence of steering angles. The problem now is how to assign values for the different factors (x_1 .. x_6) *that logically corresponds to the end result?*
  b) One approach that proved to be successful is to assign the training value of $V_{train}(s)$ for any intermediate sequence of steering angles to be $V'(Successor(s))$ where V' is the learner's current approximation of V, and $Successor(s)$ is the next sequence of angles following s.
  c) This approach is called *"Reinforcement Learning"*, and it proved to be very successful, especially if V' tend to be more accurate towards the end of the road.
  NOTE: Reinforcement Learning will be discussed in detail in chapter 13.
- Adjusting the Weights:
  a) We need to define the set of weights which minimizes the squared error E between the training values and the values predicted by the hypothesis V'.

$$E = \sum (V_{train}(s) - V'(s))^2$$

  b) One algorithm that can find this value is the Least Mean Squares, or LMS training rule. For each observed training example, it adjusts the weights a small amount in the direction that reduces the error on this training example.

- **The Final Design:**
  The final design of the autonomous driving learning algorithm can be described by four modules:
  - The Performance System: This is the module that must solve the given performance task. It takes an instance of a new problem (new road) and produces a trace of its solution (sequence of steering angles) as output.
  - The Critic: Takes as input the trace of the road and produces a set of training examples of the target function.
  - The Generalizer: Takes as input the training examples and produces an output hypothesis that is its estimate of the target function.
  - The Experiment Generator: Takes as input the current hypothesis and outputs a new problem for the Performance System to explore.



1.3 PERPECTIVES IN MACHINE LEARNING:

- Machine Learning involves searching a very large space of possible hypothesis to determine one that best fits the observed data and any prior knowledge held by the learner.