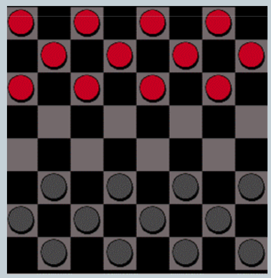
Formulation of Machine Learning Problems

MACHINE LEARNING 09/10



Well Posed Learning Problems

Learning = Improving with experience at some task.

- Improve over task *T*.
- With respect to performance measure *P*.
- Based on experience *E*.

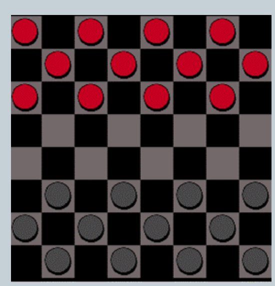
What are *T*, *P*, *E* ? How do we formulate a machine learning problem ?

Examples

- Checkers Learning
  - *T* – play checkers
  - *P* – percentage of games won in the world tournament.
  - *E* - train with peers.
- Handwriting recognition
  - *T* – classifying handwritten words within images.
  - *P* – percent of words correctly classified.
  - *E* – database of handwritten words with given classifications.
- Robot Driving
  - T – Driving on public four-lane highways using vision sensors.
  - P – Average distance traveled before an error (human supervisor).
  - E – sequences of images and steering commands recorded observing a human driver.

Case Study: Learning to Play Checkers

- What experience ?
  - Train with peers, with self, have lessons, read book, google
- What exactly should be learned ?
- How shall it be represented ?
- What specific algorithm to learn it ?



What Experience ?

- How training experience influences performance goal ?
  - Type of feedback: Direct vs Indirect.
  - Learning strategy: Have a teacher or not ? Exploration vs Exploitation?
  - Diversity of training: Is the training data representative of the task ? How many peers should we play with ? How many tactics should we try when playing with self.
- Let us decide that our program will learn by playing with itself and formulate the learning problem.

What Exactly Should be Learned ?

- For each board state our system must choose the best move among the legal ones.
  - Learn a function that maps board states to good moves.

$$ChooseMove : BoardState \rightarrow Move$$

  - Learn a function that assigns values to board states.

$$BoardValue : BoardState \rightarrow \Re$$

- What is the best one ?

Choosing the Target Function

- A possible definition is:

  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 not a final board state, 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 oponent play optimal as well.)

- This gives correct results but is not operational. Case 4 is too hard to compute. We need to approximate the value function, instead of using the optimal one.

$$V'(b) \sim V(b)$$

Representing the input space

- The space of board states is huge – there trillions possible configurationson a checkers board.
- Should select features from the board state that somehow represent it in an adequate and succinct manner for the problem at hand.
  - x_1 – the number of black pieces in the board
  - x_2 – the number of red pieces on the board
  - x_3 – the number of black kings on the board
  - x_4 – the number of red king on the board
  - x_5 – the number of black pieces threatened by red (i.e. which can be captured on red's next turn)
  - x_6 – the number of red pieces threatened by black.

What approximation to the Target Function ?

- How shall we represent in an operational manner the value of the board as a function of the board features:
  - Collection of rules ?
  - Artificial neural network ?
  - Polinomial function of board features ?
- Let us choose a first order polinomial function:

$$V'(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5$$

- Weights w_o to w_6 are to be chosen by the learning algorithm.

Checkers learning partial design

- Task T: Playing checkers
- Performance measure P: percent of games won in the world tournament.
- Training experience E: games played against itself.
- Target function: $V : Board \rightarrow \Re$
- Target function representation:

$$V'(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_5 x_5$$

We reduce the problem of learning checkers strategy to the problem of learning values for the coefficients of the weights in the target function representation.

Obtaining Training Examples

- True target function: $V(b)$
- The learned function: $\hat{V}(b)$
- The training value: $V_{train}(b)$

- We only know the value of the board in the final state. How to obtain training values for the intermediate states?

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

- It can be proved that, in certain conditions, V_{train} converges to perfect estimates.

Adjusting the Weights

- Specify the learning algorithm for choosing weights w_i to best fit the set of training examples:

$$\langle b, V_{train}(b) \rangle$$

- Minimize the squared error:

$$E \equiv \sum_{\langle b, V_{train}(b) \rangle} \left(\hat{V}(b) - V_{train}(b) \right)^2$$

- Using optimization techniques, this results in the LMS (least mean squares) weight update rule.

LMS Weight Update Rule

Do repeatedly:

1. Select a training example at random. $\langle b, V_{train}(b) \rangle$
2. Compute the approximation error:

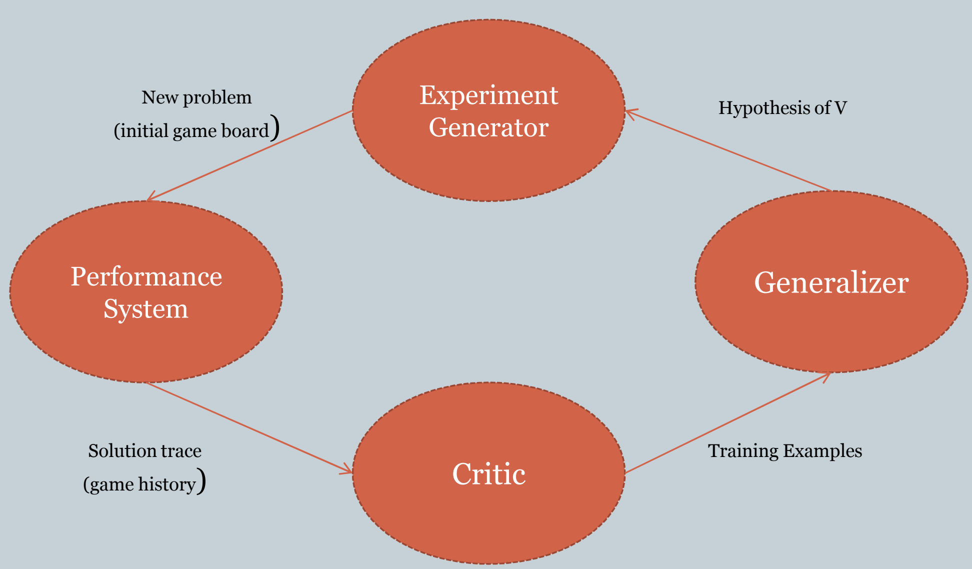
$$error(b) = \hat{V}(b) - V_{train}(b)$$

1. For each board feature x_i update weight w_i:

$$w_i \leftarrow w_i - c \cdot x_i \cdot error(b)$$

c is some small constant, say 0.1, to moderate the learing rate.

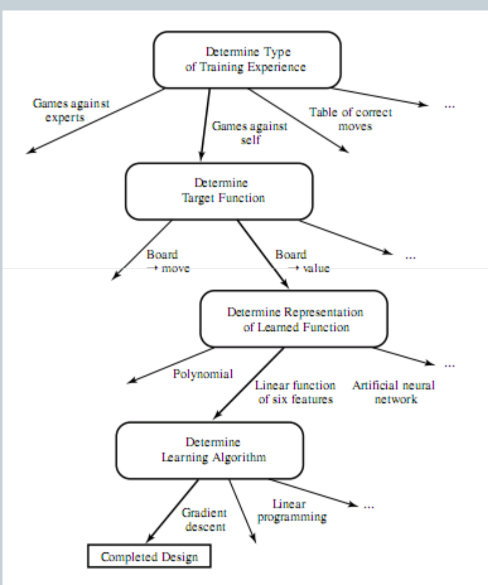
The Final Checkers Learning System



Could we beat the world champion ?

- Things yet to study:
  - What algorithms can approximate functions well (and when)?
  - How does number of training examples influence accuracy?
  - How can prior knowledge of learner help?
  - What specific function should the system attempt to learn?
  - How does noisy data influence accuracy?
  - What are the theoretical limits of learnability?
  - What clues can we get from biological learning systems?
  - How can systems alter their own representations?

Summary of Design Choices



- Most of the content of this course look into the techniques to address the last two modules:
  - Representations of the learned function.
  - Determine the learning algorithm.
- The first two blocks are application dependent. Anyway, through the laboratory exercises we will gain valuable experimental knowledge to address them.
  - Determine type of training experience.
  - Determine target function.