Machine Learning

Tom M. Mitchell, McGraw Hill

ISBN: 7-111-11502-3

1 Introduction

What is Machine Learning

- What is Machine Learning

A Computer program can improve its performance automatically with experience

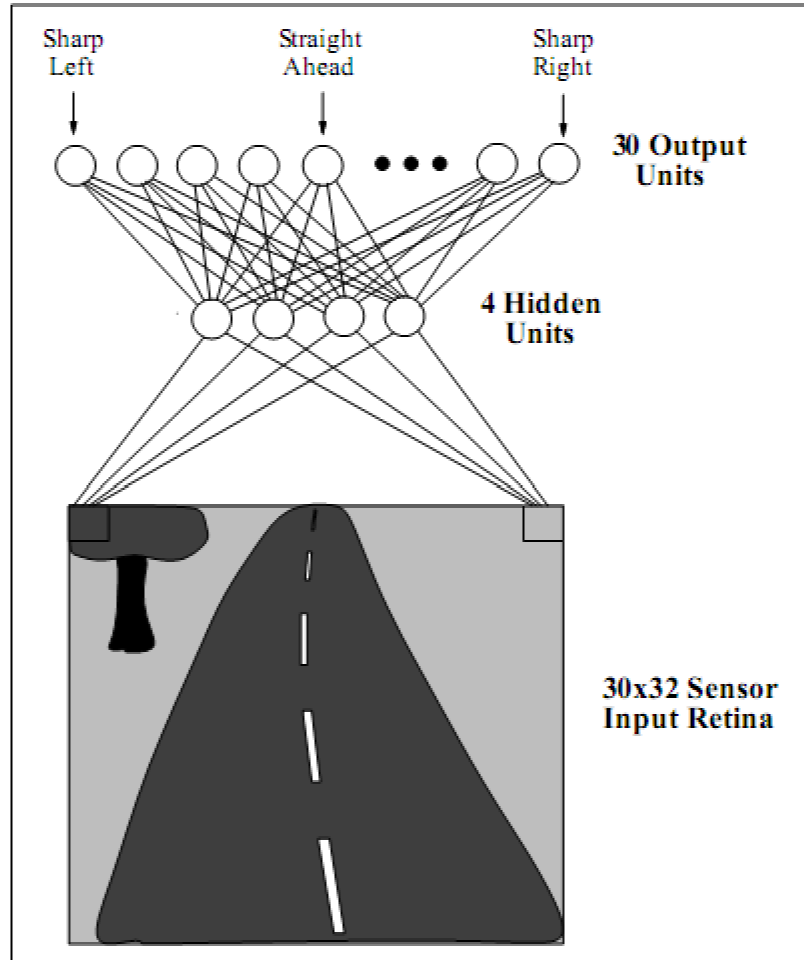
Applications of ML

- Learning to recognize spoken words
  - SPHINX (Lee 1989)
  - Apple Siri
- Learning to classify celestial objects
  - (Fayyad et al 1995)
- Learning to play world-class backgammon
  - TD-GAMMON (Tesauro 1992)
  - Deep Blue (a chess-playing computer developed by IBM,1997)
  - AlphoGo（2016，Google，Deepmind， Nature，Mastering the game of Go without human knowledge ）

Applications of ML

- Learning to drive an autonomous vehicle
  - ALVINN (Pomerleau 1989)
  - Google's self-driving cars (2005-Now)
    - driven more than 500,000 miles,40 kilometers per hour, drive, brake and recognize road dangers

- ALVINN (Pomerleau 1989) Sample



Disciplines relevant to ML

- Artificial intelligence
- Probility and Statistics
- Information theory
- Computational complexity theory
- Philosophy
- Psychology and neurobiology

1.1 Well-Posed Learning Problems

- Learning Definition
  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

Learning: improving with experience at some task

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

Example:

- A checkers learning problem:
  - T: play checkers
  - P: percentage of games won in a tournament
  - E: opportunity to play against itself

- Handwriting recognition learning problem
- A robot driving learning problem

- Definition of learning is broad
  - Encompass computer programs that improve from experience in quite straightforward ways
  - “Learning” we care:
    - Define learning problem
    - Explore algorithms that solve such problems
    - Understand the fundamental structure of learning problems and processes

1.2 Designing a Learning System

- Basic Design Method
  - Choose the training experience
  - Choose the target Function
  - Choose a representation for the target function
  - Choosing a function approximation Algorithm
  - The final design

1.2.1 Choosing the Training Experience

- Choose the type of experience
  - Attribute 1:Whether the training experience provides direct or indirect feedback regarding the choices made by the performance system
  - Attribute 2:The degree to which the learner controls the sequence of training examples
  - Attribute 3:How well the training samples represents the distribution of examples over which the final system performance P must be measured

- A Checkers learning problem

  Task T: play checkers

  Performance P: percentage of games won

  Train experience E: opportunity to play against itself

- Next System should choose
  - The exact type of knowledge to be learned
  - A representation for this target knowledge
  - a learning mechanism

1.2.2 Choosing the Target Function

- The next design
  - Determine knowledge type and how program uses it

- Checker program
  - choose the best move from all legal moves
    - Representative of a class of optimization problems
    - Search space are know, the best search strategy is not known

- Determine knowledge type of Checker Learning
  - Learn to choose among legal moves
  - One choice is learning a function

- Target Function 1 : ChooseMove
  - ChooseMove: B→M，(board state → move)
  - Maps a legal board state to a legal move
  - difficicult in this case, why?

- Target Function 2 (function V):
  - V : B→R : board state → board value
  - Assigns a numerical score to any given board state, such that better board states obtain a higher score
  - Select the best move by evaluating all successor states of legal moves and pick the one with the maximal score

- Possible Definition of Target Function 2
  - If b is a final board state that is won then V(b) = 100 (notice: two players)
  - If b is a final board state that is lost then V(b) = -100
  - If b is a final board state that is drawn then V(b)=0
  - 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.
  - This function gives correct values but is not operational

- Learning Task :
  - discovering an operational description of the ideal target function V
  - Difficult, but approximate it is enough
  - Learning process is a process of function approximation

1.2.3 Choosing a Representation for the Target Function

- Function Representation
  - Table look-up
  - Collection of rules
  - Neural networks
  - Polynomial function of board features
- We need to consider the trade-off between
  - Expressive power(Approximation accuracy)
  - Number of training examples

- $\hat{V}$ is a linear combination of the following board features
  - x1， 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
  - x6， the number of red pieces threatened by black

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

1.2.4 Choosing a function Approximation Algorithm

- Each training example is an ordered pair of the form $< b, \ \mathrm{V}_{\mathrm{train}}(b) >$
  - b: board state， $V_{train}(b)$: training value
  - Sample: $<< x_1 = 3, \mathrm{x}_2 = 0, \mathrm{x}_3 = 1, \mathrm{x}_4 = 0, \mathrm{x}_5 = 0, \mathrm{x}_6 = 0 >, +100)$
- Training Procedure:
  - Derive training samples from the indirect training experience available to the learner
  - Learning the weights to best fit those training examples

- Estimating training value

  - Difficult to assign training values to intermediate board states

  - One simple approach:

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

- Adjust the weights (LMS weight update rule):

$$E = \sum_{<b, V_{train}(b)> \in trainig \;\; examples} (V_{train}(b) - \hat{V}(b))^2$$

Select a training example $< b, \; \mathrm{V}_{\mathrm{train}}(b) >$ at random

1. Use the current value to calculate $\hat{V}(b)$
2. Compute error(b)

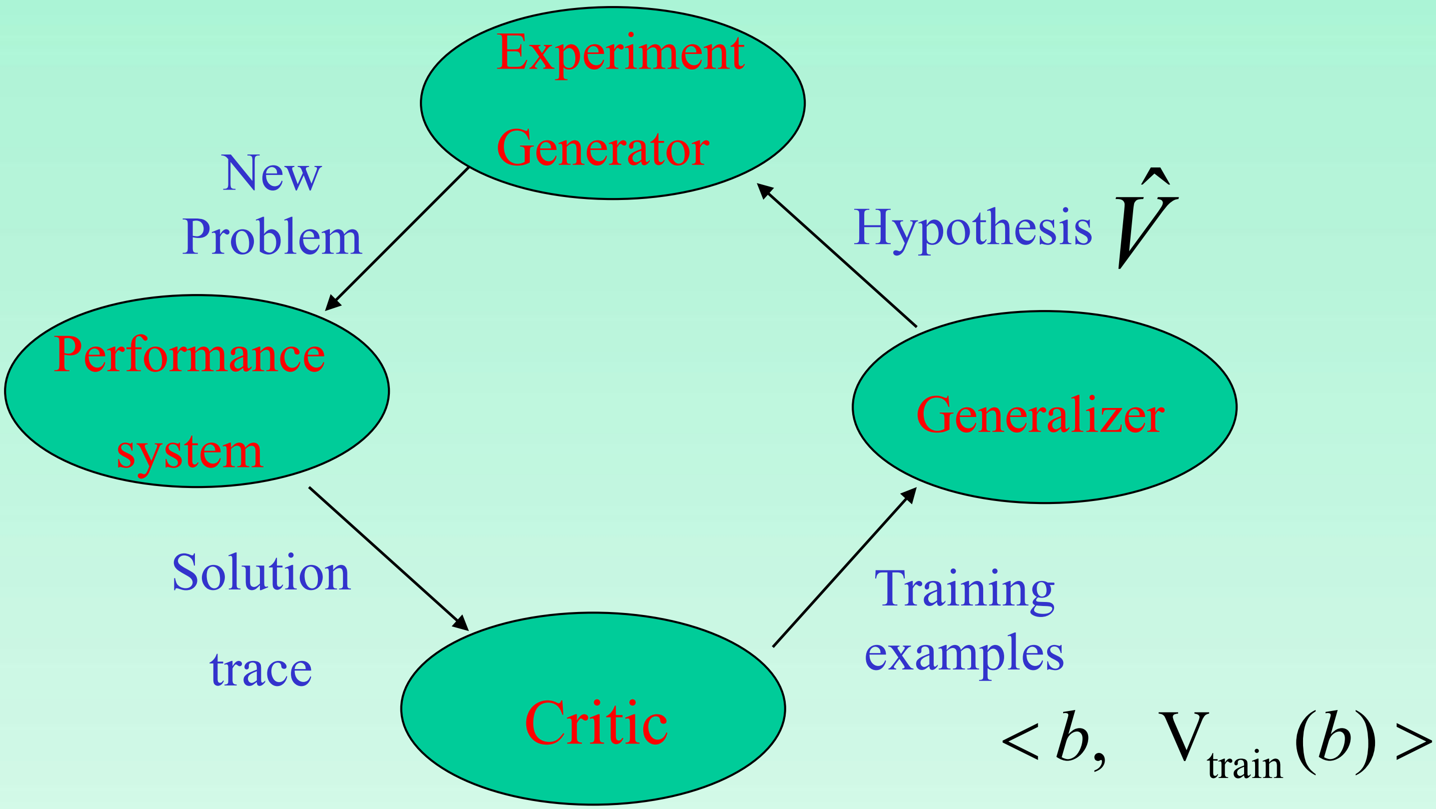
   error(b) = $\mathrm{V}_{\mathrm{train}}$(b) – $\hat{V}(b)$

3. For each board feature $\mathbf{X}_i$,

   update weight $\omega_i \leftarrow \omega_i + \eta$ error(b)$\mathbf{X}_i$

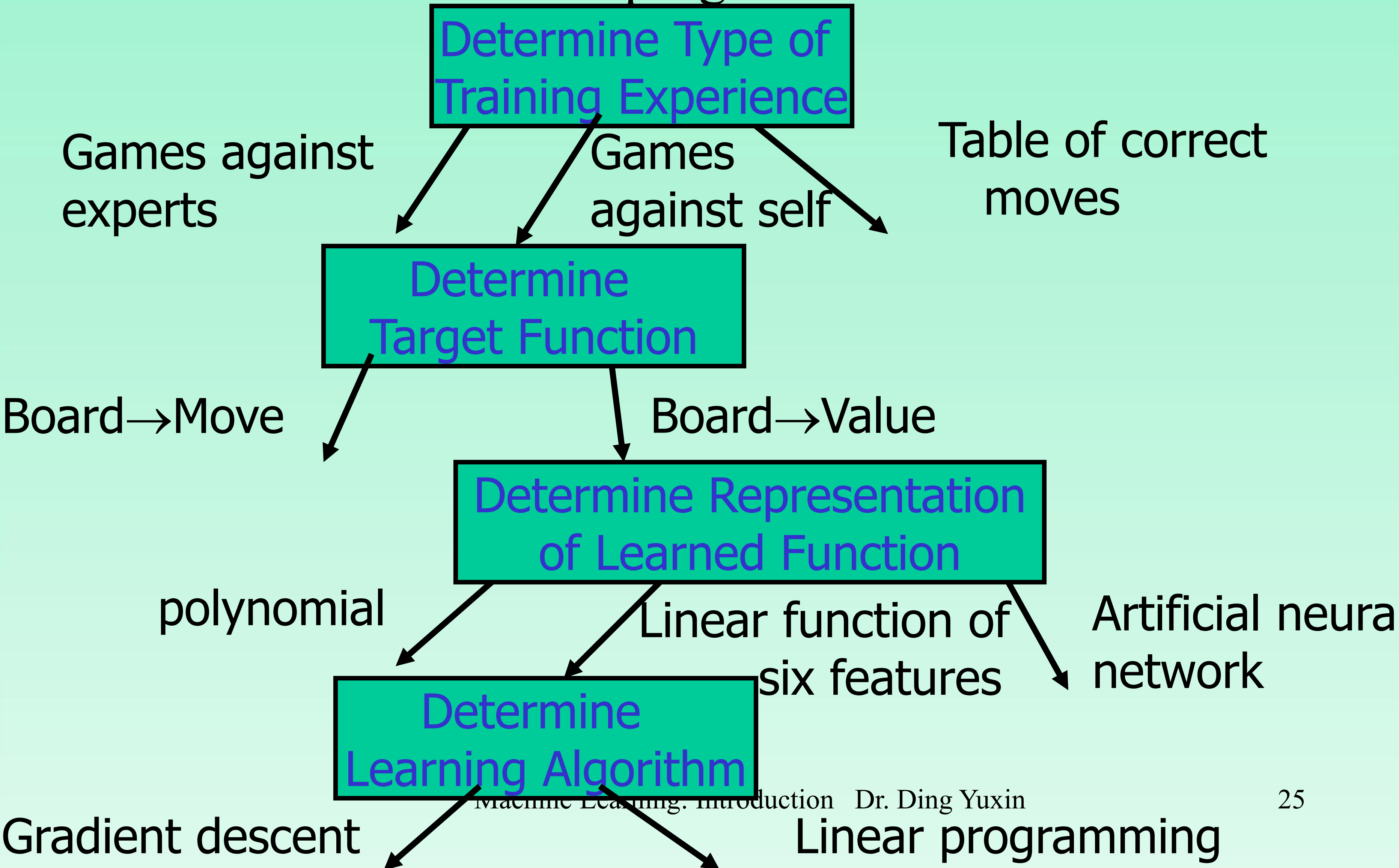
   η : learning rate approx. 0.1

1.2.5 Final Design



- The Performance System
  - Solve the given task by using the learned target functions
- The Critic
  - Take as input the history or trace of the game and produce as output a set of training examples of the target function
- The Generalizer
  - Take as input the training examples and produces an output hypothesis that is its estimate of the target function
- The Experiment Generator
  - Take as input the current hypothesis and outputs a new problem for the Performance system to explore

Summary of the design of the checkers learning program



- Whether guarantee to find a good approximation?
  - The True V is in this form
  - Chapter 13 provides a theoretical analysis showing that this approach converge to the desired evaluation function

- Use a more sophisticated representation for the target function

- Other algorithms
  - Nearest neighbor
  - Genetic algorithms
  - Explain-based learning

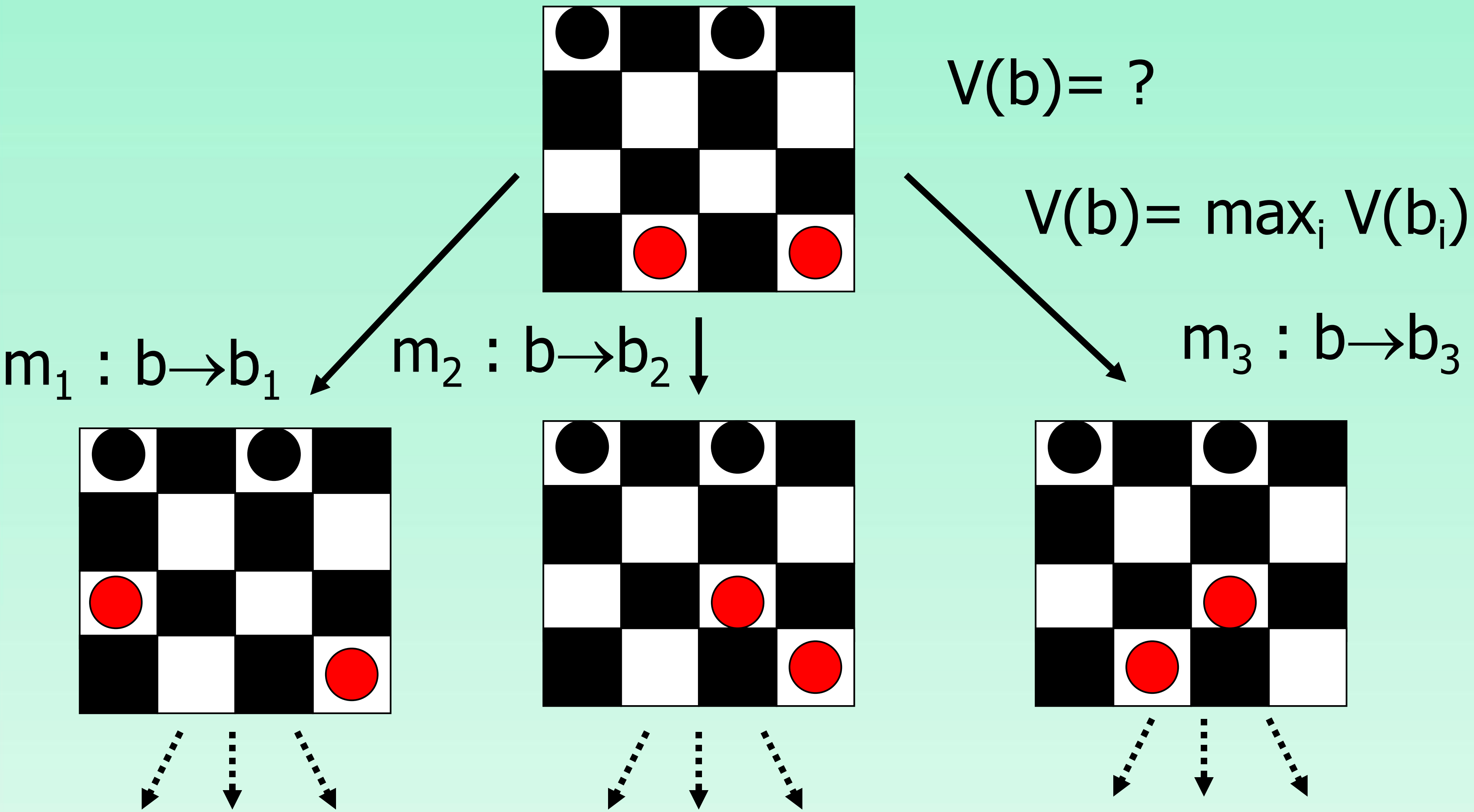
1.3 Perspectives and Issues In Machine Learning

- One useful perspective on ML
  - searching a very large space of possible hypotheses
- This book presents algorithms that search hypothesis space defined by some underlying representation
- Learning methods are characterized by their search strategies and by the underlying structure of the search spaces they explore

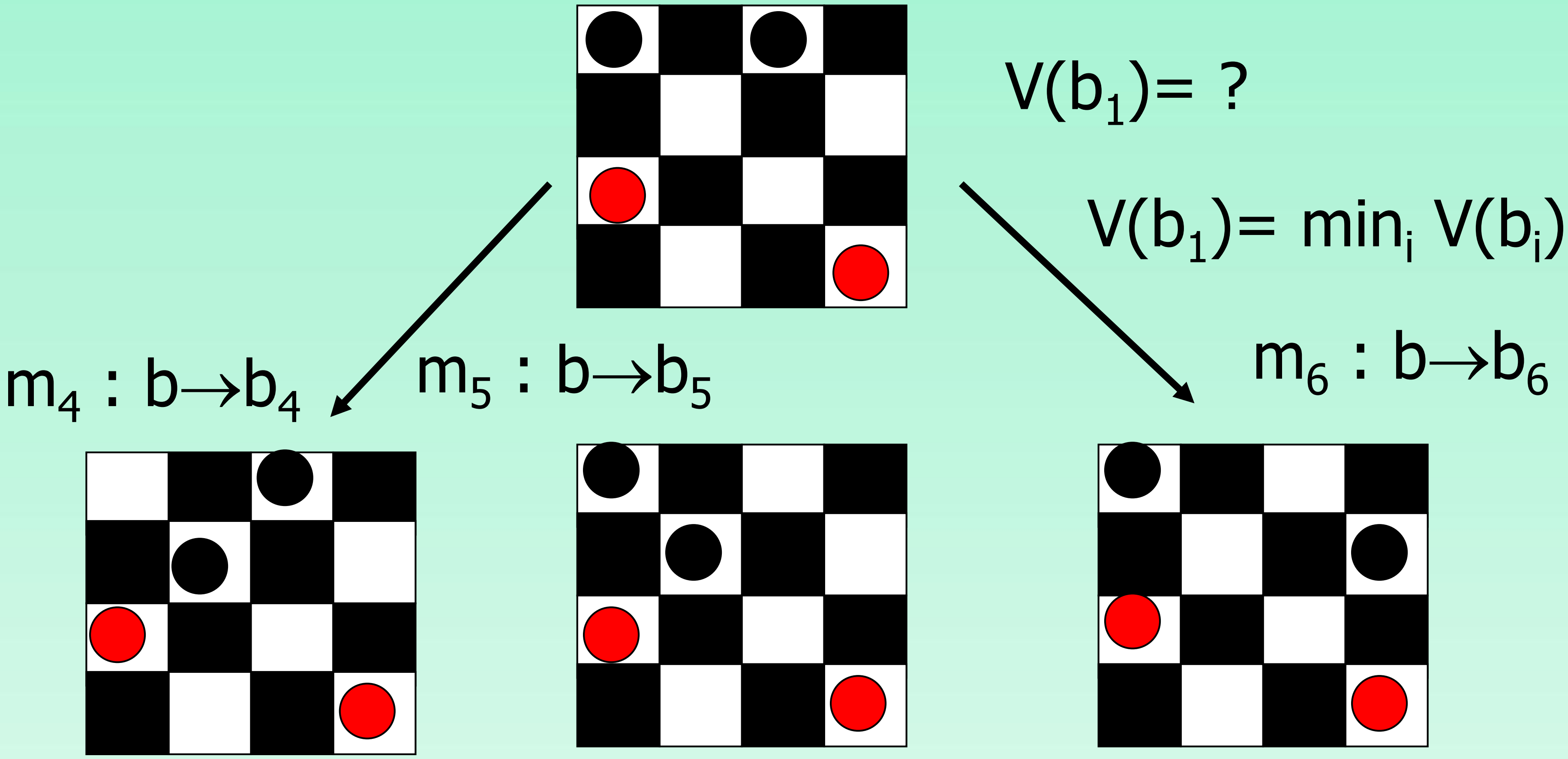
1.3.1 Issues in Machine Learning

1.4 How To Read This Book

State Space Search

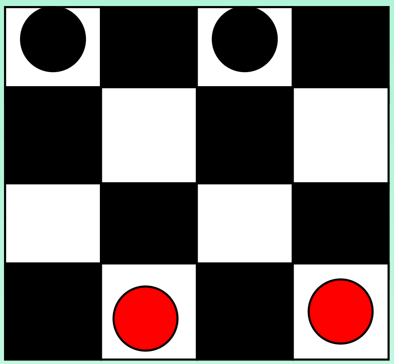


State Space Search



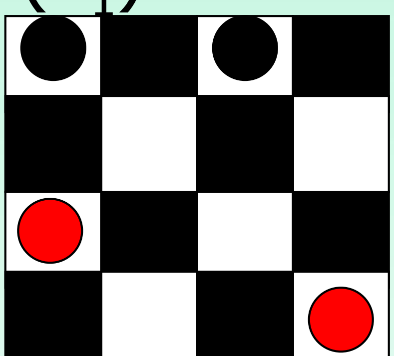
Example: 4x4 checkers

- $V(b)=\omega 0 + \omega 1 rp(b) + \omega 2 bp(b)$
- Initial weights: $\omega 0=-10$, $\omega 1 =75$, $\omega 2 =-60$

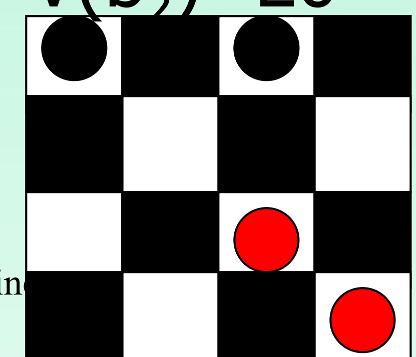


$$V(b_0)=\omega_0 + \omega_1{*}2 + \omega_2{*}2 = 20$$

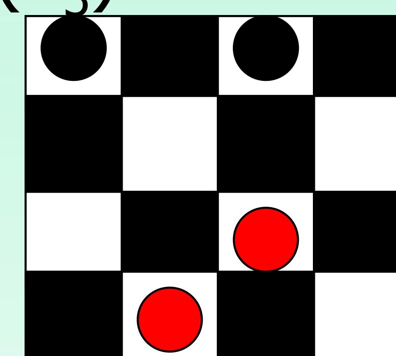
$m_1 : b \rightarrow b_1$
$V(b_1)=20$



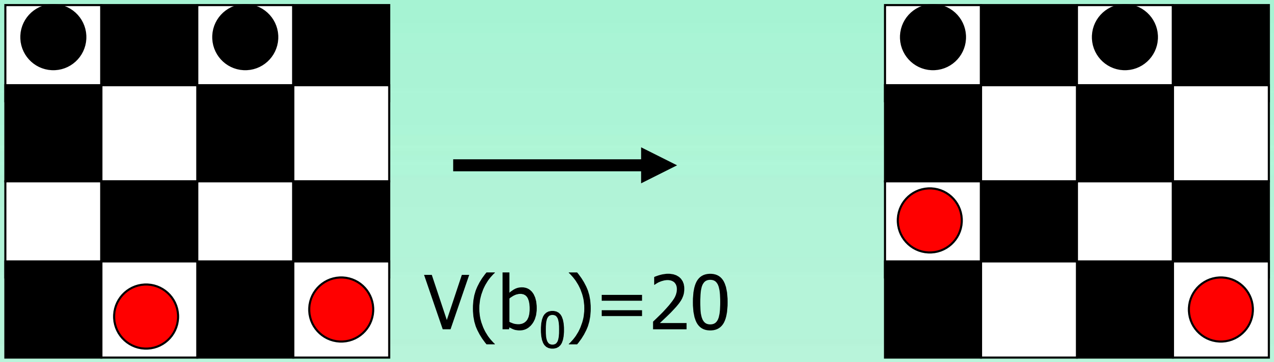
$m_2 : b \rightarrow b_2$
$V(b_2)=20$



$m_3 : b \rightarrow b_3$
$V(b_3)=20$



Example 4x4 checkers



$V(b_0)=20$ $V(b_1)=20$

1. Compute $error(b_0) = V_{train}(b) - V(b_0) = V(b_1) - V(b_0) = 0$
2. For each board feature fi, update weight

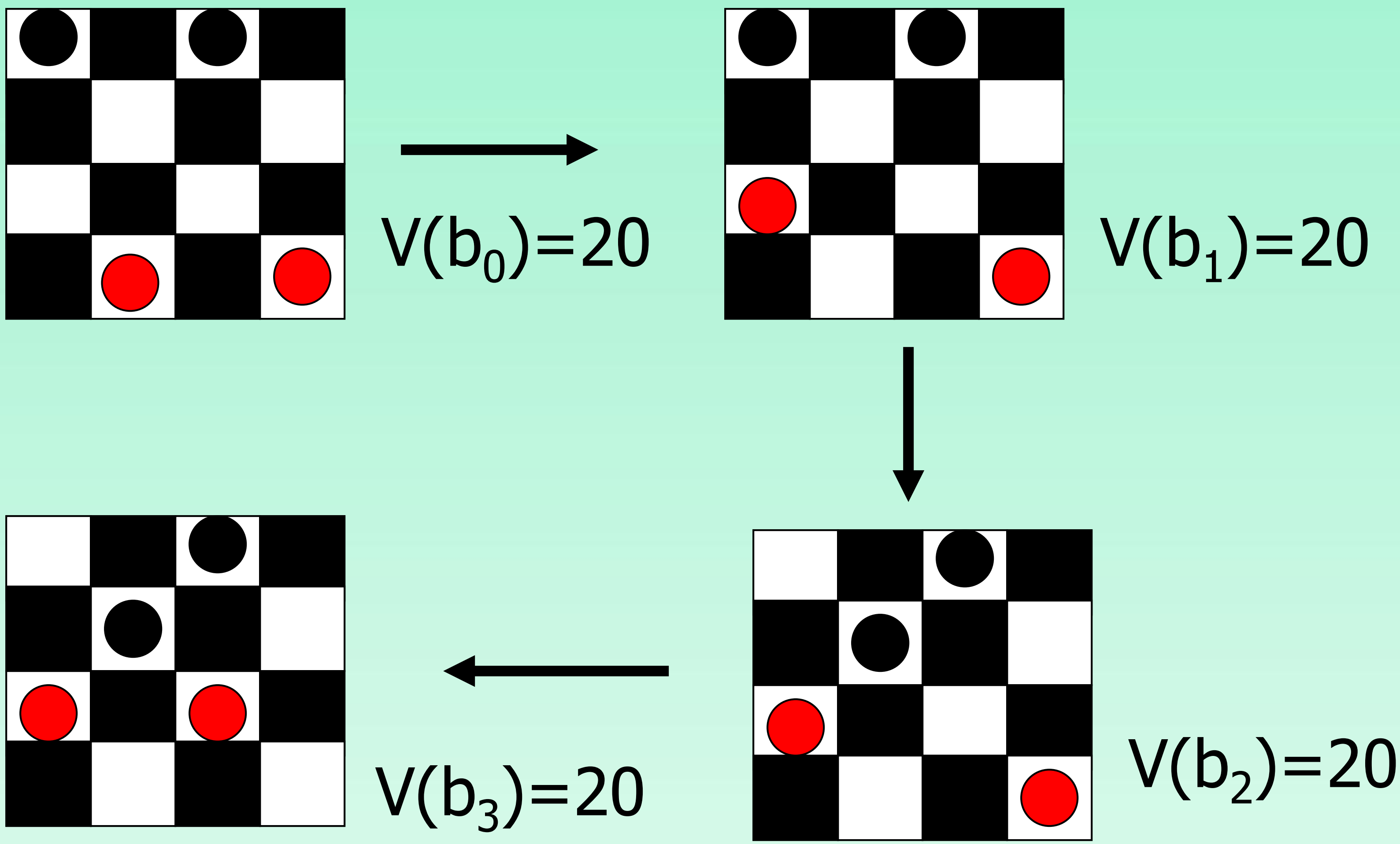
$\omega_i \leftarrow \omega_i + \eta\ f_i\ error(b)$

$\omega_0 \leftarrow \omega_0 + 0.1 * 1 * 0$

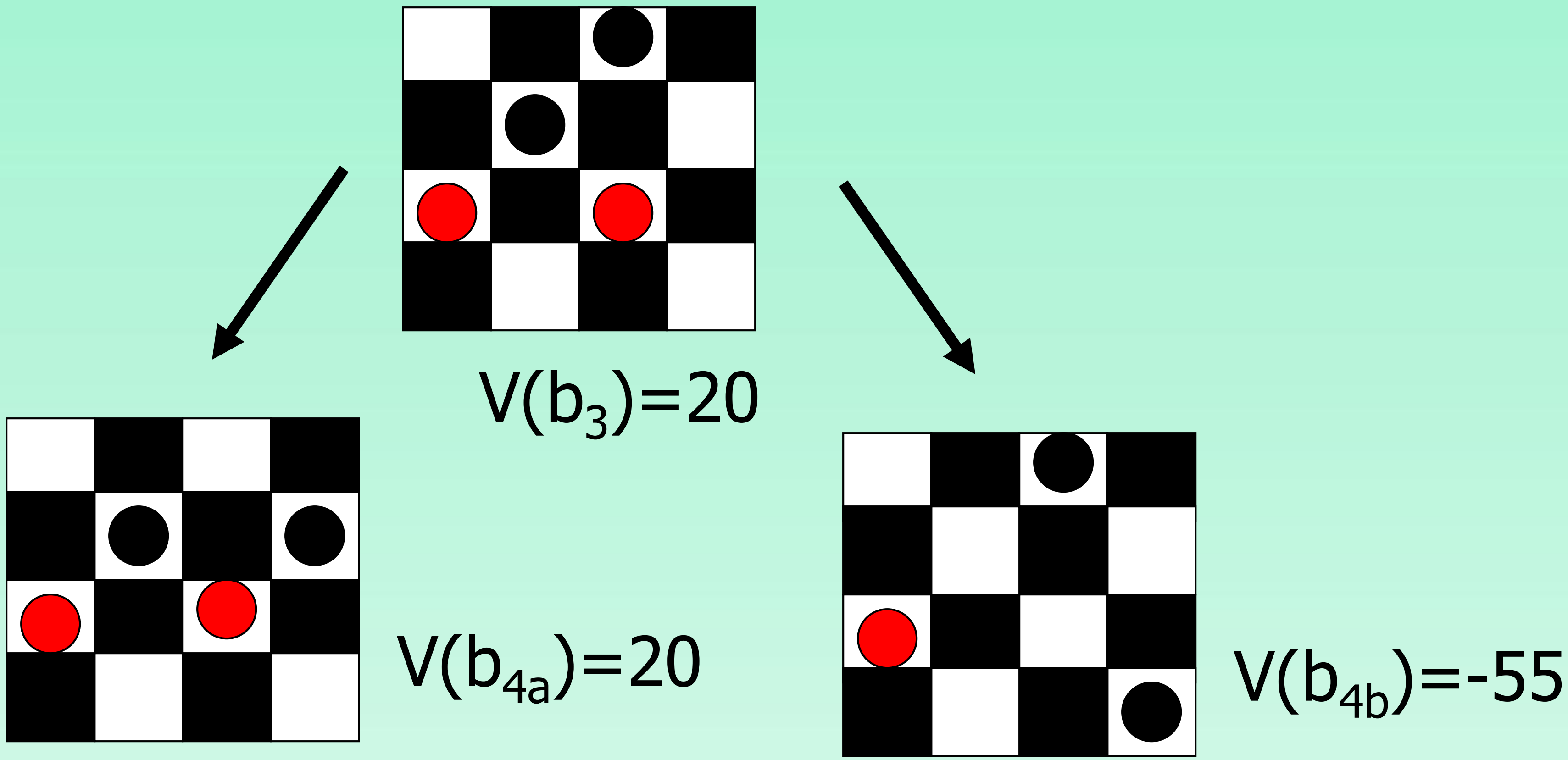
$\omega_1 \leftarrow \omega_1 + 0.1 * 2 * 0$

$\omega_2 \leftarrow \omega_2 + 0.1 * 2 * 0$

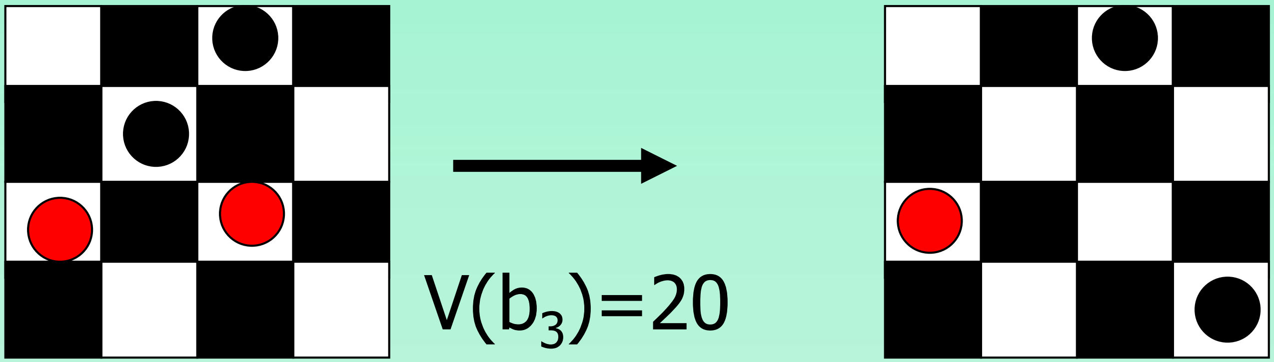
Example: 4x4 checkers



Example: 4x4 checkers



Example 4x4 checkers



$V(b_3)=20$ $V(b_4)=-55$

1. Compute $error(b_3) = V_{train}(b_3) - V(b_3) = V(b_4) - V(b_3) = -75$
2. For each board feature fi, update weight

$\omega_i \leftarrow \omega_i + \eta\ f_i\ error(b)$: $\omega_0=-10$, $\omega_1 =75$, $\omega_2 =-60$

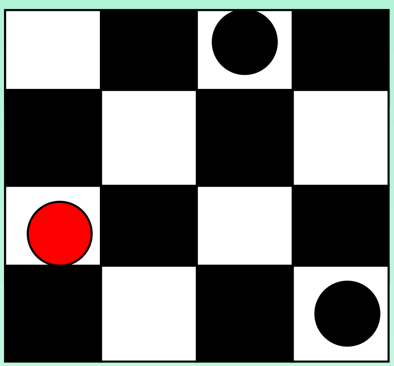
$\omega_0 \leftarrow \omega_0 - 0.1 * 1 * 75$, $\omega_0 = -17.5$

$\omega_1 \leftarrow \omega_1 - 0.1 * 2 * 75$, $\omega_1 = 60$

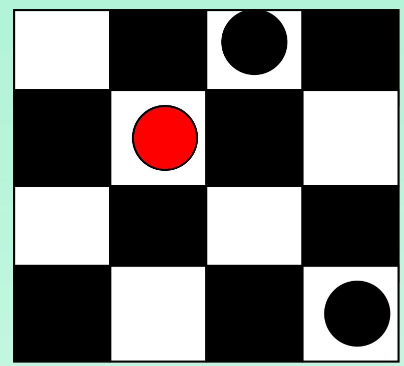
$\omega_2 \leftarrow \omega_2 - 0.1 * 2 * 75$, $\omega_2 = -75$

Example: 4x4 checkers

$\omega_0 = -17.5$, $\omega_1 = 60$, $\omega_2 = -75$

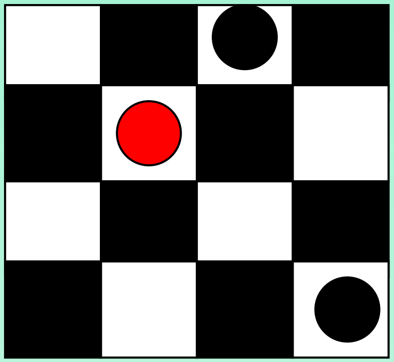


$V(b_4)$=-107.5

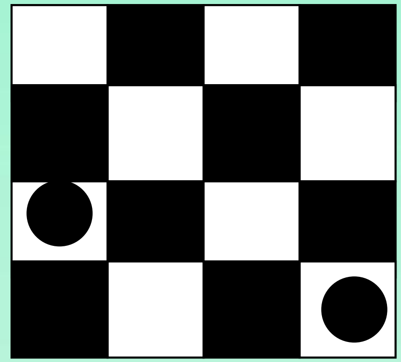


$V(b_5)$=-107.5

Example 4x4 checkers



$V(b_5)=-107.5$



$V(b_6)=-167.5$

$\text{error}(b_5) = V_{train}(b_5) - V(b_5) = V(b_6) - V(b_5) = -60$

$\omega_0=-17.5,\ \omega_1 =60,\ \omega_2 =-75$

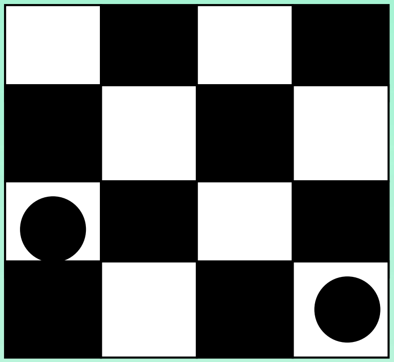
$\omega_i \leftarrow \omega_i + \eta\ f_i\ \text{error}(b)$

$\omega_0 \leftarrow \omega_0 - 0.1 * 1 * 60,\ \omega_0 = -23.5$

$\omega_1 \leftarrow \omega_1 - 0.1 * 1 * 60,\ \omega_1 = 54$

$\omega_2 \leftarrow \omega_2 - 0.1 * 2 * 60,\ \omega_2 = -87$

Example 4x4 checkers



Final board state: black won $V_f(b_6)$=-100

$V(b_6)$=-197.5

$error(b_6) = V_{train}(b_6) - V(b_6) = V_f(b_6) - V(b_6) = 97.5$

ω_0=-23.5, ω_1 =54, ω_2 =-87

$\omega_i \leftarrow \omega_i + \eta\ f_i\ error(b)$

$\omega_0 \leftarrow \omega_0 + 0.1 * 1 * 97.5,\ \omega_0 = -13.75$

$\omega_1 \leftarrow \omega_1 + 0.1 * 0 * 97.5,\ \omega_1 = 54$

$\omega_2 \leftarrow \omega_2 + 0.1 * 2 * 97.5,\ \omega_2 = -67.5$