Foundations of Artificial Intelligence

G3. Board Games: Evaluation Functions

Malte Helmert

University of Basel

May 11, 2026

G3.1 Evaluation Functions

G3.2 Summary

Board Games: Overview

chapter overview:

- G1. Introduction and State of the Art
- G2. Formal Definition and Minimax Search
- G3. Evaluation Functions
- G4. Alpha-Beta Search
- G5. Stochastic Games
- G6. Monte-Carlo Tree Search Framework
- G7. Monte-Carlo Tree Search Variants

G3.1 Evaluation Functions

Motivation

- problem: game tree too big
- idea: search only up to predefined depth
- depth reached: estimate the utility value according to heuristic criteria (as if terminal position had been reached)

Evaluation Functions

Definition (evaluation function)

Let $\mathcal{S}$ be a game with set of positions S.
An evaluation function for $\mathcal{S}$ is a function

$$h : S \to \mathbb{R}$$

which assigns a real-valued number to each position $s \in S$.

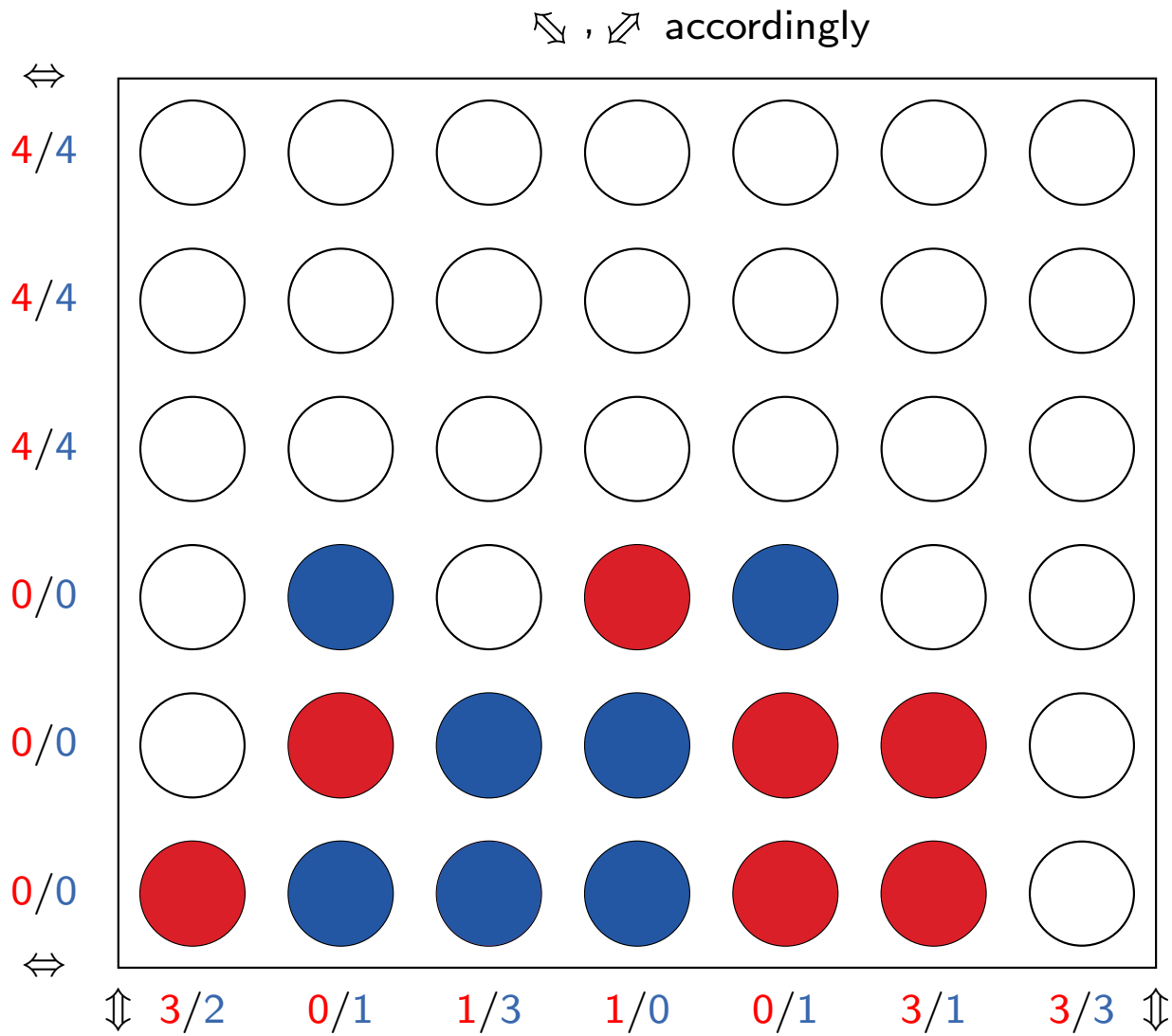
Looks familiar? Commonalities? Differences?

Intuition

accuracy of evaluation function is crucial

- high values should relate to high "winning chances"
- at the same time, the evaluation should be efficiently computable in order to be able to search deeply

Example: Connect Four



evalution function: difference of number of possible lines of four

General Method: Linear Evaluation Functions

expert knowledge often represented with weighted linear functions:

$$h(s) = w_0 + w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s),$$

where w_i are weights and f_i are features.

- assumes that feature contributions are mutually independent (usually wrong but acceptable assumption)
- features are (usually) provided by human experts
- weights provided by human experts or learned automatically

General Method: Linear Evaluation Functions

expert knowledge often represented with weighted linear functions:

$$h(s) = w_0 + w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s),$$

where w_i are weights and f_i are features.

example: evaluation function in chess (cf. Lolli 1763)

feature	f_p^{player}	f_k^{player}	f_b^{player}	f_r^{player}	f_q^{player}
no. of pieces	pawn	knight	bishop	rook	queen
weight for MAX	1	3	3	5	9
weight for MIN	-1	-3	-3	-5	-9

often additional features based on pawn structure, mobility, ...

$$\rightsquigarrow h(s) = f_p^{\text{MAX}}(s) + 3f_k^{\text{MAX}}(s) + 3f_b^{\text{MAX}}(s) + 5f_r^{\text{MAX}}(s) + 9f_q^{\text{MAX}}(s) - f_p^{\text{MIN}}(s) - 3f_k^{\text{MIN}}(s) - 3f_b^{\text{MIN}}(s) - 5f_r^{\text{MIN}}(s) - 9f_q^{\text{MIN}}(s)$$

General Method: State Value Networks

alternative: evaluation functions based on neural networks

- value network takes position features as input
  (usually provided by human experts)
- and outputs utility value prediction
- weights of network learned automatically

example: value network of AlphaGo

- start with policy network trained on human expert games
- train sequence of policy networks by self-play against earlier version
- final step: convert to utility value network
  (slightly worse informed but much faster)

$\rightsquigarrow$ Mastering the game of Go with deep neural networks and tree search
(Silver et al., 2016)

How Deep Shall We Search?

- objective: search as deeply as possible within a given time
- problem: search time difficult to predict
- solution: iterative deepening
  - sequence of searches of increasing depth
  - time expires: return result of previously finished search
  - overhead acceptable ($\rightsquigarrow$ Chapter B8)
- refinement: search deeper in "turbulent" states
  (i.e., with strong fluctuations of the evaluation function)
  $\rightsquigarrow$ quiescence search
  - example chess: deepen the search after capturing moves

G3.2 Summary

Summary

- In practice, the search depth of game tree search must be bounded $\rightsquigarrow$ apply evaluation functions.
- Evaluation functions play a similar role to heuristics in state-space search.
- Many evaluation functions are linear combinations of simple features.
- Learning is often used to tune weights of such linear combinations or to learn the entire evaluation function.