Stefan Hübecker

TECHNISCHE UNIVERSITÄT DARMSTADT

Temporal-Difference Search in Computer Go

(Combining TD Learning with Search)

Overview

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Introduction

Go

Temporal Difference Learning

Temporal Difference Search

Dyna-2

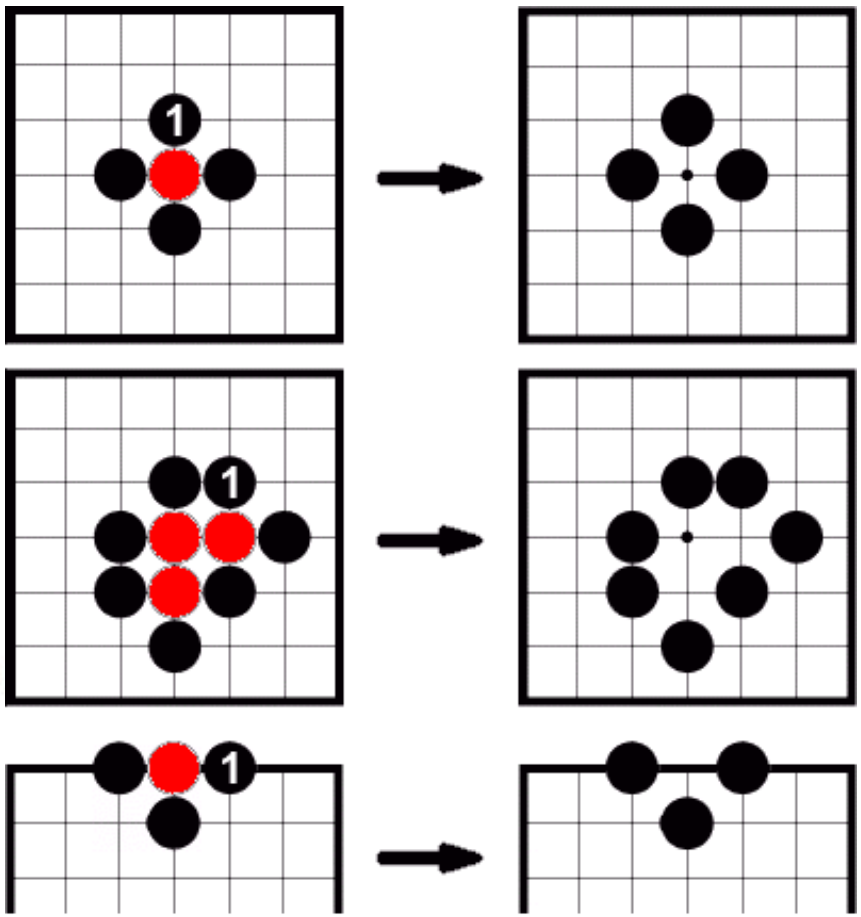
Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problems
  - Go has 10^{170} states (*only* about 10^{80} atoms in the entire universe)
  - Up to 361 legal moves
  - Long-term effect of a move may only be revealed after hundreds of additional moves
  - Difficult to construct a global value function
- Goals
  - Combine temporal-difference learning with simulation-based search
  - Update value function online
  - Value function approximation for generalizing related states
- Concept
  - *Dyna-2*
  - Combine temporal-difference learning with simulation-based search
  - Use learning and planning simultaneously
  - Online retraining instead of offline computation of all leaves

Go

TECHNISCHE UNIVERSITÄT DARMSTADT



Go

TECHNISCHE UNIVERSITÄT DARMSTADT

- Long-term effects may only be revealed after hundreds of additional moves
- Inspect individual patterns/shapes instead
- Older algorithms use handcrafted patterns to create rules
  - Interactions between patterns problematic
  - Shape knowledge hard to quantify and encode
- Supervised Learning can predict human plays with an accuracy of 30-40%
- But cannot play on its own - focus on mimicking instead of understanding a position

Temporal Difference Learning

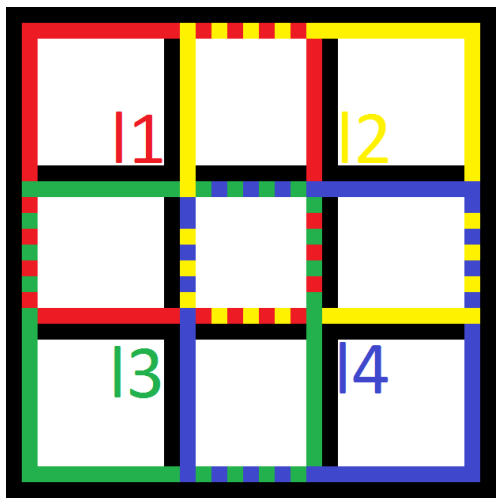
TECHNISCHE UNIVERSITÄT DARMSTADT

- Basically the same as before
- $TD(\lambda)$ is used

Temporal Difference Learning
Local Shape Features

TECHNISCHE UNIVERSITÄT DARMSTADT

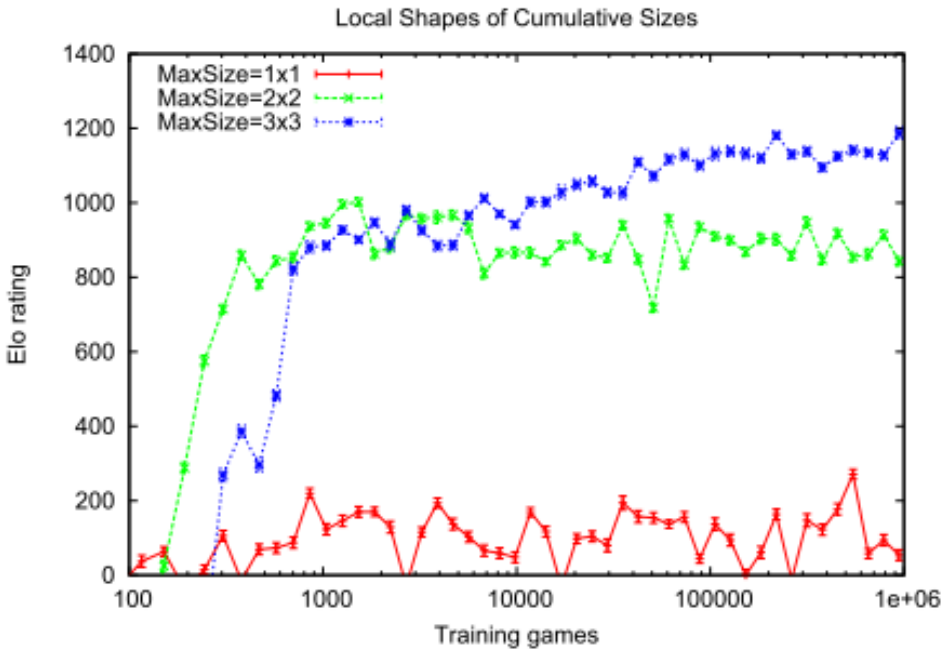
- Simple representation of a local board configuration within a square $k \times k$
- State in Go: $s \in \{\cdot, \circ, \bullet\}^{N \times N}$
- Shape in Go: $l \in \{\cdot, \circ, \bullet\}^{k \times k}$
- Create a feature vector $\phi(s)$ of the shape l at each position on the board
- A local shape feature $\phi_i(s)$ is 1 if the shape matches exactly, otherwise 0

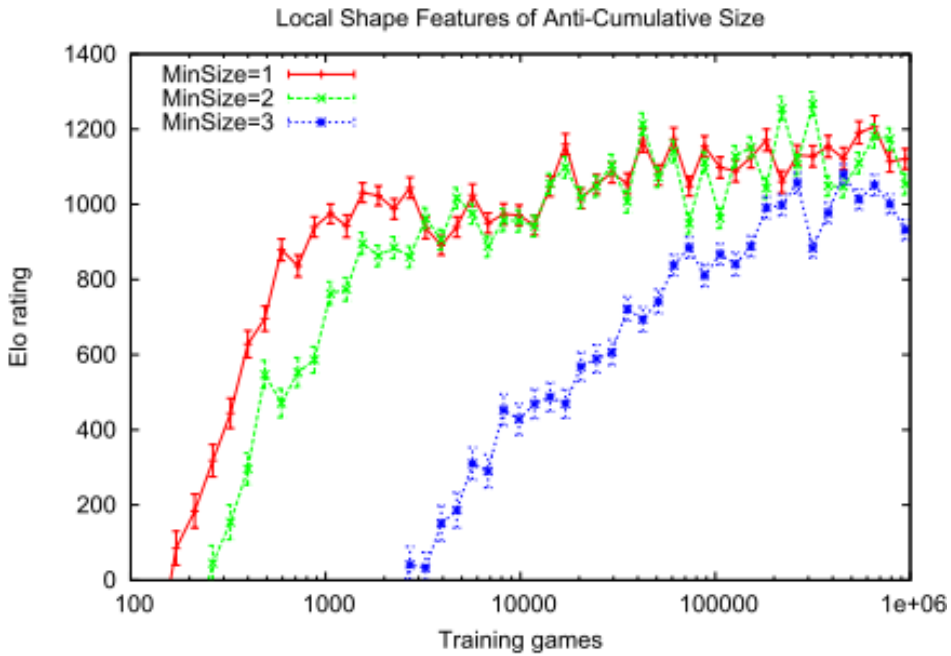


$\phi(s) = \{l1, l2, l3, l4\}$

Temporal Difference Learning
Local Shape Features

TECHNISCHE UNIVERSITÄT DARMSTADT

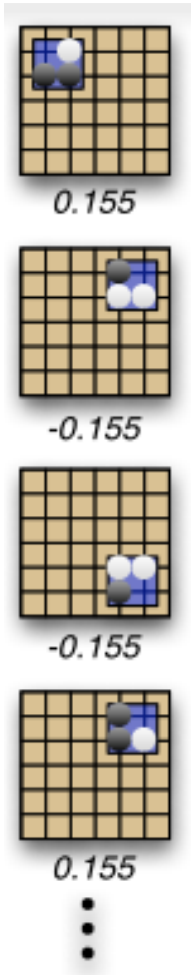




Temporal Difference Learning
Weight sharing

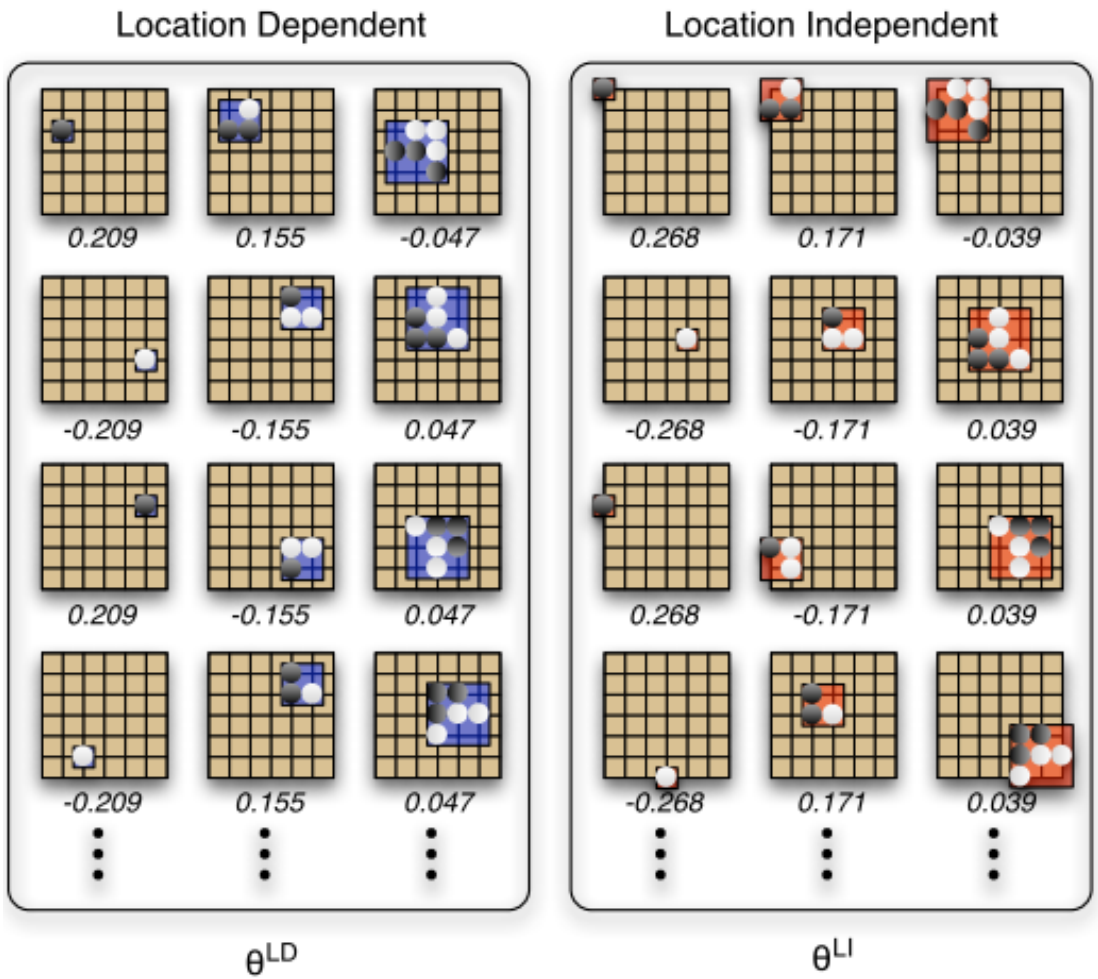
TECHNISCHE UNIVERSITÄT DARMSTADT

- Define equivalence relationships over local shapes
- Rotations, reflections and inversions(black flipped to white and white to black)
- Every local shape feature shares the same weight θ, but the sign may differ
- Inversions get negative weight



Temporal Difference Learning
Location dependency

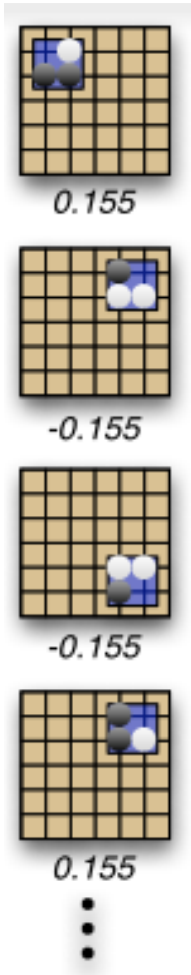
TECHNISCHE UNIVERSITÄT DARMSTADT



Temporal Difference Learning
Weight sharing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Define equivalence relationships over local shapes
- Rotations, reflections and inversions(black flipped to white and white to black)
- Every local shape feature shares the same weight *theta*, but the sign may differ
- Inversions get negative weight

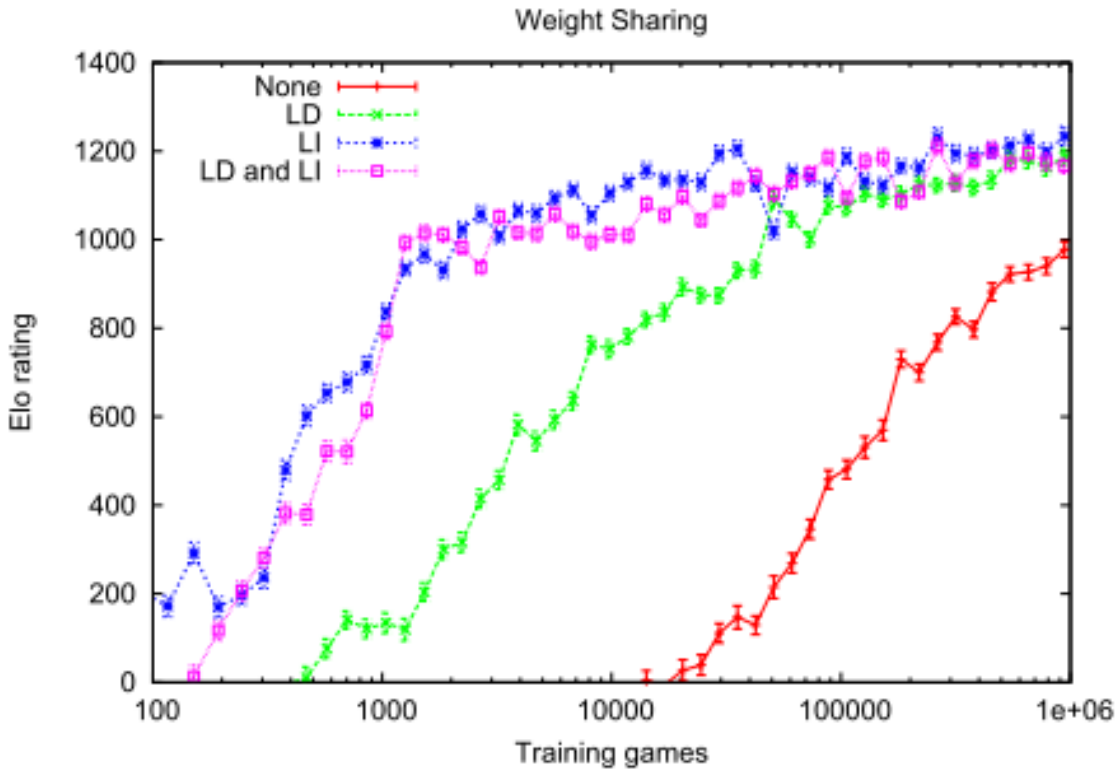


Local shape features	Total features	Unique weights	Max active features
1×1 Location Independent 1×1 Location Dependent	243	1 15	81 81
2×2 Location Independent 2×2 Location Dependent	5184	8 344	64 64
3×3 Location Independent 3×3 Location Dependent	964467	1418 61876	49 49
Total	969894	63303	388

On a 9x9 Board

Temporal Difference Learning
Weight sharing

TECHNISCHE UNIVERSITÄT DARMSTADT



- Computational performance of LD and LI was equal

Temporal Difference Learning
Value Function

TECHNISCHE UNIVERSITÄT DARMSTADT

- Value function should give reward of 1 if Black wins and 0 if White wins
- Basically the value function represents Black's winning probability
- Black tries to maximizes the value function while White tries to minimize it
- Approximation by a logistic-linear combination of local shape feature and location dependent/independent weights

$$V(s) = \sigma(\phi(s) * \theta^{LI} + \phi(s) * \theta^{LD}) \tag{1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

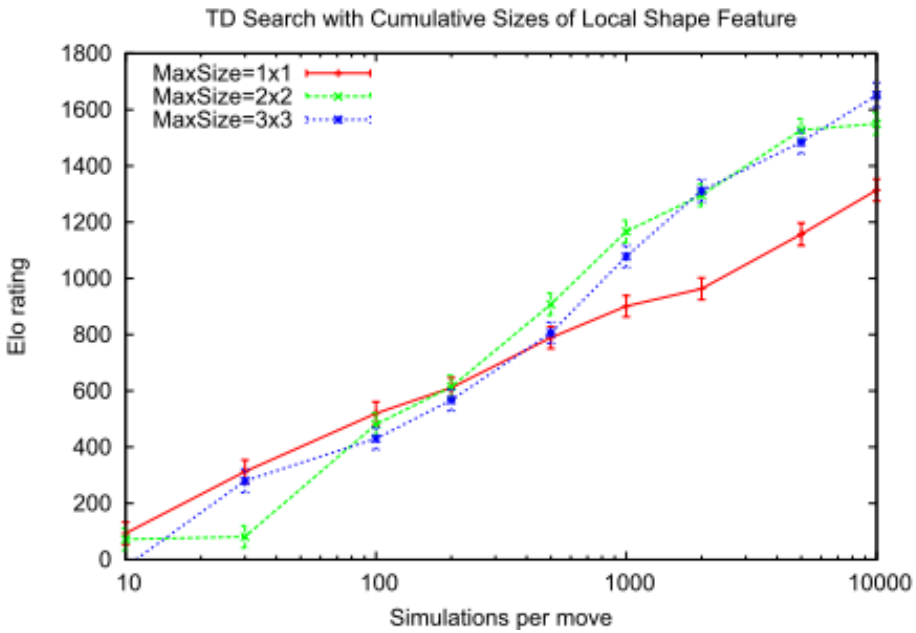
Temporal Difference Search

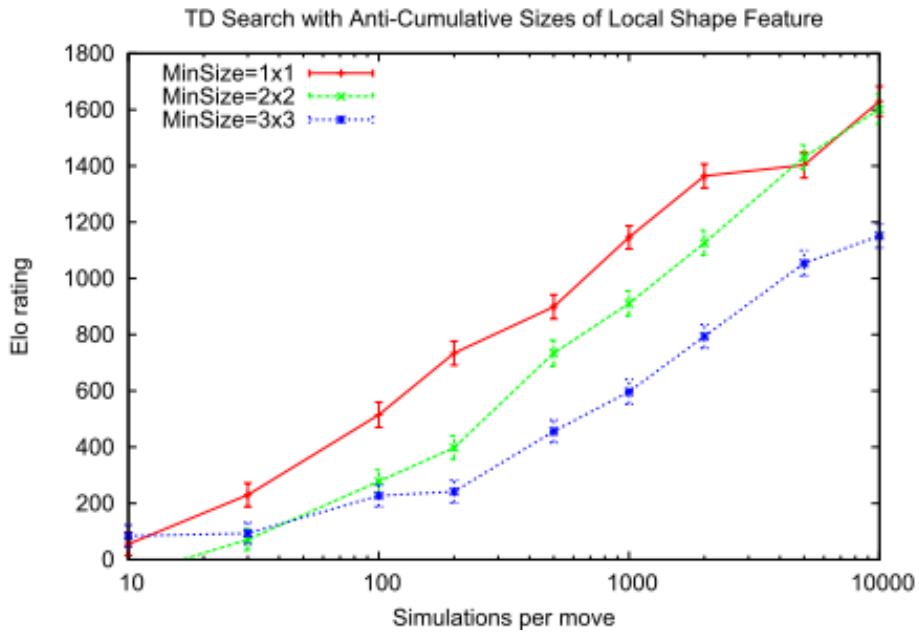
TECHNISCHE UNIVERSITÄT DARMSTADT

- Idea: Apply $TD(0)$ learning beginning at the current state
- In a game G the current state s_t defines a new game G'_t
- Begin searches always in the current subgame G'_t
- Search online, new search after each real action
- Needs excellent understanding of game model and rules for simulation
- Value function specializes on what happens next rather than learning all possibilities
- Improved version: Linear TDS
  - Also restarts search every real game step
  - But saves and updates value function - $TD(\lambda)$

Temporal Difference Search
Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT





- TD learning achieved about 1200 ELO after 1 million games
- 3x3 shapes bring no significant improvement

Dyna-2

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dyna
  - Combines TD learning and TD search
  - Applies TD learning to real experience and simulated experience
  - Learns a model of MDP from real experience
  - Updates action value function from both experiences
- Dyna-Q
  - Remembers all state transitions and rewards from visited states and actions from its real experience
- Dyna-2
  - Maintain two separate memories
    - Long-term memory: Real experience
    - Short-term memory: Used during search and updated from simulated experience
    - Only consider states that have their root in the current one, no global distribution over all states

Dyna-2
Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

- The memory $M = (\phi, \theta)$ is represented via a vector of features ϕ and a vector of corresponding parameters θ
- The feature vector $\phi(s, a)$ represents the state s and action a
- The parameter vector θ is used to approximate the value function by forming a linear combination $\phi(s, a) * \theta$ of the features and parameters in M
- *Long-term memory*: $M = (\phi, \theta)$, *Short-term memory*: $\overline{M} = (\overline{\phi}, \overline{\theta})$
- *Long-term value function* $Q(s, a)$ uses only long-term memory to approximate the *true value function* $Q^{\pi}(s, a)$
- Short-term value function $\overline{Q}(s, a)$ uses both memories by forming a linear combination of both feature- and both parameter vectors

$$Q(s, a) = \phi(s, a) * \theta \tag{3}$$

$$\overline{Q}(s, a) = \phi(s, a) * \theta + \overline{\phi}(s, a) * \overline{\theta} \tag{4}$$

Dyna-2
Memory

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Long-term memory* is used for general knowledge which is independent of the agents current state
- E.g. in chess it would be the knowledge that the Bishop is worth about 3.5 pawns
- *Short-term memory* represents local knowledge which is specific to the agents current region of the state space
- It is used to correct the *long-term value function*, representing adjustments to provide a more accurate local approximation of the *true value function*
- Back to the chess example it would mean the knowledge that the bishop is worth 1 pawn less in the endgame
- Continuous adjustments of the short-term memory increase the overall quality of approximation

Dyna-2
Algorithm Learn

TECHNISCHE UNIVERSITÄT DARMSTADT

1: **procedure** LEARN
2: Initialise $\mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}$ ▷ State transition and reward models
3: $\theta \leftarrow 0$ ▷ Clear long-term memory
4: **loop**
5: $s \leftarrow s_0$ ▷ Start new episode
6: $\overline{\theta} \leftarrow 0$ ▷ Clear short-term memory
7: $e \leftarrow 0$ ▷ Clear eligibility trace
8: SEARCH(s)
9: $a \leftarrow \epsilon$–greedy$(s; \overline{Q})$
10: **while** s is not terminal **do**
11: Execute a, observe reward r, state s'
12: $\mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'} \leftarrow$ UPDATEMODEL(s, a, r, s')
13: SEARCH(s')
14: $a' \leftarrow \epsilon$–greedy$(s'; \overline{Q})$
15: $\delta Q \leftarrow r + Q(s', a') - Q(s, a)$ ▷ TD-error
16: $\theta \leftarrow \theta + \alpha \delta Q e$ ▷ Update weights
17: $e \leftarrow \lambda e + \phi$ ▷ Update eligibility trace
18: $s \leftarrow s', a \leftarrow a'$
19: **end while**
20: **end loop**
21: **end procedure**

Dyna-2
Algorithm Search

TECHNISCHE UNIVERSITÄT DARMSTADT

procedure SEARCH(s)
  while time available **do**
    $\overline{e} \leftarrow 0$ ▷ Clear eligibility trace
    $a \leftarrow \overline{\epsilon}$–greedy$(s; \overline{Q})$
    while s is not terminal **do**
      $s' \sim \mathcal{P}^a_{ss'}$ ▷ Sample state transition
      $r \leftarrow \mathcal{R}^a_{ss'}$ ▷ Sample reward
      $a' \leftarrow \overline{\epsilon}$–greedy$(s'; \overline{Q})$
      $\overline{\delta}Q \leftarrow r + \overline{Q}(s', a') - \overline{Q}(s, a)$ ▷ TD-error
      $\overline{\theta} \leftarrow \overline{\theta} + \overline{\alpha}\overline{\delta}Q\overline{e}$ ▷ Update weights
      $\overline{e} \leftarrow \overline{\lambda}\overline{e} + \overline{\phi}$ ▷ Update eligibility trace
      $s \leftarrow s', a \leftarrow a'$
    end while
  end while
end procedure

- $\overline{\epsilon}$-greedy chooses an action that maximizes the short-term value function
  $a_u = argmax\overline{Q}(s_u, b)$

Dyna-2
Algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- Without short-term memory $\overline{\theta} = \emptyset$ the search has no effect and is reduced to a linear Sarsa algorithm
- Without long-term memory $\theta = \emptyset$ the Dyna-2 is reduced to a TD search
- The real experience for the long-term memory may be accumulated offline in a given training environment
- The short-term memory will still adapt to new situations online later to correct learned misconceptions

Dyna-2
in Computer Go

TECHNISCHE UNIVERSITÄT DARMSTADT

- Special features specific to long/short-term memory might bring improvements
- Remember the missing improvements from 3x3 shapes in TDS
- But using the same ones brings a different advantage...

Dyna-2
in Computer Go

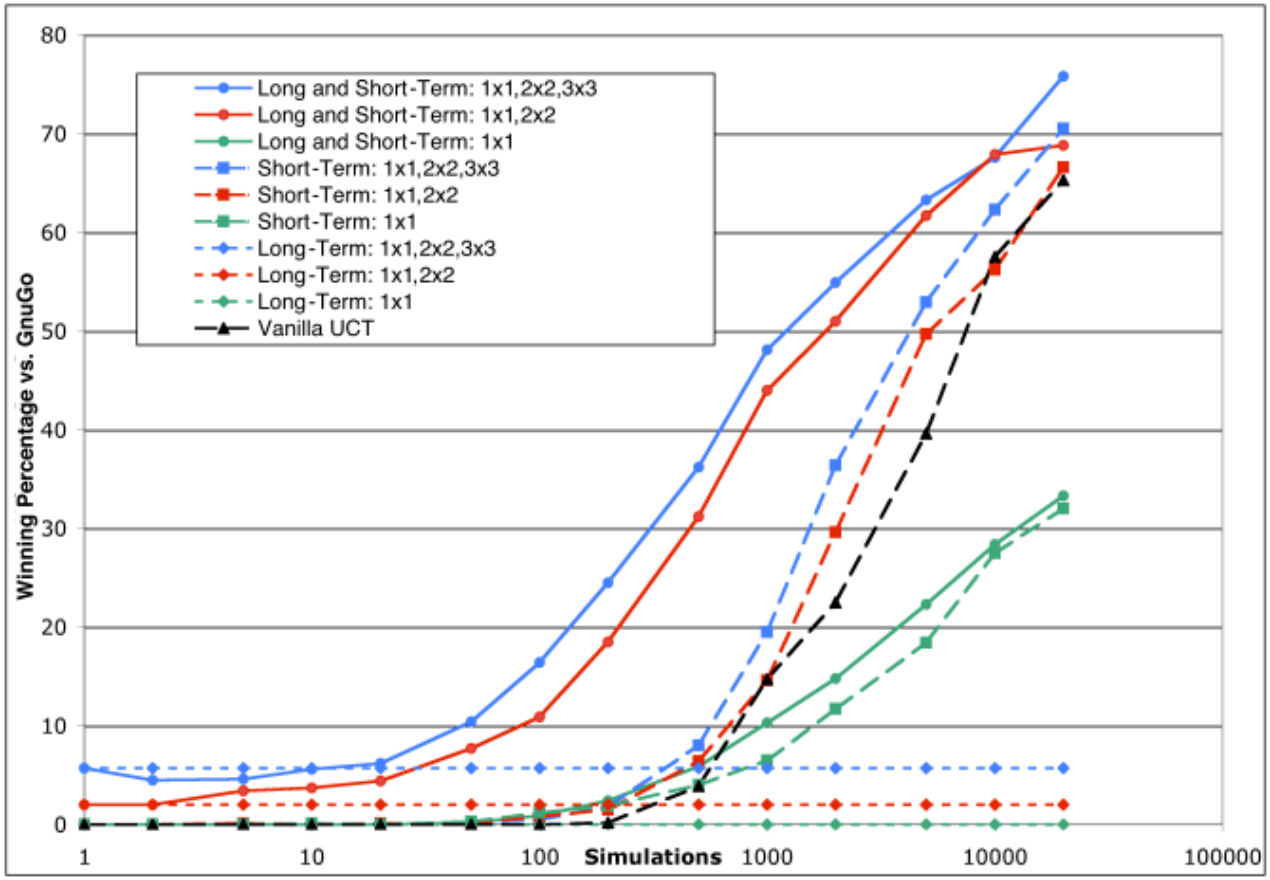
TECHNISCHE UNIVERSITÄT DARMSTADT

- Special features specific to long/short-term memory might bring improvements
- Remember the missing improvements from 3x3 shapes in TDS
- But using the same ones brings a different advantage:

- Only one memory needed for both
- During initialization of the Search: $\overline{\theta} \leftarrow \theta$
- And therefore only one value function too
- $\overline{Q}(s, a) = \overline{\phi}(s, a) * \overline{\theta}$

Dyna-2
Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT



Dyna-2
Too greedy?

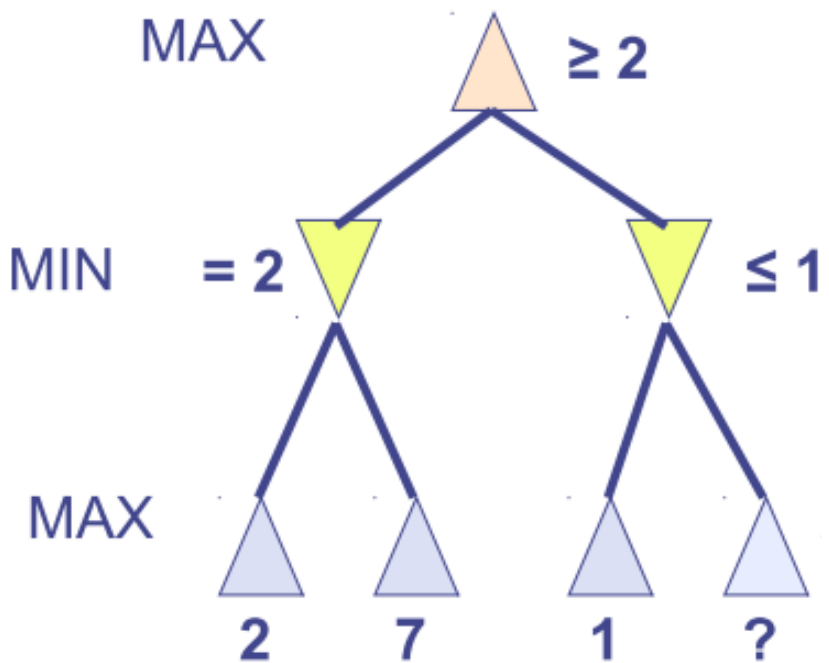
TECHNISCHE UNIVERSITÄT DARMSTADT

- Currently using $\bar{\epsilon}$-greedy to choose an action that maximizes the short-term value function
- Is that the best option for a two player game?

Dyna-2
Too greedy?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Currently using $\bar{\epsilon}$-greedy to choose an action that maximizes the short-term value function
- Is that the best option for a two player game?



Dyna-2
Two-Phase Alpha-Beta Search

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduce a second search for the best action: Alpha-Beta Search(Pruning)
- Use the short-term value function to evaluate the leaves at depth *d*
- Improved by iterative deepening, killer-move heuristics and null-move pruning
- In practice shares its time with the TDS 50%-50%

Search algorithm	Memory	Elo rating
Alpha-beta	Long-term	1350
Dyna-2	Long&Short-term	2030
Dyna-2 + $\alpha\beta$	Long&Short-term	2130

Dyna-2
Two-Phase Alpha-Beta Search

TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduce a second search for the best action: Alpha-Beta Search(Pruning)
- Use the short-term value function to evaluate the leaves at depth *d*
- Improved by iterative deepening, killer-move heuristics and null-move pruning
- In practice shares its time with the TDS 50%-50%

Search algorithm	Memory	Elo rating
Alpha-beta	Long-term	1350
Dyna-2	Long&Short-term	2030
Dyna-2 + $\alpha\beta$	Long&Short-term	2130
GnuGo		$\sim$ 1800
NeuroGo/Honte		$\sim$ 1700

Dyna-2 and $\alpha\beta$ Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- So we perform better than MCTS?

Search algorithm	Elo rating
NeuroGo/Honte	$\sim$ 1700
GnuGo	$\sim$ 1800
UCT	$\sim$ 1950
Dyna-2 + $\alpha\beta$	2130

Dyna-2 and $\alpha\beta$ Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

- So we perform better than MCTS?

Search algorithm	Elo rating
NeuroGo/Honte	$\sim$ 1700
GnuGo	$\sim$ 1800
UCT	$\sim$ 1950
Dyna-2 + $\alpha\beta$	2130
Zen/Fuego/MoGo	2600+

- Only better than a basic one though
- Improved versions of MCTS are simply better and also faster than Dyna-2
- But MCTS got much attention and improvements recently
- Still much potential to improve Dyna-2/TD searches

That’s all Folks!

TECHNISCHE UNIVERSITÄT DARMSTADT

Thank you for your attention

Any questions?