Sample-Based Learning and Search with Permanent and Transient Memories

David Silver, Richard S. Sutton, Martin Müller
A Brief History of Computer Go

- **2005**: Computer Go is impossible!
- **2006**: UCT invented and applied to 9x9 Go (Kocsis, Szepesvari; Gelly et al.)
- **2007**: Human master level achieved at 9x9 Go (Gelly, Silver; Coulom)
- **2008**: Human grandmaster level achieved at 9x9 Go (Teytaud et al.)
- **2009**: Human master level achieved at 19x19 Go?
Big Ideas for Big Environments

1. Function approximation
2. Tracking
3. Sample-based planning
4. Bootstrap
1. Function Approximation

- **Problem:**
  - State space too large to represent all values
  - No generalisation between states

- **Solution:**
  - Approximate value function with a smaller number of parameters
2. Tracking

- **Problem:**
  - Value function too complex to represent

- **Solution:**
  - Focus on current region of state space
3. Sample-Based Planning (*Dyna*)

- **Problem:**
  - Insufficient experience to learn good policy

- **Solution:**
  - Sample experience from a *model* of the world
  - Learn from simulated experience
4. Bootstrapping ($TD$)

- **Problem:**
  - Return has high variance

- **Solution:**
  - Learn from value of successor states
Sample-Based Search

- Sample-based planning + tracking
- Simulate episodes starting from current state
- Learn from simulated experience
Sample-Based Search

Current State

Simulation

Terminal State
Sample-Based Search Algorithms

- Monte-Carlo simulation
- Monte-Carlo tree search
- UCT
- Heuristic UCT

States represented individually
Monte-Carlo learning
New Idea

- Dyna-2

  Linear value function approximation

  Temporal difference learning
Dyna-2: Two Memories

- A memory is a vector of features $\phi(s,a)$ and parameters $\theta$
- Permanent memory $(\phi, \theta)$ learned from all experience
- Transient memory $(\bar{\phi}, \bar{\theta})$ learned from recent experience
- Transient memory is forgotten over time
Dyna-2: Learning and Search

- **Sample-based learning:**
  - Permanent memory is updated from real experience using Sarsa($\lambda$)

- **Sample-based search:**
  - Transient memory is updated from simulated experience using Sarsa($\lambda$)
  - Transient memory is forgotten at the end of each real episode
Real Experience

Current State

Learning

Permanent memory

\[ Q(s, a) = \phi(s, a)^T \theta \]
Simulated Experience

Current State

Transient memory

$$\bar{Q}(s, a) = \phi(s, a)^T \theta + \bar{\phi}(s, a)^T \bar{\theta}$$
Shape Knowledge in Go

- Go players utilise a large vocabulary of shapes:
  - One-point jump
  - Ponnuki
  - Hane
Local Shape Features

- Binary features matching a local configuration of stones
- All possible locations and configurations from 1x1 to 3x3
- ~1 million features for 9x9 Go
Empty Triangle

- Permanent memory is based on all experience
- General knowledge about world
Guzumi

- Transient memory forgets old experience
- Local knowledge about current situation
“Blood Vomiting Game”
Results for Dyna-2
Dyna-2 + Alpha-Beta Search

- Like a traditional alpha-beta search, except:
  - Evaluation function is re-learned every move
  - Based on simulated experience
  - Using permanent + transient memories
Dyna-2 + Alpha-Beta

The graph shows the comparison of Wins vs. GnuGo between different strategies and UCT. The strategies include Permanent + Transient with various ply counts (1 to 6 ply) and Permanent with ply counts (1 to 6-ply). The graph plots the number of simulations against the number of wins, demonstrating the performance of each strategy over different simulation counts.
Bootstrapping Results

![Graph showing the relationship between Wins vs. GnuGo and Lambda. The graph peaks at Lambda values around 0.5 and then decreases sharply as Lambda increases.](image-url)
Conclusions (9x9 Go)

- Sample-based search methods outperform other approaches
- Function approximation improves performance
- Combining permanent and transient memories significantly improves performance
- TD learning outperforms Monte-Carlo learning
Extrapolation (19x19 Go)

- Sample-based search methods **massively** outperform other approaches
- Function approximation is **crucial**
- Combining permanent and transient memories significantly improves performance
- TD learning outperforms Monte-Carlo learning
Questions?
Permanent and Transient Memories