Keepaway Soccer:

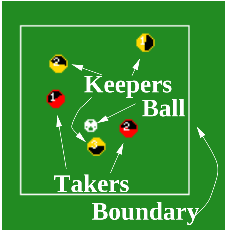
From Machine Learning Testbed to Benchmark

Peter Stone Gregory Kuhlmann Matthew E. Taylor
Yaxin Liu

Learning Agents Research Group
Department of Computer Sciences
The University of Texas at Austin

RoboCup Symposium, 2005

Keepaway: A Subtask of Simulated Soccer



- Play in a **small area**
- **Keepers** try to keep the ball
- **Takers** try to get the ball
- **Episode:**
  - Players and ball reset randomly
  - Ball starts near a keeper
  - Ends when taker gets the ball or ball goes out
- Performance measure: **average possession duration**
- Use **pre-defined skills**:
  - HoldBall, PassBall(k), GoToBall, GetOpen

Previous Studies

Keepaway as a Successful Machine Learning Testbed

- **Temporal-Difference Learning** [Stone and Sutton, 2001]
- **Evolutionary Algorithms** [DiPietro et al., 2002]
- **Genetic Programming** [Hsu and Gustafson, 2002]
- **Relational Reinforcement Learning** [Walker et al., 2005]
- **Transfer Learning** [Taylor and Stone, 2005]

Previous Studies

Problems as a Benchmark

- Results not directly comparable
  - Different simulators, low-level behaviors, experimental setups, and evaluation metrics.
- Prohibitively large startup cost
  - Must reimplement keepaway players
  - Requires domain expertise

Previous Studies

Problems as a Benchmark

- Results not directly comparable
  - Different simulators, low-level behaviors, experimental setups, and evaluation metrics.
- Prohibitively large startup cost
  - Must reimplement keepaway players
  - Requires domain expertise

Outline

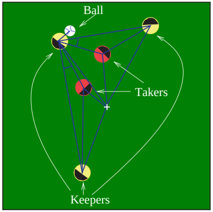
1. Benchmark Repository
   - Standardized Players
   - Online Resources
2. An Empirical Study
   - Fixed Policies
   - Function Approximators

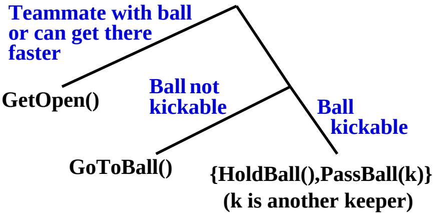
Outline

Player Framework

- **Open-source** player (C++)
  - built on UvA Trilearn base player
- Implements all but learning
  - high-level skills, world model, communication.
- Simple interface to learning algorithm

Standardized Learning Scenario





- **Keeper With Ball** can HoldBall(), PassBall(k)
- **Relational State Features:**
  - 11 distances among players, ball, and center
  - 2 angles to takers along passing lane
- **Takers** follows fixed policy

Learning Agent Interface

- Learner must implement three C++ functions
  - `int startEpisode(double state[])`
    called on first action opportunity
  - `int step(double reward, double state[])`
    called on each action opportunity after first
  - `void endEpisode(double reward)`
    called once at end of episode
- Presents domain as continuous state Semi-Markov Decision Process
- Allows for **TD-learning** or **policy search**

Online Resources

- Complete player **source code**
- Concrete **results** for comparison
- Graphical **tools** to evaluate performance
  - Generate learning curves
- Step-by-step **tutorials**
  - Walk through how to apply new learning algorithm
  - Students with **no prior keepaway experience** completed basic tutorial in **under an hour**

Outline

1. Benchmark Repository
   - Standardized Players
   - Online Resources

2. An Empirical Study
   - Fixed Policies
   - Function Approximators

Fixed Policies

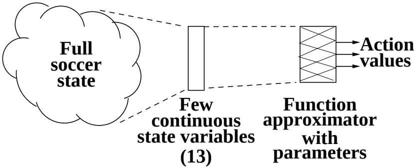
- Fixed policies included with benchmark players
  - **Always Hold:** always HoldBall
  - **Random:** HoldBall or PassBall(k) randomly
  - **Hand-coded:** simple heuristic policy

- Report **average performance** over 1000 episodes
  - sanity check for new installations

Policy	3 vs. 2	4 vs. 3	5 vs. 4
Always Hold	**3.4**±1.5	**4.1**±1.8	**4.8**±2.2
Random	**7.5**±3.7	**8.3**±4.4	**9.5**±5.1
Hand-coded	**8.3**±4.7	**9.2**±5.2	**10.8**±6.7

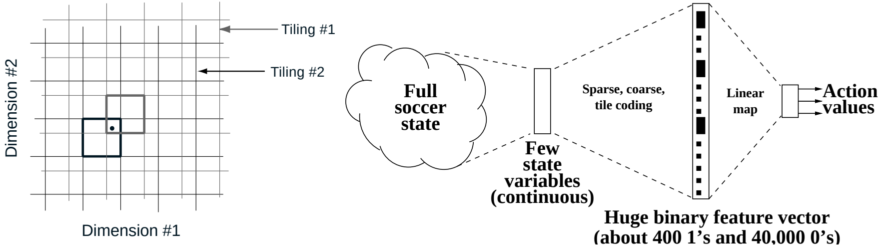
Experimental Setup

- Learn $Q^{\pi}(s, a)$: **Expected possession time**
- Sarsa(λ)
  - **On-policy method**: advantages over e.g. Q-learning
  - Not known to converge, but works (e.g. [Sutton, 1996])
- **3 Keepers** vs. **2 Takers**
  - Each keeper learns **separate value function**
- Compare different **function approximators**



CMAC tile-coding

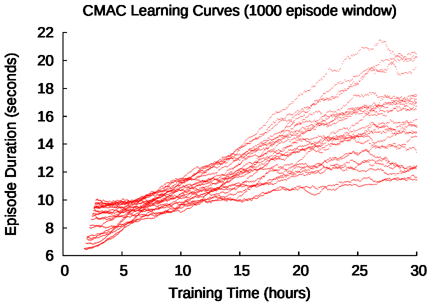
- Form of sparse, coarse coding based on **CMACs** [Albus, 1981]



- Tiled state variables **individually** (13)
- Linear FA on thousands of features

CMAC tile-coding

- 24 independent learning trials
- 1000 episode sliding window



- Reproduction of previous results

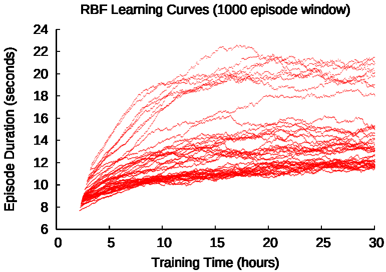
Radial Basis Functions

- Generalization of tile-coding to a **continuous function**
- **Gaussian:** $\phi(x) = \exp(-\frac{x^2}{2\sigma^2})$

- Better best-case performance than CMAC
  - between 5 and 25 hours
- Not significantly better on average

Radial Basis Functions

- Generalization of tile-coding to a **continuous function**
- **Gaussian:** $\phi(x) = \exp(-\frac{x^2}{2\sigma^2})$



- **Better best-case** performance than CMAC
  - between 5 and 25 hours
- Not significantly better on average

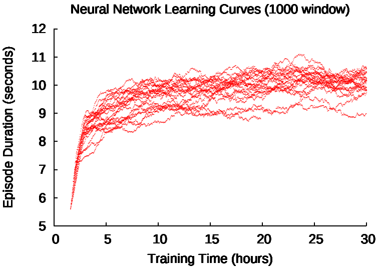
Neural Network

- **3 separate feed-forward networks** (1 per action)
  - 20 (sigmoid) nodes in hidden layer
  - Linear output node
- Trained with standard **back-propagation**

- Learned policies **not as good** as with CMAC or RBF
  - But **more consistent** (lower variance)
- Room for **parameter optimization**

Neural Network

- **3 separate feed-forward networks** (1 per action)
  - 20 (sigmoid) nodes in hidden layer
  - Linear output node
- Trained with standard **back-propagation**



- Learned policies **not as good** as with CMAC or RBF
  - But **more consistent** (lower variance)
- Room for parameter optimization

Conclusion and Future Work

- Keepaway can now be used as an **RL benchmark**
- **Benchmark Repository** publicly available since December at:
  `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/`
  - **Mailing list** has 16 (11 non-UT) subscribers
  - At least one outside researcher with learning results
- Possible **future uses** and **extensions**
  - Compare policy search and TD methods
  - Explore **multi-agent learning** questions
  - Different state and action representations
  - Transfer learning