Reinforcement Learning
Applications

Andrea Bonarini

Artificial Intelligence and Robotics Lab
Department of Electronics and Information
Politecnico di Milano

E-mail: bonarini@elet.polimi.it
URL:http://www.elet.polimi.it/~bonarini

Applications in many fields (1)

Robotics

- (***Quadruped Gait Control***) Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion by Nate Kohl and Peter Stone
- (***Quadruped Ball Acquisition***) Learning Ball Acquisition on a Physical Robot by Peggy Fidelman and Peter Stone
- (***Air Hockey***) Learning from Observation Using Primitives, and particularly the movie of a humanoid robot playing air hockey. An example paper.
- (***Active Sensing***) Active Sensing Using Reinforcement Learning by Cody Kwok and Dieter Fox.

Robot playing Air Hockey (1)

The game consists of two paddles, a puck and a board to play on. A human player using a mouse controls one paddle. At the other end is a cyber-human.

The following primitives have been explored:

- **Left Bank Shot** the player hits the puck, the puck hits the left wall once and then travels toward the goal.
- **Straight Shot** the player hits the puck, the puck travels straight toward the goal without hitting a wall.
- **Right Bank Shot** the player hits the puck, the puck hits the right wall once and then travels toward the goal.
- **Block** the player does not make a shot but attempts to block the puck from entering the player's goal area.
- **Setup** the player is positioning their paddle in preparation to make a shot.
- **Multi-shot** the player has blocked or made a shot and the puck does not have enough velocity to return to the other side of the board. Therefore the player has the opportunity to make another shot.

Robot playing Air Hockey (2)

Input for primitives

- XY location of the puck when it was hit
- velocity of the puck when it was hit
- absolute velocity of the puck after it was hit
- the point of the backwall that would be hit if the puck is not blocked

Output

- Paddle's velocity components when hit
- the location of the paddle relatively to the puck when in contact

Applications in many fields (2)

Control

- (***Helicopter control***) Inverted autonomous helicopter flight via reinforcement learning, by Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger and Eric Liang. In International Symposium on Experimental Robotics, 2004.
- (***Helicopter control***) Autonomous helicopter control using Reinforcement Learning Policy Search Methods, by J.A. Bagnell and J. Schneider. In Proceedings of the International Conference on Robotics and Automation, 2001.

Operations Research

- (***Pricing***) Opportunities and Challenges in Using Online Preference Data for Vehicle Pricing: A Case Study at General Motors by P. Rusmevichientong, J. A. Salisbury, L. T. Truss, B. Van Roy, and P. W. Glynn.
- (***Vehicle Routing***) Scaling Average-reward Reinforcement Learning for Product Delivery by S. Proper and P. Tadepalli.

Helicopter control (1)

First a stochastic, non-linear model of the helicopter has been build by supervised learning.

Reward function is a quadratic function of the error w.r.t. the position and speed of the helicopter

Monte Carlo learning on a NN model

Helicopter control (2)

Inverted fly control

(http://www.cs.stanford/edu/~ang/rl-videos/helicopter)

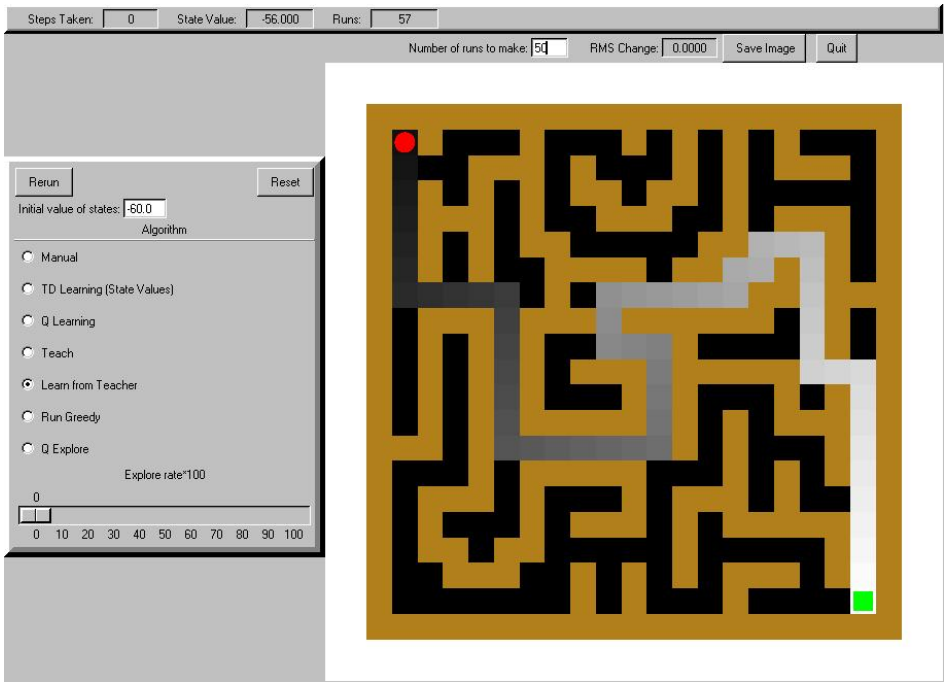
Applications in many fields (3)

Games

- (***Backgammon***) Temporal difference learning and TD-Gammon by Gerald Tesauro, Communications of the ACM, 38(3), March 1995.
- (***Solitaire***) Solitaire: Man Versus Machine, by X. Yan, P. Diaconis, P. Rusmevichientong, and B. Van Roy, to appear in Advances in Neural Information Processing Systems 17, MIT Press, 2005.
- (***Chess***) The KnightCap program, which went from a rating of 1600 to a rating of 2100 by altering its heuristic evaluation function using TD-lambda. CiteSeer has a link to the paper.
- (***Checkers***) Temporal Difference Learning Applied to a High-Performance Game-Playing Program by Jonathan Schaeffer, Markian Hlynka, and Vili Jussila, International Joint Conference on Artificial Intelligence (IJCAI), pp. 529-534, 2001...

Robot Maze (1)

This environment uses a very straightforward Q-learning algorithm. The robot decides on the action to perform by looking at the values of the next possible actions that can be taken from the current state.



Robot Maze (2)

The value of a state/action pair, Q(s,a), is the future discounted reward that the agent can expect to receive by taking action a from state s. Some examples of state/action pairs would be ((1,1), down) and ((1,3), up). The goal of the agent is to reach the goal in the shortest amount of steps. The agent receives a reward of -1 for each step that is taken. The value of the goal state is 0. The values are updated each time a move is made using the standard Q-learning function.

Used to study thepossibilities of Q-learning

TD-Gammon

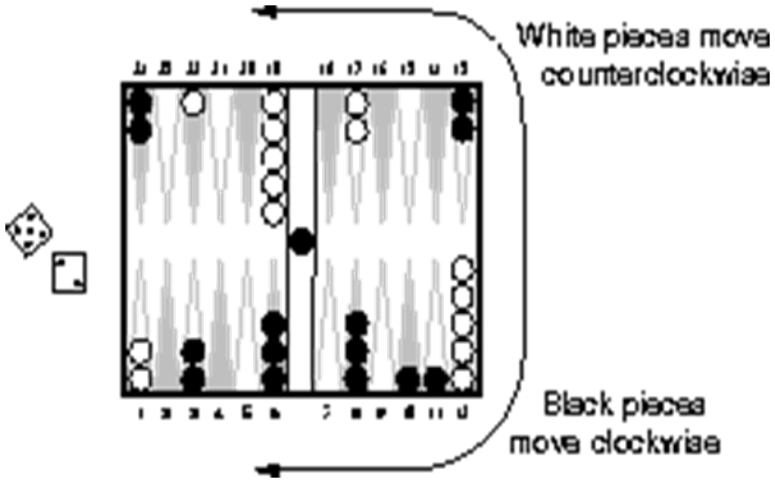
The problem

From (Tesauro, 1992, 1994, 1995)

Play backgammon

Backgammon is a major game, played by more people than chess. Both chance and strategy are important.

In this figure, white has just rolled the dice and obtained a 5 and a 2. This means that he can move one of his pieces 5 steps and one (possibly the same piece) 2 steps. For example, he could move two pieces from the 12 point, one to the 17 point, and one to the 14 point. White's objective is to advance all of his pieces into the last quadrant (points 19-24) and then off the board. The first player who removes all his pieces wins. One complication is that the pieces interact as they pass each other going in different directions. For example, if it were black's move in the figure, he could use the dice roll of 2 to move a piece from the 24 point to the 22 point, ``hitting" the white piece there. Pieces that have been hit are placed on the ``bar" in the middle of the board (where we already see one previously hit black piece), from whence they re-enter the race from the start. However, if there are two pieces on a point, then the opponent cannot move to that point; the pieces are protected from being hit. Thus, white cannot use his 5-2 dice roll to move either of his pieces on the 1 point, because their possible resulting points are occupied by groups of black pieces. Forming contiguous blocks of occupied points to block the opponent is one of the elementary strategies of the game.



TD-Gammon: RL formulation

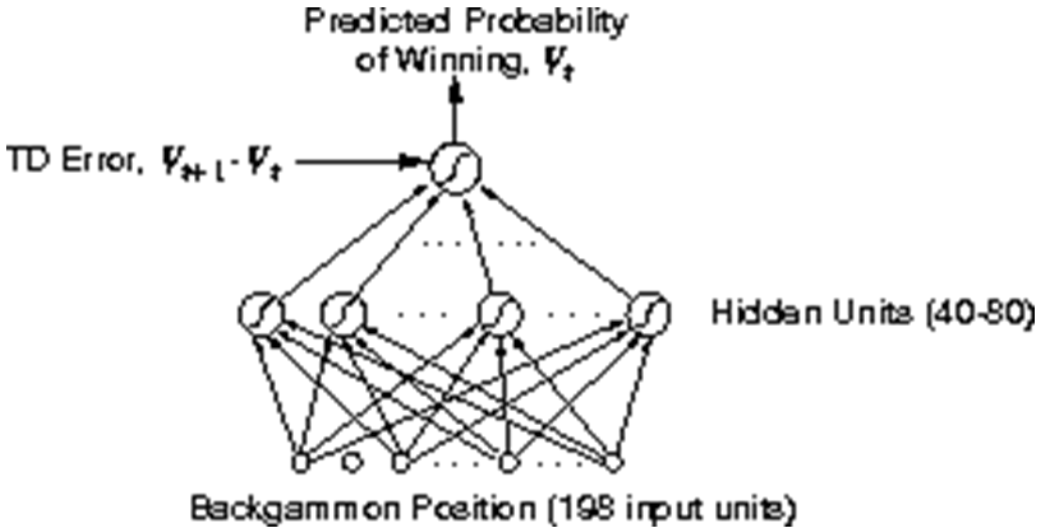
The **state** is represented as follows.

For each point on the backgammon board, 4 units indicate the number of white pieces on the point. If there were no white pieces, then all 4 units took on the value zero. If there was one piece, then the first unit took on the value 1. If there were two pieces, then both the first and the second unit were 1. If there were three or more pieces on the point, then all of the first three units were 1. If there were more than three pieces, the fourth unit also came on, to a degree indicating the number of additional pieces beyond three. Two additional units encode the number of white and black pieces on the bar, and two more encode the number of black and white pieces already successfully removed from the board. Finally, two units indicate in a binary fashion whether it was white's or black's turn to move.

The **decision/control** is the move to take, based on the estimation of the move.

TD-Gammon: Model

The model of the system is approximated by a two-layer neural network, trained by TD(0). In input is the state as described above. There are a total of 198 input units to the network. In output the estimate of the value of the input configuration.



At each step, the weights of the NN are updated by gradient descent on the square error of J:

$$(J_{t+1}-J_t)^2$$

TD-Gammon: TD(λ) solution

TD(λ) is used, by updating the weights of the network by:

$$\vec{\vartheta}_{t+1}(x) = \vec{\vartheta}_t(x) + \alpha\left[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right]\vec{e}_t$$

where x is a configuration, and e is the vector of eligibility traces updated by:

$$\vec{e}_t = \gamma\vec{e}_{t-1} - \nabla_{\vec{\vartheta}_t} V_t(s_t)$$

where the gradient is computed by backpropagation.

In TD-Gammon γ=1 and the reward is always zero except when the player wins, where it takes 100 or loses (-100).

TD-Gammon: Results

Search space is about 10^{20} states.

It required 1.500.000 games to learn, most of which generated by itself.

TD-Gammon 3.0 plays better than the world champions, and also suggested to them some openings different from the ones used up to then.

Applications in many fields (4)

Human-Computer Interaction

- (***Spoken Dialogue Systems***) Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. S. Singh, D. Litman, M. Kearns and M. Walker. In Journal of Artificial Intelligence Research (JAIR), Volume 16, pages 105-133, 2002
- (***Software Agent in MOOs***) Cobot: A Social Reinforcement Learning Agent. C. Isbell, C. Shelton, M. Kearns, S. Singh, and P. Stone (2002). In Proceedings of Neural Information Processing Systems 14 (NIPS), pp. 1393-1400.

Economics/Finance

- (***Trading***) Learning to Trade via Direct Reinforcement. John Moody and Matthew Saffell, IEEE Transactions on Neural Networks, Vol 12, No 4, July 2001.

Applications in many fields (5)

Complex Simulation

- (***Robot_Soccer***) Scaling Reinforcement Learning toward RoboCup Soccer, by Peter Stone and Richard S. Sutton, Proceedings of the Eighteenth International Conference on Machine Learning, pp. 537–544, Morgan Kaufmann, San Francisco, CA, 2001.

Marketing

- (***Targeted_Marketing***) Cross Channel Optimized Marketing by Reinforcement Learning, by Naoki Abe, Naval Verma, Chid Apte and Robert Schroko, Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2004.

Telecommunications

- (**Channel allocation on cell phone systems**) Reinforcement Learning for Dynamic Channel Allocation in Cellular Telephone Systems Satinder Singh, Dimitri Bertsekas, Advances in Neural Information Processing Systems (1997)

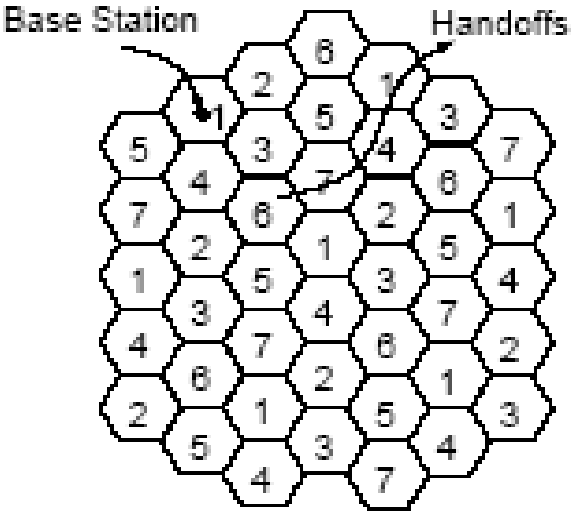
Channel allocation in cell phone systems

From (S. Singh, D. Bartsekas, 1996)

The problem

Allocate channels in cell phone systems

The market area is divided up into cells, shown here as hexagons. The available bandwidth is divided into channels.
Each cell has a base station responsible for calls within its area. Calls arrive randomly, have random durations and callers may move around in the market area creating handoffs.
The channel reuse constraint requires that there be a minimum distance between simultaneous reuse of the same channel.



RL formulation

The **state** is represented as:

- The list of occupied and unoccupied channels at each cell. This is the configuration of the cellular system. It is exponential in the number of cells.
- The event that causes the state transition (arrival, departure, or handoff). This component of the state is uncontrollable.

The **decision/control** applied at the time of a call departure is the reassignment of the channels in use with the aim of creating a more favorable channel packing pattern among the cells (one that will leave more channels free for future assignments).

Optimization function

We have to maximize

$$J = E\left\{\int_0^{\infty} e^{-\beta t} c(t)\,dt\right\}$$

where E{.} is the expectation operator, c(t) is the number of ongoing calls at time t, and β is a discount factor that makes immediate profit more valuable than future profit. Maximizing J is equivalent to minimizing the expected (discounted) number of blocked calls over an infinite horizon.

A TD(0) solution

TD(0) is used, by updating the estimate of J with:

$$J_{t+1}(x) = (1-\alpha)J_t(x) + \alpha \max_{a \in A(x,e)} \left[c(x,a,\Delta t) + \gamma(\Delta t) J_{t+1}(y) \right]$$

where x is a configuration, e is the random event (a call arrival or departure), A(x, e) is the set of actions available in the current state (x, e), Δt is the random time until the next event, c(x, a, Δt) is the effective immediate payoff with the discounting, and γ (Δt) is the effective discount for the next configuration y.

Decisions

Call Arrival:

When a call arrives, evaluate the next configuration for each free channel and assign the channel that leads to the configuration with the largest estimated value. If there is no free channel at all, no decision has to be made.

Call Termination:

When a call terminates, one by one each ongoing call in that cell is considered for reassignment to the just freed channel; the resulting configurations are evaluated and compared to the value of not doing any reassignment at all. The action that leads to the highest value configuration is then executed.

Model

The model of the system is approximated by a linear neural network, trained by TD(0). In input are the number of free channels for each cell, and the number of times a channel is used in a four cells radius, for each cell-channel pair.

The problem is exponential and the state space for a 7x7 grid consists of about 70^{49} states.

At each step, the weights of the NN are updated by gradient descent on the square error of J:

$$(J_{t+1}-J_t)^2$$

Tests and results

Demo: http://www.eecs.umich.edu/~baveja/Demo.html

Graphs from (S. Singh, D. Bartsekas, 1996)

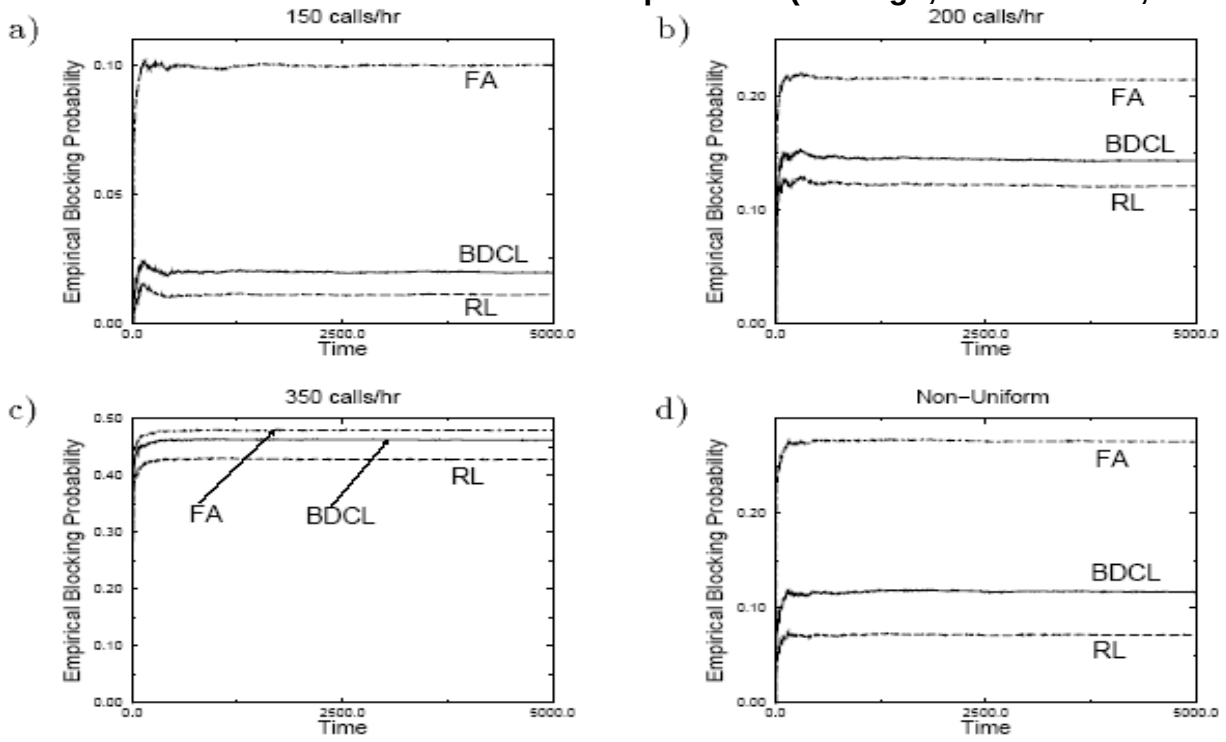


Figure 2: a), b), c) & d) These figures compare performance of RL, FA, and BDCL on the 7 by 7 cellular array of Figure 1a. The means of the call arrival (Poisson) processes are shown in the graph titles. Each curve presents the cumulative empirical blocking probability as a function of time elapsed in minutes. All simulations start with no ongoing calls and therefore the blocking probabilities are low in the early minutes of the performance curves. The RL curves presented here are for a linear function approximator and show performance while learning. Note that learning is quite rapid.