Slide 0

Lecture 3.2

Applications of Reinforcement Learning

Jonathan Shapiro

Department of Computer Science, University of Manchester

February 4, 2003

Slide 1

Applications: TD-Gammon

Tesauro, 1994

TD(λ) learning has been used to produce a very good backgammon playing program.

Predictor Network: A multi-layer perceptron is used to predict the outcome of the game from the current board position (originally coded as raw board position; later hand-crafted features used).

Controller: A program generates all legal moves from the current position. Predictor network scores them; that with the highest predicted outcome (J) is the moved used.

Slide 2

Training: Data consists of sequences of moves from the standard start position to a result. Program plays itself over and over.

1. Network starts from random state (no knowledge of game)
2. Initially, games incredibly long (100's to 1000's of moves versus 50 – 60 moves for typical human game).
3. Later, basic strategies emerge: hitting opponents, building points, playing safe, ...

Results: TD-Gammon 1.0 Contained 40 hidden units, trained for 200,000 games. Plays at strong intermediate level; good enough to win regional tournaments.

Results: TD-Gammon 2.1 Close to the world's best player.

Has changed how humans play certain board positions (discovered new strategies which appear to be better).

Slide 3

TD-Gammon (cont)

"TD-Gammon has definitely come into its own. There is no question in my mind that its positional judgment is far better than mine. Only on small technical areas can I claim a definite advantage over it . . . . I find a comparison of TD-Gammon and the high-level chess computers fascinating. The chess computers are tremendous in tactical positions where variations can be calculated out. Their weakness is in vague positional games, where it is not obvious what is going on . . . . TD-Gammon is just the opposite. Its strength is in the vague positional battles where judgment, not calculation , is the key. There, it has a definite edge over humans . . . . In particular, its judgment on bold vs. safe play decisions, which is what backgammon really is all about, is nothing short of phenomenal . . . . Instead of a dumb machine which can calculate things much faster than humans such as the chess playing computers, you have built a smart machine which learns from experience pretty much the same way that humans do"

Kit Woolsey, perennial world top 10 backgammon player (ranked #3 when this was written), quoted in Tesauro, 1995.

Slide 4

Elevator (lift) Displatching

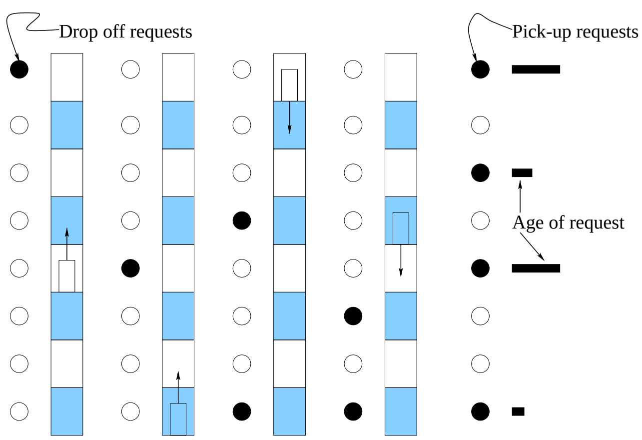
Q-learning has been shown to find successfull strategies for distributing elevators among floors in simulations.

The situation: A multi-story hotel with four elevators, filled with people who want to go from upper floors to the lobby (typically), from the lobby to upper floors, and between floors (occasionally).

The problem: People can wait a long time if there are too many requests, not enough elevators, or *a poor strategy for controlling the elevators*.

Performance criteria: *Mean waiting time* to get into an elevator, total time to get to destination, fraction of passengers who wait more than one minute.

Slide 5



Slide 6

Reinforcement-learning approach

Crites and Barto, 1996. Assumptions

- Each elevator makes an independent decision;
- an elevator cannot pass a floor if it contains passengers wanting to get off there;
- an elevator cannot change directions until all passengers wanting to go in that direction get off;
- an elevator cannot stop at a floor unless there is a request there;
- an elevator cannot stop to pick up passengers if an elevator is already stopped there;
- given a choice of going up or down, go up.

Thus, the only decision occurs when approaching a floor with a pick-up request, whether to stop or not.

Slide 7

Crites and Barto used 1-step Q-learning.

Reward: was the mean-squared waiting time between two subsequent times requiring decisions (discounted).

- Using squared waiting time requires times to be short, but also penalizes more long times.
- True waiting time not known because number of people waiting at a stop unknown. They tried two approaches:
  1. assumed it was known during training (gave best results)
  2. inferred under assumptions about distribution of requests.

Slide 8

Neural Network: 47-20-2 MLP

Inputs:

- Inputs represented the hall request buttons. Real-valued representation of wait time; binary representation of pressed/not pressed;
- one-of-n encoding of current location and direction of elevator being controlled;
- coarse-grained representation of location, direction, and speed of other elevators;
- binary input about whether the elevator was at the highest floor with a passenger waiting;
- binary input about whether the elevators was at a floor with the longest wait time.

Slide 9

Output: State-action value Q.

Two controller architectures: RL1: A network for each elevator (although one reinforcement signal for all). RL2: A network for all elevators.

Slide 10

Results

Several conditions. Below is for "down-peak profile" with small number of up traffic. .

Algorithm	ave wait	squared wait	ave total time	percent ¿60s
SECTOR	27.3	1252	54.8	9.24
LQF	21.9	732	50.7	2.87
HUFF	19.6	608	50.5	1.99
RL1	16.9	476	42.7	1.53
RL2	16.9	468	42.7	1.40

SECTOR: A sector-based method, similar to those used in real systems.

LQF: Longest queue first, gives priority to the person waiting the longest;

HUFF: Highest unanswered floor first, gives priority to highest floor with people waiting.

Slide 11

Classical Conditioning

References: Moore et. al. 1990; Desmond 1990, Sutton and Barto 1986.

TD-learning was motivated in part by classical conditioning experiments.

Helped to explain some mysteries in that fundamental aspect of animal learning.

Slide 12

Classical Conditioning

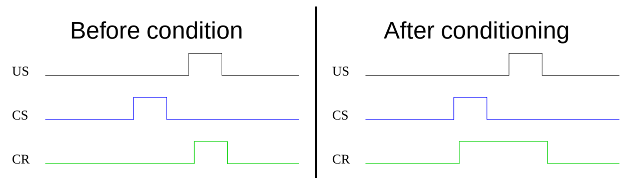
Unconditioned Stimulus(US): Stimulus which would cause response in the untrained animal.

Conditioned Stimulus (CS): Stimulus caused response only due to training.

Conditioned Response CR: The response.

Famous example discovered by Pavlov. Ringing of bell (CS) followed by food (US) resulted in salivation (CR). Other experimental procedures: CS followed by puff of air directed at eye cause animal to close eye in response to CS.

Slide 13



Question: how does the CS and the CR become associated across time?

Slide 14

Blocking

- After conditioning on stimulus A, US is preceded by *simultaneous* presence of stimuli A and B.

  Result: *Not* conditioned on B.
- However, if B precedes A, then conditioning on A is lost and US becomes conditioned on B. (More complicated scenarios also studied).

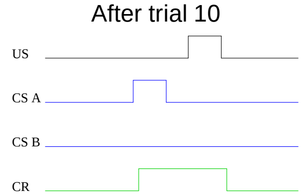
Question: what prevents the association with stimulus B?

Slide 15

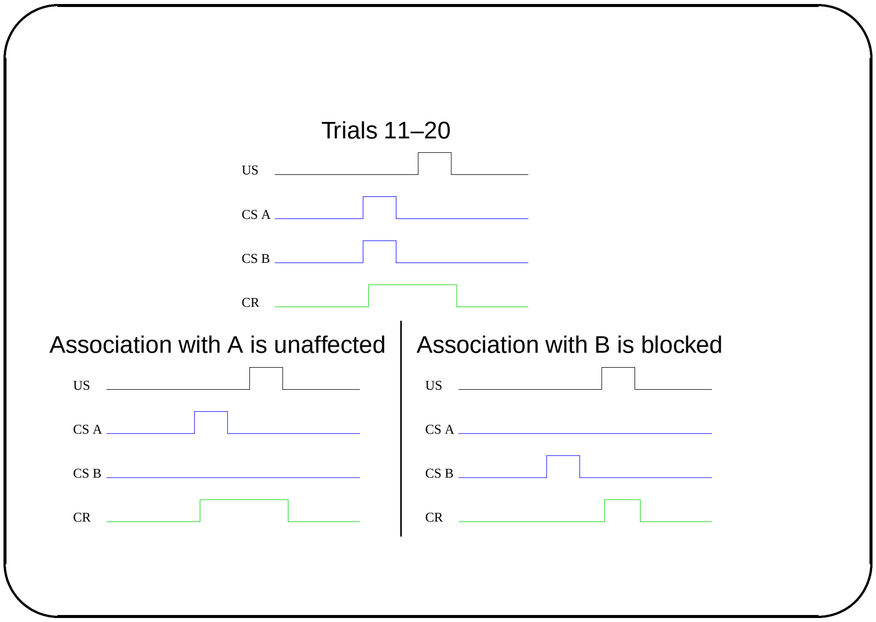
Blocking Experiments

First 10 trials associate A with the US.

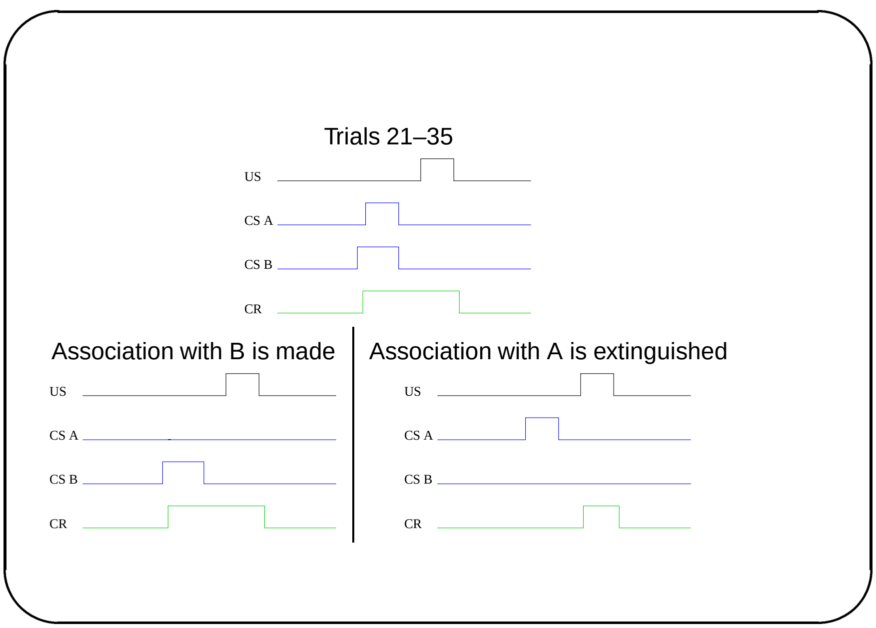
After trial 10



Slide 16



Slide 17



Slide 18

TD model of classical conditioning

The principles of TD learning can explain these puzzles:

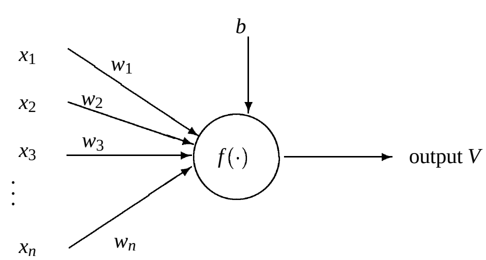
- the animal associates the stimulus with the (discounted) expected future reward,
- the eligibility for the stimulus to participate in learning persists, decaying amount λ per unit time.

Slide 19

A simple neural network model

Inputs Possible stimuli $x_i(t)$.

Output: V represents the predicted value of the response. $V = \sum_i w_i x_i$ or possible related through a sigmoid function.



Slide 20

Learning Rule

$$\Delta w_i = \alpha \left[r(t) + \gamma f(\sum_j w_j(t) x_j(t+1)) - f(\sum_i w_j(t) x_j(t)) \right] \overline{X_i(t)}$$

Here r is the reward, $f(\cdot)$ is the output function for the neural network, and the eligibility trace obeys,

$$\overline{X_i(t)} = \gamma\lambda \overline{X_i(t-1)} + x_i(t) f'(\sum_i w_j(t) x_j(t)).$$

Slide 21

This is precisely the TD learning rule for neural networks used in TD-gammon.

Hebbian like rule, where input activation replaced by slowly decaying eligibility trace, δ is a difference between outputs at different times.

Slide 22

How this gives classical results

1. Eligibility trace cause information about CS to persist.
2. Temporal difference rule causes information about the CR (*r*) to act during earlier times.

Slide 23

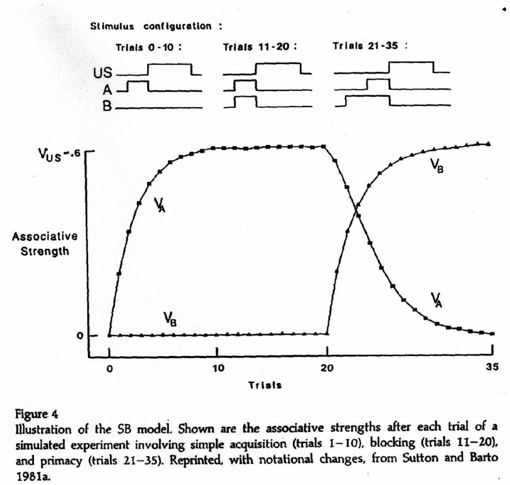


Figure 4
Illustration of the SB model. Shown are the associative strengths after each trial of a simulated experiment involving simple acquisition (trials 1–10), blocking (trials 11–20), and primacy (trials 21–35). Reprinted, with notational changes, from Sutton and Barto 1981a.

Slide 24

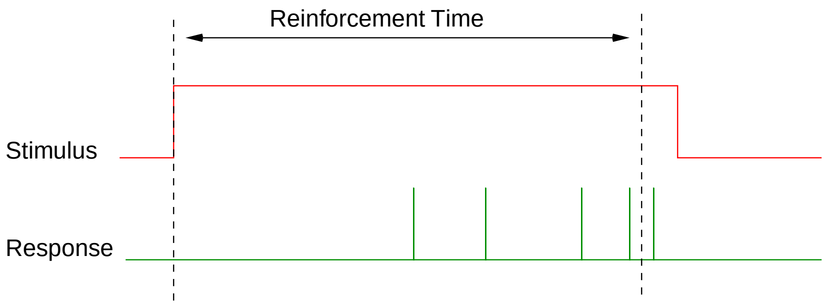
Learning expected time to a reward

- One aspect of delayed-reward reinforcement learning involves learning the expected time to wait for the reward.
- This may be useful for determining when the expected reward is not forthcoming (e.g. the foraging patch is deleted and it is time to move on), comparing reinforcement rates, etc.
- Numerous experiments that animals can do this, and do it in a characteristic and striking manner — they exhibit *scalar timing*.

Slide 25

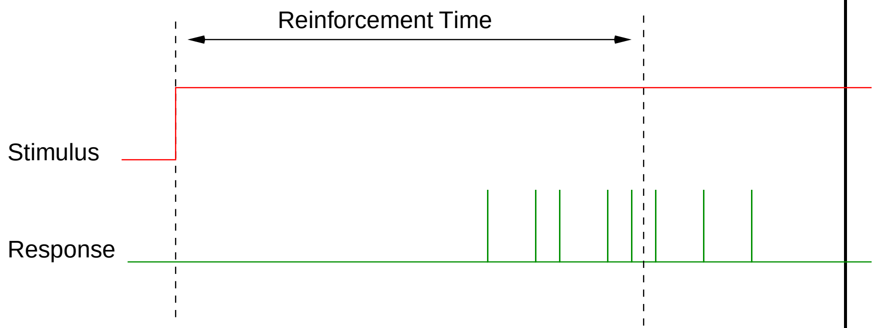
Peak procedure experiments

Reward Trials (training) Stimulus (bell, light) comes on to indicate start of trial. Animal is rewarded for the first response after a reinforcement time has passed.



Slide 26

Non-reward Trials Stimulus remains on even after animal responds after reinforcement time has passed. This is to see for how long the animal continues responding.



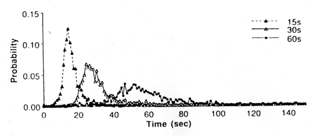
Slide 27

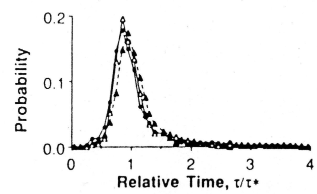
Results on non-reward trials — the scalar property

Responses averaged over many non-reward trials give a smooth curve. Studies of the curves for different reinforcement times show the following features:

- The highest rate of response is at the expected reinforcement time.
- Scaled curves from different reinforcement times collapse when scaled relative to the reinforcement time

Slide 28





Slide 29

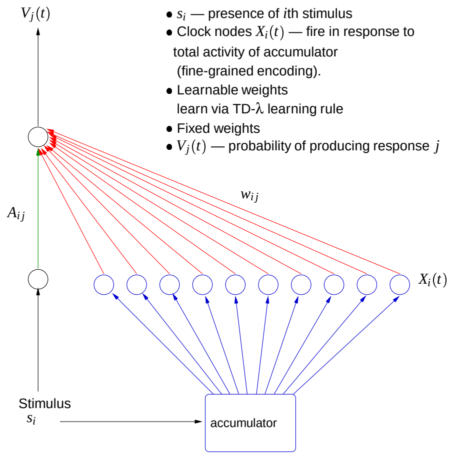
Let T be the actual time since the start of the trial and $\tilde{T}$ be subjective time.

$$\boxed{P(\tilde{T}|T) \approx \frac{1}{T} P_{\text{inv}}\left(\frac{\tilde{T}}{T}\right)}. \qquad (1)$$

Here P_{inv} is the function which describes the shape of the scaled curves; it is peaked near one. *This is the scalar property of interval timing.*

Slide 30

Reinforcement Learning of Time Intervals



- s_i — presence of ith stimulus
- Clock nodes $X_i(t)$ — fire in response to total activity of accumulator (fine-grained encoding).
- Learnable weights learn via TD-λ learning rule
- Fixed weights
- $V_j(t)$ — probability of producing response j

Slide 31

Learning model

Time perception system: consists of

- Accumulator network — measures time as accumulation of neural spikes (presented in Shapiro and Wearden, 2001).
- Clock nodes $X_i(t)$ — fire in response to total activity of the accumulator.

Slide 32

Stimulus-response: Both the stimulus and the appropriate clock node must be present for there to be reasonable chance of response.

- Response nodes $V_j(t)$ — determine the response. Response j is triggered with probability $\min[V_j(t), 1]$,

$$V_j(t) = \left\lfloor \sum_i w_{ij} X_i(t) + A_{ij} s_i - \theta_j \right\rfloor, \tag{2}$$

  where $\lfloor \cdot \rfloor$ denotes a function which is the identity for positive argument, zero otherwise. The threshold θ_j is constant.

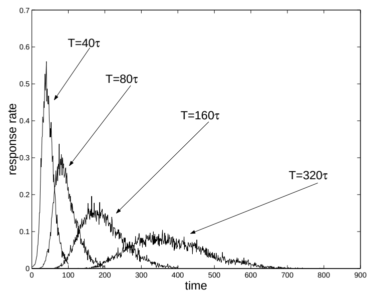
Reinforcement learning: Learnable weights learn via TD-λ learning rule.

Slide 33

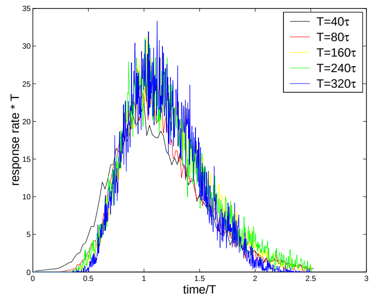
Simulations of Peak Procedure

Method: 50 trials with forced response followed by 1150 trials. Every 10th one is a non-reinforced trial. Learning takes place throughout. Results averaged over the last 100 non-reward trials (i.e. the last 1000 trials).

Slide 34



Response per unit time for different reinforcement times T. Parameters: $m_I = 10$, $C\sigma_v^2 = 1$ (Poisson limit), $\gamma = 0.75$, $\lambda = 1$, $\alpha = 0.5$.



Response per relative time versus relative time for different reinforcement times T. The data collapses, clearly showing the scalar property (equation (1)).

Slide 35

Conclusions

- Reinforcement learning is an important paradigm of adaptive computation.
- It is often useful to separate learning the expected reward, and the policy for choosing actions.
- Heuristics which are important for delayed reward problems are:
  - Learn the discounted reward,
  - associate action with reward by discounting a factor λ per unit time.
- These heuristics result in useful examples of AI (TD-gammon) and effective models of animal experiments.