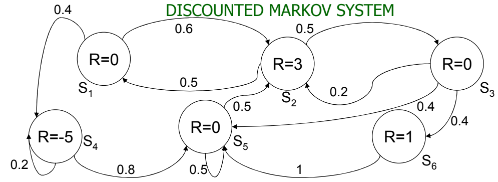
Reinforcement Learning

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

Andrew W. Moore
Associate Professor
School of Computer Science
Carnegie Mellon University
www.cs.cmu.edu/~awm
awm@cs.cmu.edu
412-268-7599

 April 23rd, 2002

Predicting Delayed Rewards IN A DISCOUNTED MARKOV SYSTEM



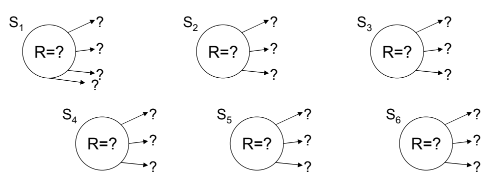
Prob(next state = S_5|this state = S_4) = 0.8 etc…

What is expected sum of future rewards (discounted) ?

$$E\left[\left(\sum_{t=0}^{\infty}\gamma^t R(S[t])\right) \;\middle|\; S[0]=S\right]$$

Just Solve It! We use standard Markov System Theory

Learning Delayed Rewards…



All you can see is a series of states and rewards:

$S_1(R=0) \rightarrow S_2(R=0) \rightarrow S_3(R=4) \rightarrow S_2(R=0) \rightarrow S_4(R=0) \rightarrow S_5(R=0)$

Task: Based on this sequence, estimate $J^*(S_1), J^*(S_2) \cdots J^*(S_6)$

Idea 1: Supervised Learning

Assume $\gamma=1/2$

$S_1(R=0) \rightarrow S_2(R=0) \rightarrow S_3(R=4) \rightarrow S_2(R=0) \rightarrow S_4(R=0) \rightarrow S_5(R=0)$

At t=1 we were in state S_1 and eventually got a long term discounted reward of $0+\gamma 0+\gamma^2 4+\gamma^3 0+\gamma^4 0\ldots = 1$

At t=2 in state S_2 ltdr = 2
At t=3 in state S_3 ltdr = 4
At t=4 in state S_2 ltdr = 0
At t=5 in state S_4 ltdr = 0
At t=6 in state S_5 ltdr = 0

State	Observations of LTDR	Mean LTDR	
S_1	1	1	$=J^{est}(S_1)$
S_2	2 , 0	1	$=J^{est}(S_2)$
S_3	4	4	$=J^{est}(S_3)$
S_4	0	0	$=J^{est}(S_4)$
S_5	0	0	$=J^{est}(S_5)$

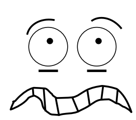
Supervised Learning ALG

- Watch a trajectory
  S[0] r[0] S[1] r[1] ···· S[T]r[T]
- For t=0,1, ··· T , compute $J[t] = \sum_{i=0}^{\infty}\gamma^i r[t+i]$
- Compute $J^{est}(S_i) = \left(\begin{array}{c}\text{mean value of } J[t] \\ \text{among all transitions beginning} \\ \text{in state } S_i \text{ on the trajectory}\end{array}\right)$

  Let $\text{MATCHES}(S_i) = \{t \mid S[t] = S_i\}$, then define

  $$J^{est}(S_i) = \frac{\sum_{t\in \text{MATCHES}(S_i)} J[t]}{|\text{MATCHES}(S_i)|}$$
- You're done!

Supervised Learning ALG for the timid



If you have an anxious personality you may be worried about edge effects for some of the final transitions. With large trajectories these are negligible.

Online Supervised Learning

Initialize: Count[S_i] = 0 $\forall S_i$
SumJ[S_i] = 0 $\forall S_i$
Eligibility[S_i] = 0 $\forall S_i$

Observe:

When we experience S_i with reward r do this:

$\forall j$ Elig[S_j] $\leftarrow \gamma$Elig[S_j]
Elig[S_i] $\leftarrow$ Elig[S_i] + 1

$\forall j$ SumJ[S_j] $\leftarrow$ SumJ[S_j]+r$\times$Elig[S_j]
Count[S_i] $\leftarrow$ Count[S_i] + 1

Then at any time,
$J^{est}(S_j)$= SumJ[S_j]/Count[S_j]

Online Supervised Learning Economics

Given N states $S_1 \cdots S_N$, OSL needs O(N) memory.

Each update needs O(N) work since we must update all Elig[ ] array elements

Idea: Be sparse and only update/process Elig[ ] elements with values >ξ for tiny ξ

There are only $\log\left(\frac{1}{\xi}\right) \Big/ \log\left(\frac{1}{\gamma}\right)$ such elements

Easy to prove:

$$\text{As } T \to \infty,\ J^{est}(S_i) \to J^*(S_i) \quad \forall S_i$$

Online Supervised Learning

COMPLAINT

Let's grab OSL off the street, bundle it into a black van, take it to a bunker and interrogate it under 600 Watt lights.

$S_1(r=0) \to S_2(r=0) \to S_3(r=4) \to S_2(r=0) \to S_4(r=0) \to S_5(r=0)$

State	Observations of LTDR	$\hat{J}(S_i)$
S_1	1	1
S_2	2 , 0	1
S_3	4	4
S_4	0	0
S_5	0	0

There's something a little suspicious about this (efficiency-wise)

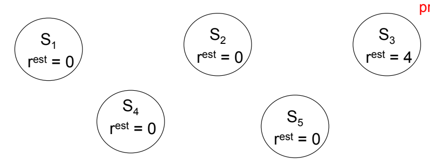
Certainty-Equivalent (CE) Learning

Idea: Use your data to estimate the underlying Markov system, instead of trying to estimate J directly.

$S_1(r=0) \to S_2(r=0) \to S_3(r=4) \to S_2(r=0) \to S_4(r=0) \to S_5(r=0)$

Estimated Markov System:

You draw in the transitions + probs



What're the estimated J values?

C.E. Method for Markov Systems

Initialize:

Count[S_i] = 0 $\forall S_i$ #Times visited S_i
SumR[S_i] = 0 Sum of rewards from S_i
Trans[S_i,S_j] = 0 $\forall S_j$ #Times transitioned from $S_i \to S_j$

When we are in state S_i , and we receive reward r , and we move to S_j …

Count[S_i] $\leftarrow$Count[S_i] + 1
SumR[S_i] $\leftarrow$SumR[S_i] + r
Trans[S_i,S_j] $\leftarrow$ Trans[S_i,S_j] + 1

Then at any time

$r^{est}(S_j)$ = SumR[S_j] / Count[S_j]
P^{est}_{ij} = Estimated Prob(next = S_j | this = S_i)
= Trans[S_i,S_j] / Count[S_i]

C.E. for Markov Systems (continued) …

So at any time we have

$r^{est}(S_j)$ and P^{est} (next=S_j | this=S_i)
$\forall S_i S_j$ = P^{est}_{ij}

So at any time we can solve the set of linear equations

$$J^{est}(S_i) = r^{est}(S_i) + \gamma \sum_{S_j} P^{est}(S_j|S_i) J^{est}(S_j)$$

[In vector notation,
$\mathbf{J}^{est} = \mathbf{r}^{est} + \gamma \mathbf{P}^{est} \mathbf{J}$
=> $\mathbf{J}^{est} = (\mathbf{I}-\gamma\mathbf{P}^{est})^{-1}\mathbf{r}^{est}$
where $\mathbf{J}^{est}$ $\mathbf{r}^{est}$ are vectors of length N
$\mathbf{P}^{est}$ is an NxN matrix
N = # states]

C.E. Online Economics

Memory: $O(N^2)$

Time to update counters: $O(1)$

Time to re-evaluate J^{est}

- $O(N^3)$ if use matrix inversion
- $O(N^2 k_{CRIT})$ if use value iteration and we need k_{CRIT} iterations to converge
- $O(N k_{CRIT})$ if use value iteration, and k_{CRIT} to converge, and M.S. is Sparse (i.e. mean # successors is constant)

Certainty Equivalent Learning

COMPLAINT

Memory use could be $O(N^2)$!

And time per update could be $O(N k_{CRIT})$ up to $O(N^3)$!

Too expensive for some people.

Prioritized sweeping will help, (see later), but first let's review a very inexpensive approach

Why this obsession with onlineiness?

I really care about supplying up-to-date J^{est} estimates all the time.

Can you guess why?

If not, all will be revealed in good time…

Less Time: More Data Limited Backups

- Do previous C.E. algorithm.
- At each time timestep we observe $S_i(r) \rightarrow S_j$ and update Count[S_i], SumR[S_i], Trans[S_i,S_j]
- And thus also update estimates

$$r_i^{est} \text{ and } P_{ij}^{est} \quad \forall_j \in \text{outcomes}(S_i)$$

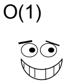
But instead of re-solving for J^{est}, do much less work. Just do one "backup" of $J^{est}[S_i]$

$$J^{est}[S_i] \leftarrow r_i^{est} + \gamma \sum_j P_{ij}^{est} J^{est}[S_j]$$

"One Backup C.E." Economics

Space : $O(N^2)$ NO IMPROVEMENT THERE!

Time to update statistics : $O(1)$

Time to update J^{est} : $O(1)$ 

- Good News: Much cheaper per transition
- Good News: Contraction Mapping proof (modified) promises convergence to optimal
- Bad News: Wastes data

Prioritized Sweeping

[Moore + Atkeson, '93]

Tries to be almost as data-efficient as full CE but not much more expensive than "One Backup" CE.

On every transition, some number (β) of states may have a backup applied. Which ones?

- The most "deserving"
- We keep a priority queue of which states have the biggest potential for changing their $J^{est}(Sj)$ value

Where Are We?

Trying to do online J^{est} prediction from streams of transitions

Data Efficiency:

	Space	J^{est} Update Cost
Supervised Learning	$0(N_s)$	$0(\frac{1}{\log(1/\gamma)})$
Full C.E. Learning	$0(N_{so})$	$0(N_{so}N_s)$ $0(N_{so}k_{CRIT})$
One Backup C.E. Learning	$0(N_{so})$	0(1)
Prioritized Sweeping	$0(N_{so})$	0(1)

N_{so}= # state-outcomes (number of arrows on the M.S. diagram)

N_s= # states

What Next ?
Sample Backups !!!

Temporal Difference Learning

[Sutton 1988]

Only maintain a J^{est} array… nothing else

So you've got

$J^{est}(S_1)$ $J^{est}(S_2)$, ··· $J^{est}(S_N)$

and you observe

$S_i \xrightarrow{r} S_j$

A transition from i that receives an immediate reward of r and jumps to j

what should you do?

Can You Guess ?

TD Learning

$S_i \xrightarrow{r} S_j$

We update $= J^{est}(S_i)$

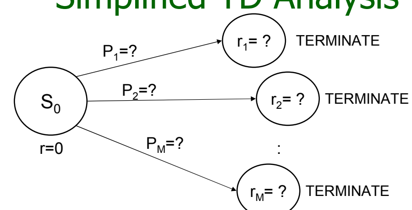
We nudge it to be closer to expected future rewards

$$\left.\begin{aligned} J^{est}(S_i) \leftarrow & (1-\alpha)J^{est}(S_i) + \\ & \alpha\left[\text{Expected future rewards}\right] \end{aligned}\right\} \text{WEIGHTED SUM}$$

$$= (1-\alpha)J^{est}(S_i) + \alpha\left[r + \gamma J^{est}(S_j)\right]$$

α is called a "learning rate" parameter. (See "η" in the neural lecture)

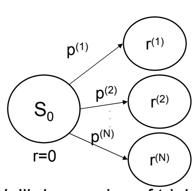
Simplified TD Analysis



- Suppose you always begin in S_0
- You then transition at random to one of M places. You don't know the transition probs. You then get a place-dependent reward (unknown in advance).
- Then the trial terminates.

Define $J^*(S_0)$= Expected reward

Let's estimate it with TD



$r^{(k)}$ = reward of k'th terminal state

$p^{(k)}$ = prob of k'th terminal state

We'll do a series of trials. Reward on t'th trail is r_t

$$= E[r_t] = \sum_{k=1}^{n} p^{(k)} r^{(k)} \quad [\text{Note } E[r_t] \text{ is independent of } t]$$

Define $J^*(S_0) = J^* = E[r_t]$

Let's run TD-Learning, where

J_t = Estimate $J^{est}(S_0)$ before the t'th trial.

From definition of TD-Learning:

$J_{t+1} = (1-\alpha)J_t + \alpha r_t$

Useful quantity: Define

$$\sigma^2 = \text{Variance of reward} = E\left[(r_t - J^*)^2\right]$$

$$= \sum_{k=1}^{M} P^{(k)} \left(r^{(k)} - J^*\right)^2$$

Remember $J^* = E[r_t]$, $\sigma^2 = E[(r_t-J^*)^2]$
$J_{t+1} = \alpha r_t + (1-\alpha)J_t$

$$E[J_{t+1} - J^*] =$$
$$= E[\alpha r_t + (1-\alpha)J_t - J^*]$$

WHY?

$$= (1-\alpha)E[J_t - J^*]$$

Thus...

$$\lim_{t\to\infty} E[J_t] = J^*$$

Is this impressive??

Remember $J^* = E[r_t]$, $\sigma^2 = E[(r_t-J^*)^2]$
$J_{t+1} = \alpha r_t + (1-\alpha)J_t$

Write S_t = Expected squared error between J_t and J^* before the t'th iteration

$S_{t+1} = E[(J_{t+1}-J^*)^2]$
$= E[(\alpha r_t+(1-\alpha)J_t - J^*)^2]$
$= E[(\alpha[r_t-J^*]+(1-\alpha)[J_t - J^*])^2]$
$= E[\alpha^2(r_t-J^*)^2+\alpha(1-\alpha)(r_t-J^*)(J_t - J^*)+(1-\alpha)^2(J_t - J^*)^2]$
$= \alpha^2E[(r_t-J^*)^2]+\alpha(1-\alpha)E[(r_t-J^*)(J_t - J^*)]+(1-\alpha)^2E[(J_t - J^*)^2]$
$=$
$= \alpha^2\sigma^2+(1-\alpha)^2S_t$

WHY?

And it is thus easy to show that ….

$$\lim_{t\to\infty} S_t = \lim_{t\to\infty} E\left[(J_t - J^*)^2\right] = \frac{\alpha\sigma^2}{(2-\alpha)}$$

- What do you think of TD learning?
- How would you improve it?

Decaying Learning Rate

[Dayan 1991ish] showed that for General TD learning of a Markow System (not just our simple model) that if you use update rule

$$J^{est}(S_i) \leftarrow \alpha_t\left[r_i + \gamma J^{est}(S_j)\right] + (1-\alpha_t)J^{est}(S_i)$$

then, as number of observations goes to infinity $J^{est}(S_i) \to J^*(S_i)\,\forall i$

PROVIDED

- All states visited ∞ly often
- $\sum_{t=1}^{\infty} \alpha_t = \infty$
- $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$

This means $\forall k. \exists T. \sum_{t=1}^{T} \alpha_t > k$

This means $\exists k. \forall T. \sum_{t=1}^{T} \alpha_t^2 < k$

Decaying Learning Rate

This Works: $\alpha_t = 1/t$
This Doesn't: $\alpha_t = \alpha_0$
This Works: $\alpha_t = \beta/(\beta+t)$ [e.g. β=1000]
This Doesn't: $\alpha_t = \beta\alpha_{t-1}$ (β<1)
IN OUR EXAMPLE….USE $\alpha_t = 1/t$

Remember $J^* = E[r_t]$, $\sigma^2 = E[(r_t - J^*)^2]$

$$J_{t+1} = \alpha_t r_t + (1-\alpha_t)J_t = \frac{1}{t}r_t + \left(1-\frac{1}{t}\right)J_t$$

Write $C_t = (t-1)J_t$ and you'll see that

$$C_{t+1} = r_t + C_t \quad \text{so} \quad J_{t+1} = \frac{1}{t}\left[\sum_{i=1}^{t} r_t + J_0\right]$$

And…

Decaying Learning Rate con't…

$$\ldots \quad E\left[(J_t - J^*)^2\right] = \frac{\sigma^2 + (J_0 - J^*)^2}{t}$$

so, ultimately $\lim_{t\to\infty} E\left[(J_t - J^*)^2\right] = 0$

A Fancier TD…

Write S[t] = state at time t

Suppose $\alpha = 1/4$ $\gamma = 1/2$

Assume $J^{est}(S_{23})=0$ $J^{est}(S_{17})=0$ $J^{est}(S_{44})=16$

Assume t = 405 and $S[t] = S_{23}$

Observe $S_{23} \xrightarrow{(r=0)} S_{17}$ with reward 0

Now t = 406, $S[t] = S_{17}$, $S[t-1] = S_{23}$

$J^{est}(S_{23})=$, $J^{est}(S_{17})=$, $J^{est}(S_{44})=$

Observe $S_{17} \xrightarrow{(r=0)} S_{44}$

Now t = 407, S[t] = S44

$J^{est}(S_{23})=$, $J^{est}(S_{17})=$, $J^{est}(S_{44})=$

INSIGHT: $J^{est}(S_{23})$ might think

I gotta get me some of that !!!

TD(λ) Comments

TD(λ=0) is the original TD

TD(λ=1) is almost the same as supervised learning (except it uses a learning rate instead of explicit counts)

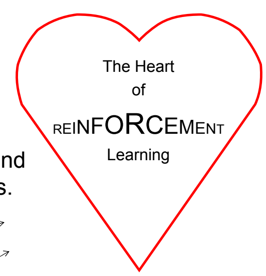
TD(λ=0.7) is often empirically the best performer

- Dayan's proof holds for all $0 \le \lambda \le 1$
- Updates can be made more computationally efficient with "eligibility" traces (similar to O.S.L.)
- Question:
  - Can you invent a problem that would make TD(0) look bad and TD(1) look good?
  - How about TD(0) look good & TD(1) bad??

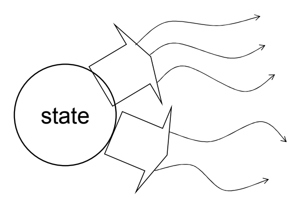
Learning M.S. Summary

		Space	J Update Cost	Data Efficiency
MODEL-BASED	Supervised Learning	$0(N_s)$	$0\left(\frac{1}{\log \frac{1}{\gamma}}\right)$	
MODEL-BASED	Full C.E. Learning	$0(N_{so})$	$0(N_{so}N_s)$ $0(N_{so}k_{CRIT})$	
MODEL-BASED	One Backup C.E. Learning	$0(N_{so})$	$0(1)$	
MODEL-BASED	Prioritized Sweeping	$0(N_{so})$	$0(1)$	
MODEL FREE	TD(0)	$0(N_s)$	$0(1)$	
MODEL FREE	TD(λ) , $0<\lambda\le 1$	$0(N_s)$	$0\left(\frac{1}{\log \frac{1}{\gamma\lambda}}\right)$	

Learning Policies for MDPs



See previous lecture slides for definition of and computation with MDPs.



The task:

World: You are in state 34.
Your immediate reward is 3. You have 3 actions.

Robot: I'll take action 2.

World: You are in state 77.
Your immediate reward is -7. You have 2 actions.

Robot: I'll take action 1.

World: You're in state 34 (again).
Your immediate reward is 3. You have 3 actions.

The Markov property means once you've selected an action the P.D.F. of your next state is the same as the last time you tried the action in this state.

The "Credit Assignment" Problem

I'm in state 43, reward = 0, action = 2
" " " 39, " = 0, " = 4
" " " 22, " = 0, " = 1
" " " 21, " = 0, " = 1
" " " 21, " = 0, " = 1
" " " 13, " = 0, " = 2
" " " 54, " = 0, " = 2
" " " 26, " = 100,

Yippee! I got to a state with a big reward! But which of my actions along the way actually helped me get there??

This is the Credit Assignment problem.

It makes Supervised Learning approaches (e.g. Boxes [Michie & Chambers]) very, very slow.

Using the MDP assumption helps avoid this problem.

MDP Policy Learning

	Space	Update Cost	Data Efficiency
Full C.E. Learning	$0(N_{sAo})$	$0(N_{sAo}k_{CRIT})$	
One Backup C.E. Learning	$0(N_{sAo})$	$0(N_{A0})$	
Prioritized Sweeping	$0(N_{sAo})$	$0(\beta N_{A0})$	

- We'll think about Model-Free in a moment…
- The C.E. methods are very similar to the MS case, except now do value-iteration-for-MDP backups

$$J^{est}(S_i) = \max_a \left[r_i^{est} + \gamma \sum_{S_j \in SUCCS(S_i)} P^{est}(S_j | S_i, a) J^{est}(S_j) \right]$$

Choosing Actions

We're in state S_i

We can estimate r_i^{est}

" " " $P^{est}(\text{next} = S_j \mid \text{this} = S_i, \text{action } a)$

" " " $J^{est}(\text{next} = S_j)$

So what action should we choose ?

$$\text{IDEA 1:} \quad a = \arg\max_{a'} \left[r_i + \gamma \sum_j P^{est}(S_j | S_i, a') J^{est}(S_j) \right]$$

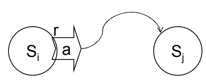
$$\text{IDEA 2:} \quad a = \text{random}$$

- Any problems with these ideas?
- Any other suggestions?
- Could we be optimal?

Model-Free R.L.

Why not use T.D. ?

Observe



update

$$J^{est}(S_i) \leftarrow \alpha\left(r_i + \gamma J^{est}(S_j)\right) + (1-\alpha) J^{est}(S_i)$$

What's wrong with this?

Q-Learning: Model-Free R.L.

[Watkins, 1988]

Define

$Q^*(S_i,a)$= Expected sum of discounted future rewards if I start in state S_i, if I then take action a, and if I'm subsequently optimal

Questions:

Define $Q^*(S_i,a)$ in terms of J^*

Define $J^*(S_i)$ in terms of Q^*

Q-Learning Update

Note that

$$Q^*(S,a) = r_i + \gamma \sum_{S_j \in SUCCS(S_i)} P(S_j | S_i, \alpha) \max_{a'} Q^*(S_j, a')$$

In Q-learning we maintain a table of Q^{est} values instead of J^{est} values…

When you see $S_i \xrightarrow[\text{action a}]{\text{reward}} S_j$ do…

$$Q^{est}(S_i,a) \leftarrow \alpha \left[r_i + \gamma \max_{a'} Q^{est}(S_j, a^1) \right] + (1-\alpha) Q^{est}(S_i, a)$$

This is even cleverer than it looks: the Q^{est} values are not biased by any particular exploration policy. It avoids the Credit Assignment problem.

$$V^*(s) = \max_{a'} Q(s, a')$$

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \gamma \max_{a'} \hat{Q}_{n-1}(s', a')]$$

$$\alpha_n = \frac{1}{1 + visits_n(s,a)}$$

Q-Learning: Choosing Actions

Same issues as for CE choosing actions

- Exploration vs. Exploitation
- Don't always be greedy, so don't always choose: $\arg\max_a Q(s_i, a)$
- Don't always be random (otherwise it will take a long time to reach somewhere exciting)
- Boltzmann exploration [Watkins]

  Prob(choose action a) $\propto \exp\left(-\frac{Q^{est}(s,a)}{K_t}\right)$
- Optimism in the face of uncertainty [Sutton '90, Kaelbling '90]
  - Initialize Q-values optimistically high to encourage exploration
  - Or take into account how often each s,a pair has been tried

Q-Learning Comments

- [Watkins] proved that Q-learning will eventually converge to an optimal policy.
- Empirically it is cute
- Empirically it is very slow
- Why not do Q(λ) ?
  - Would not make much sense [reintroduce the credit assignment problem]
  - Some people (e.g. Peng & Williams) have tried to work their way around this.

If we had time…

- Value function approximation
  - Use a Neural Net to represent J^{est} [e.g. Tesauro]
  - Use a Neural Net to represent Q^{est} [e.g. Crites]
  - Use a decision tree

    …with Q-learning [Chapman + Kaelbling '91]

    …with C.E. learning [Moore '91]

    …How to split up space?
    - Significance test on Q values [Chapman + Kaelbling]
    - Execution accuracy monitoring [Moore '91]
    - Game Theory [Moore + Atkeson '95]
    - New influence/variance criteria [Munos '99]

If we had time…

- R.L. Theory
  - Counterexamples [Boyan + Moore], [Baird]
  - Value Function Approximators with Averaging will converge to something [Gordon]
  - Neural Nets can fail [Baird]
  - Neural Nets with Residual Gradient updates will converge to something
  - Linear approximators for TD learning will converge to something useful [Tsitsiklis + Van Roy]

TD-Gammon

Current Issues

- Making function approximation work
- Abstraction (options, macros, …), learning structure
- Partially Observable Markov Processes, POMDPs

What You Should Know

- Supervised learning for predicting delayed rewards
- Certainty equivalent learning for predicting delayed rewards
- Model free learning (TD) for predicting delayed rewards
- Reinforcement Learning with MDPs: What's the task?
- Why is it hard to choose actions?
- Q-learning (including being able to work through small simulated examples of RL)