Reinforcement Learning: Dynamic Programming

Csaba Szepesvári
University of Alberta

Kioloa, MLSS'08

Slides: http://www.cs.ualberta.ca/~szepesva/MLSS08/

Reinforcement Learning

RL =

"Sampling based methods to solve optimal control problems"

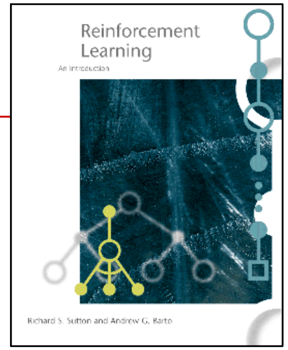

"Ideas matter"

(Rich Sutton)

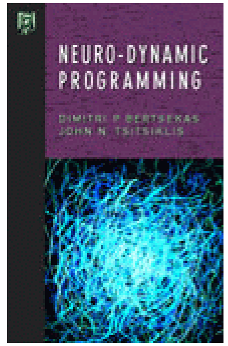
- Contents
  - Defining AI
  - Markovian Decision Problems
  - Dynamic Programming
  - Approximate Dynamic Programming
  - Generalizations

Literature

- Books
  - Richard S. Sutton, Andrew G. Barto: *Reinforcement Learning: An Introduction*, MIT Press, 1998
  - Dimitri P. Bertsekas, John Tsitsiklis: *Neuro-Dynamic Programming*, Athena Scientific, 1996
- Journals
  - JMLR, MLJ, JAIR, AI
- Conferences
  - NIPS, ICML, UAI, AAAI, COLT, ECML, IJCAI




"Ideas matter"



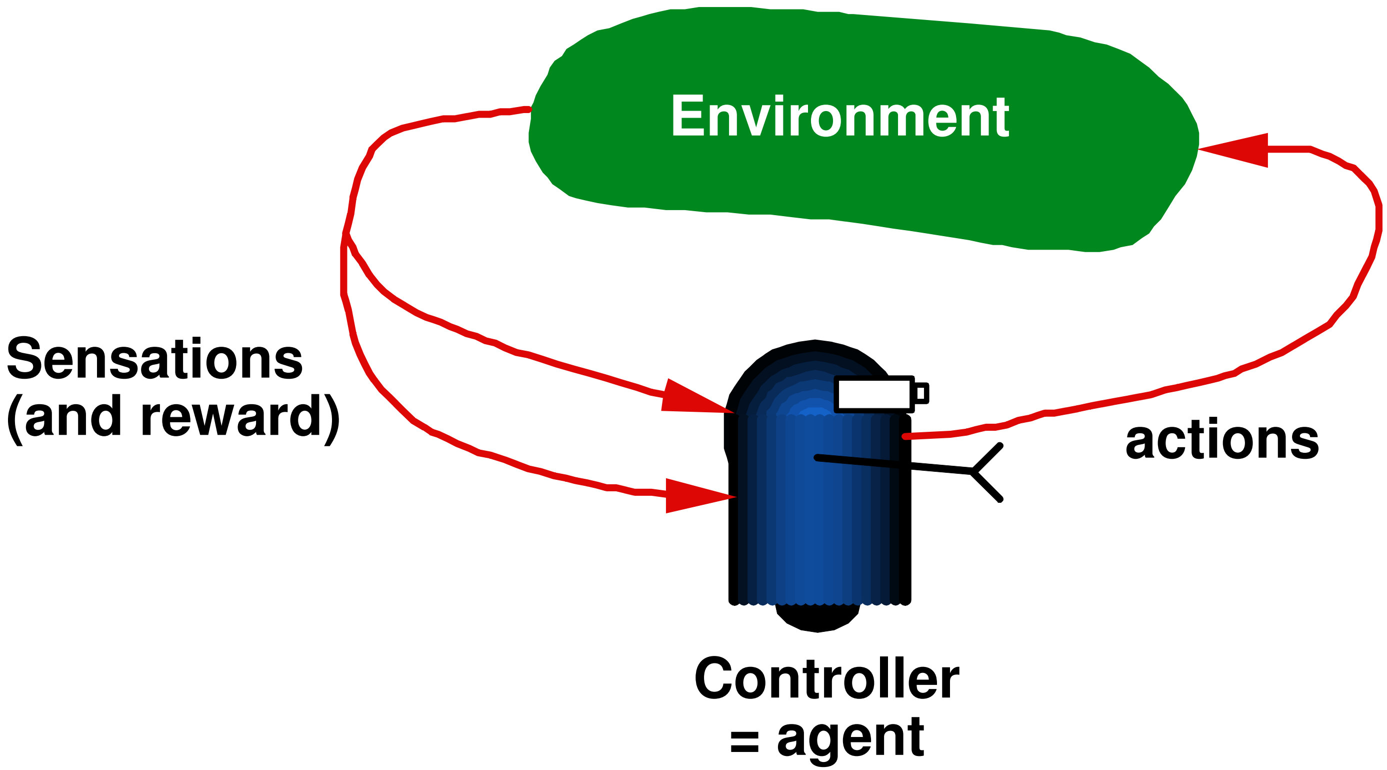
Some More Books

- Martin L. Puterman. *Markov Decision Processes*. Wiley, 1994.
- Dimitri P. Bertsekas: *Dynamic Programming and Optimal Control. Athena Scientific*. Vol. I (2005), Vol. II (2007).
- James S. Spall: *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley, 2003.

Resources

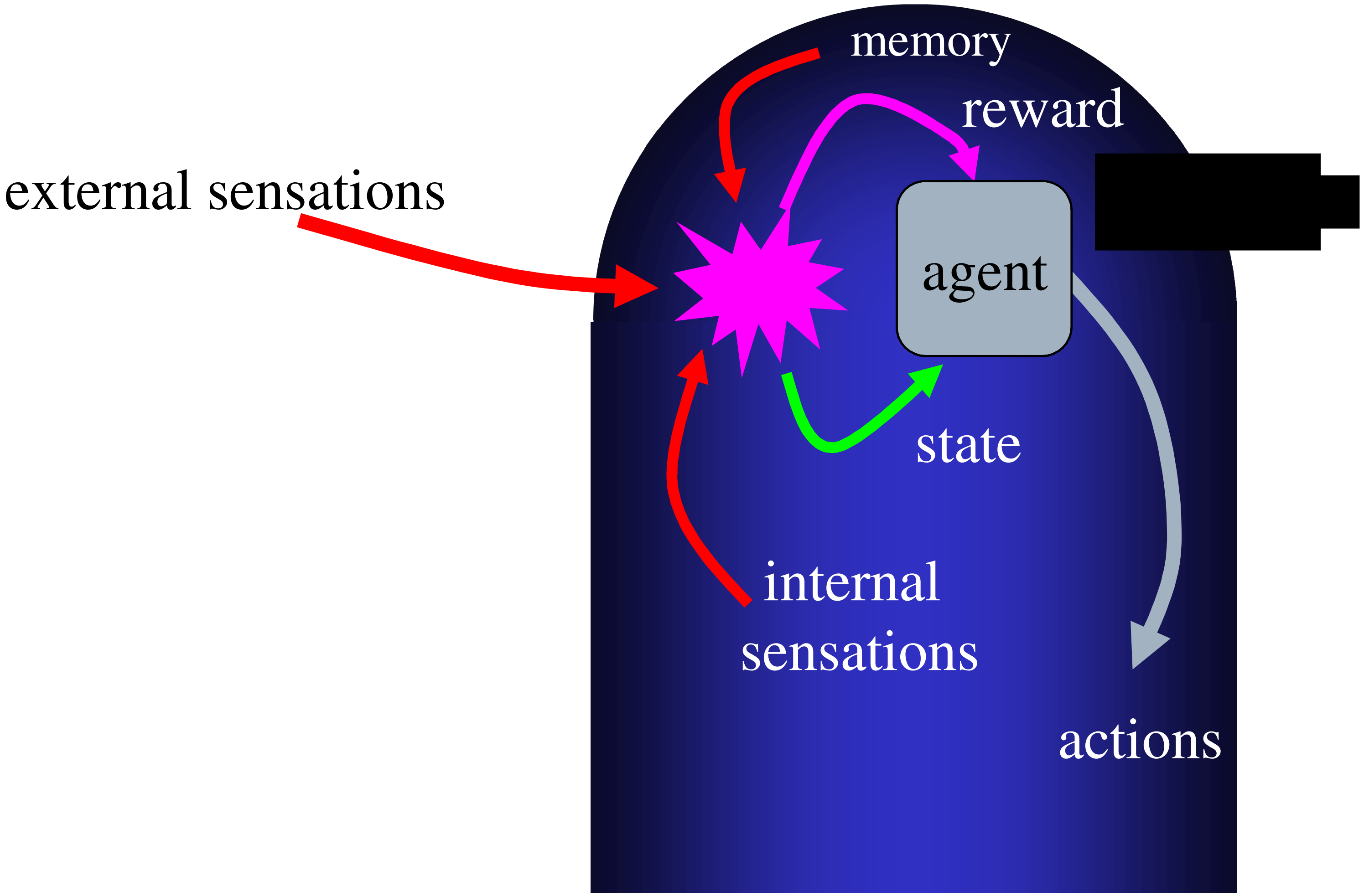
- RL-Glue
  - http://rlai.cs.ualberta.ca/RLBB/top.html
- RL-Library
  - http://rlai.cs.ualberta.ca/RLR/index.html
- The RL Toolbox 2.0
  - http://www.igi.tugraz.at/ril-toolbox/general/overview.html
- OpenDP
  - http://opendp.sourceforge.net
- RL-Competition (2008)!
  - http://rl-competition.org/
  - **June 1st, 2008:** Test runs begin!
- Related fields:
  - Operations research (MOR, OR)
  - Control theory (IEEE TAC, Automatica, IEEE CDC, ECC)
  - Simulation optimization (Winter Simulation Conference)

Abstract Control Model



"Perception-action loop"

Zooming in..


external sensations
memory
reward
agent
state
internal
sensations
actions

A Mathematical Model

- Plant (controlled object):
  - $x_{t+1} = f(x_t, a_t, v_t)$ — x_t : state, v_t : noise
  - $z_t \;\; = g(x_t, w_t)$ — z_t : sens/obs, w_t : noise
- State: Sufficient statistics for the future
  - Independently of what we measure

    ..or..
  - Relative to measurements
- Controller
  - $a_t = F(z_1, z_2, \ldots, z_t)$ — a_t: action/control

  => PERCEPTION-ACTION LOOP

  "CLOSED-LOOP CONTROL"
- Design problem: $F = ?$
- *Goal:* $\sum_{\tau=1}^{T} r(z_t, a_t) \rightarrow max$

"Subjective State"

A Classification of Controllers

- Feedforward:
  - $a_1, a_2, \ldots$ is designed ahead in time
  - ???
- Feedback:
  - Purely reactive systems: $a_t = F(z_t)$
  - Why is this bad?
  - Feedback with memory:

    $m_t = M(m_{t-1}, z_t, a_{t-1})$

    ~interpreting sensations

    $a_t = F(m_t)$

    decision making: deliberative vs. reactive

Feedback controllers

- Plant:
  - $x_{t+1} = f(x_t, a_t, v_t)$
  - $z_{t+1} = g(x_t, w_t)$
- Controller:
  - $m_t = M(m_{t-1}, z_t, a_{t-1})$
  - $a_t = F(m_t)$
- $m_t \approx x_t$: state estimation, “filtering”

  difficulties: noise,unmodelled parts
- How do we compute a_t?
  - With a model (*f'*): model-based control
    - ..assumes (some kind of) state estimation
  - Without a model: model-free control

Markovian Decision Problems

Markovian Decision Problems

- (X,A,p,r)
- X – set of states
- A – set of actions (controls)
- p – transition probabilities

  p(y|x,a)
- r – rewards

  r(x,a,y), or r(x,a), or r(x)
- γ – discount factor

  $0 \leq \gamma < 1$

The Process View

- (X_t, A_t, R_t)
- X_t – state at time t
- A_t – action at time t
- R_t – reward at time t
- Laws:
  - $X_{t+1} \sim p(.|X_t, A_t)$
  - $A_t \sim \pi(.|H_t)$
  - π: policy
  - $H_t = (X_t, A_{t-1}, R_{t-1}, .., A_1, R_1, X_0)$ – history
  - $R_t = r(X_t, A_t, X_{t+1})$

The Control Problem

- Value functions:

$$V_\pi(x) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R_t | X_0 = x]$$

- Optimal value function:

$$V^*(x) = \max_\pi V_\pi(x)$$

- Optimal policy:

$$V_{\pi^*}(x) = V^*(x)$$

Applications of MDPs

- Operations research
- Econometrics

- Optimal investments
- Replacement problems
- Option pricing
- Logistics, inventory management
- Active vision
- Production scheduling
- Dialogue control

- Control, statistics
- Games, AI

- Bioreactor control
- Robotics (Robocup Soccer)
- Driving
- Real-time load balancing
- Design of experiments (Medical tests)

Variants of MDPs

- Discounted
- Undiscounted: Stochastic Shortest Path
- Average reward
- Multiple criteria
- Minimax
- Games

MDP Problems

- **Planning**
  The MDP (X,A,P,r,γ) is known.
  Find an optimal policy π^*!
- **Learning**
  The MDP is unknown.
  You are allowed to interact with it.
  Find an optimal policy π^*!
- **Optimal learning**
  While interacting with the MDP,
  minimize the loss due to not using an
  optimal policy from the beginning

Solving MDPs – Dimensions

- Which problem? (Planning, learning, optimal learning)
- Exact or approximate?
- Uses samples?
- Incremental?
- Uses value functions?
  - Yes: Value-function based methods
    - Planning: DP, Random Discretization Method, FVI, …
    - Learning: Q-learning, Actor-critic, …
  - No: Policy search methods
    - Planning: Monte-Carlo tree search, Likelihood ratio methods (policy gradient), Sample-path optimization (Pegasus),
- Representation
  - Structured state:
    - Factored states, logical representation, …
  - Structured policy space:
    - Hierarchical methods

Dynamic Programming

Richard Bellman (1920-1984)

- Control theory
- Systems Analysis
- Dynamic Programming:
  RAND Corporation, 1949-1955

- Bellman equation
- Bellman-Ford algorithm
- Hamilton-Jacobi-Bellman equation
- “Curse of dimensionality”
- invariant imbeddings
- Grönwall-Bellman inequality

Bellman Operators

- Let π:X $\rightarrow$ A be a stationary policy
- B(X) = { V | V:X$\rightarrow$ R, $||V||_\infty < \infty$ }
- T_π:B(X)$\rightarrow$ B(X)
- $(T_\pi V)(x) = \sum_y p(y|x,\pi(x)) [r(x,\pi(x),y)+\gamma V(y)]$
- **Theorem**:

$$T_\pi V_\pi = V_\pi$$

- **Note:** This is a linear system of equations: $r_\pi + \gamma P_\pi V_\pi = V_\pi$

$$\Rightarrow V_\pi = (I-\gamma P_\pi)^{-1} r_\pi$$

Proof of $T_\pi V_\pi = V_\pi$

- What you need to know:
  - Linearity of expectation: E[A+B] = E[A]+E[B]
  - Law of total expectation:
    $E[\,Z\,] = \sum_x P(X=x)\, E[\,Z \mid X=x\,]$, and
    $E[\,Z \mid U=u\,] = \sum_x P(X=x \mid U=u)\, E[Z \mid U=u, X=x]$.
  - Markov property:
    $E[\,f(X_1,X_2,..) \mid X_1=y, X_0=x] = E[\,f(X_1,X_2,..) \mid X_1=y]$
- $V_\pi(x) = E_\pi\,[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x]$
  $= \sum_y P(X_1=y \mid X_0=x)\, E_\pi[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x, X_1=y]$
  (by the law of total expectation)
  $= \sum_y p(y \mid x,\pi(x))\, E_\pi[\sum_{t=0}^{\infty} \gamma^t R_t \mid X_0 = x, X_1=y]$
  (since $X_1 \sim p(.\mid X_0,\pi(X_0))$)
  $= \sum_y p(y \mid x,\pi(x))$
  $\{E_\pi[\,R_0 \mid X_0=x, X_1=y] + \gamma\, E_\pi\,[\,\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid X_0=x, X_1=y]\}$
  (by the linearity of expectation)
  $= \sum_y p(y \mid x,\pi(x))\, \{r(x,\pi(x),y) + \gamma\, V_\pi(y)\}$
  (using the definition of r, V_π)
  $= (T_\pi V_\pi)(x)$. (using the definition of T_π)

The Banach Fixed-Point Theorem

- $B = (B, ||.||)$ Banach space
- $T: B_1 \to B_2$ is L-Lipschitz ($L>0$) if for any U,V,

  $$|| T U - T V || \leq L ||U-V||.$$
- T is contraction if $B_1=B_2$, $L<1$; L is a contraction coefficient of T
- **Theorem [Banach]**: Let $T:B\to B$ be a γ-contraction. Then T has a unique fixed point V and $\forall V_0 \in B$, $V_{k+1}=T V_k$, $V_k \to V$ and $||V_k-V||=O(\gamma^k)$

An Algebra for Contractions

- **Prop**: If T_1:$B_1 \to B_2$ is L_1-Lipschitz, T_2: $B_2 \to B_3$ is L_2-Lipschitz then T_2 T_1 is L_1 L_2 Lipschitz.
- **Def**: If T is 1-Lipschitz, T is called a non-expansion
- **Prop**: M: $B(X \times A) \to B(X)$, $M(Q)(x) = \max_a Q(x,a)$ is a non-expansion
- **Prop**: Mul_c: $B \to B$, Mul_c V = c V is |c|-Lipschitz
- **Prop**: Add_r: $B \to B$, Add V = r + V is a non-expansion.
- **Prop**: K: $B(X) \to B(X)$, $(K\,V)(x)=\sum_y K(x,y)\,V(y)$ is a non-expansion if $K(x,y) \geq 0$, $\sum_y K(x,y) =1$.

Policy Evaluations are Contractions

- **Def:** $||V||_\infty = \max_x |V(x)|$, supremum norm; here $||.||$
- **Theorem**: Let T_π the policy evaluation operator of some policy π. Then T_π is a γ-contraction.
- **Corollary**: V_π is the unique fixed point of T_π. $V_{k+1} = T_\pi V_k \to V_\pi$, $\forall V_0 \in B(X)$ and $||V_k - V_\pi|| = O(\gamma^k)$.

The Bellman Optimality Operator

- Let $T: B(X) \to B(X)$ be defined by $(TV)(x) = \max_a \sum_y p(y|x,a) \{ r(x,a,y) + \gamma V(y) \}$
- **Def**: π is greedy w.r.t. V if $T_\pi V = T V$.
- **Prop**: T is a γ-contraction.
- **Theorem (BOE)**: $T V^* = V^*$.
- **Proof**: Let V be the fixed point of T. $T_\pi \leq T \Rightarrow V^* \leq V$. Let π be greedy w.r.t. V. Then $T_\pi V = T V$. Hence $V_\pi = V \Rightarrow V \leq V^* \Rightarrow V = V^*$.

Value Iteration

- **Theorem**: For any $V_0 \in B(X)$, $V_{k+1} = T\,V_k$, $V_k \to V^*$ and in particular $||V_k - V^*||=O(\gamma^k)$.
- What happens when we stop "early"?
- **Theorem**: Let π be greedy w.r.t. V. Then $||V_\pi - V^*|| \leq 2||TV-V||/(1-\gamma)$.
- **Proof**: $||V_\pi-V^*|| \leq ||V_\pi-V||+||V-V^*||$ …
- **Corollary:** In a finite MDP, the number of policies is finite. We can stop when $||V_k-TV_k|| \leq \Delta(1-\gamma)/2$, where $\Delta = \min\{ ||V^*-V_\pi|| : V_\pi \neq V^* \}$

  ➔Pseudo-polynomial complexity

Policy Improvement [Howard '60]

- **Def**: $U,V \in B(X)$, $V \geq U$ if $V(x) \geq U(x)$ holds for all $x \in X$.
- **Def**: $U,V \in B(X)$, $V > U$ if $V \geq U$ and $\exists\, x \in X$ s.t. $V(x) > U(x)$.
- **Theorem (Policy Improvement)**: Let π' be greedy w.r.t. V_π. Then $V_{\pi'} \geq V_\pi$. If $T V_\pi > V_\pi$ then $V_{\pi'} > V_\pi$.

Policy Iteration

- Policy Iteration(π)
- $V \leftarrow V_\pi$
- Do {improvement}
  - $V' \leftarrow V$
  - Let π: $T_\pi V = T V$
  - $V \leftarrow V_\pi$
- While ($V>V'$)
- Return π

Policy Iteration Theorem

- **Theorem**: In a finite, discounted MDP policy iteration stops after a finite number of steps and returns an optimal policy.
- **Proof**: Follows from the Policy Improvement Theorem.

Linear Programming

- $V \geq T\,V \Rightarrow V \geq V^* = T\,V^*$.
- Hence, V^* is the “largest” V that satisfies $V \geq T\,V$.

$V \geq T\,V \quad \Leftrightarrow$

(*) $V(x) \geq \sum_y p(y|x,a)\{r(x,a,y)+\gamma\, V(y)\},$
$\forall x,a$

- **LinProg(**V):
  - $\sum_x V(x) \rightarrow \min$ s.t. V satisfies (*).
- **Theorem**: LinProg(V) returns the optimal value function, V^*.
- **Corollary**: Pseudo-polynomial complexity

Variations of a Theme

Approximate Value Iteration

- **AVI**: $V_{k+1} = T V_k + \epsilon_k$
- **AVI Theorem**:
  Let $\epsilon = \max_k ||\epsilon_k||$. Then
  $\limsup_{k\to\infty} ||V_k - V^*|| \le 2\gamma \epsilon / (1-\gamma)$.
- **Proof**: Let $a_k = ||V_k - V^*||$.
  Then $a_{k+1} = ||V_{k+1} - V^*|| = ||T V_k - T V^* + \epsilon_k|| \le \gamma ||V_k - V^*|| + \epsilon = \gamma a_k + \epsilon$.
  Hence, a_k is bounded. Take "limsup" of both sides: $a \le \gamma a + \epsilon$; reorder.//

(e.g., [BT96])

Fitted Value Iteration – Non-expansion Operators

- **FVI**: Let A be a non-expansion, $V_{k+1} = A\ T\ V_k$. Where does this converge to?
- **Theorem**: Let U,V be such that A T U = U and T V = V. Then $||V-U|| \leq ||AV - V||/(1-\gamma)$.
- **Proof**: Let U' be the fixed point of TA. Then $||U'-V|| \leq \gamma\ ||AV-V||/(1-\gamma)$. Since A U' = A T (AU'), U=AU'. Hence, $||U-V|| = ||AU'-V||$ $\leq ||AU'-AV|| + ||AV-V||$ ...

[Gordon '95]

Application to Aggregation

- Let Π be a partition of X, S(x) be the unique cell that x belongs to.
- Let A: B(X)$\rightarrow$ B(X) be
  (A V)(x) = $\sum_z \mu$(z;S(x)) V(z), where μ is a distribution over S(x).
- p'(C|B,a) =
  $\sum_{x\in \mathcal{B}} \mu$(x;B) $\sum_{y\in \mathcal{C}}$ p(y|x,a),
  r'(B,a,C) =
  $\sum_{x\in \mathcal{B}} \mu$(x;B) $\sum_{y\in \mathcal{C}}$ p(y|x,a) r(x,a,y).
- **Theorem**: Take (Π,A,p',r'), let V' be its optimal value function, $V'_E(x) = V'(S(x))$.
  Then $||V'_E - V^*|| \leq ||AV^*-V^*||/(1-\gamma)$.

Action-Value Functions

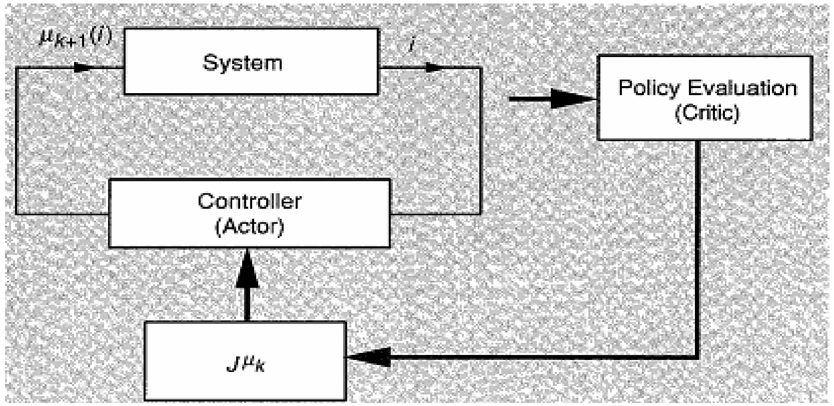
- L: $B(X) \to B(X \times A)$,
  $(L\,V)(x,a) = \sum_y p(y|x,a) \{r(x,a,y) + \gamma V(y)\}$.
  "One-step lookahead".
- **Note**: π is greedy w.r.t. V if
  $(LV)(x,\pi(x)) = \max_a (LV)(x,a)$.
- **Def**: $Q^* = L\,V^*$.
- **Def**: Let Max: $B(X \times A) \to B(X)$,
  $(\text{Max}\,Q)(x) = \max_a Q(x,a)$.
- Note: Max L = T.
- **Corollary**: $Q^* = L\,\text{Max}\,Q^*$.
  - Proof: $Q^* = L\,V^* = L\,T\,V^* = L\,\text{Max}\,L\,V^* = L\,\text{Max}\,Q^*$.
- T = L Max is a γ-contraction
- Value iteration, policy iteration, …

Changing Granularity

- Asynchronous Value Iteration:
  - Every time-step update only a few states
- **AsyncVI Theorem**: If all states are updated infinitely often, the algorithm converges to V^*.
- How to use?
  - Prioritized Sweeping
- IPS [MacMahan & Gordon ’05]:
  - Instead of an update, put state on the priority queue
  - When picking a state from the queue, update it
  - Put predecessors on the queue
- <u>Theorem</u>: Equivalent to Dijkstra on shortest path problems, provided that rewards are non-positive
- LRTA* [Korf ’90] ~ RTDP [Barto, Bradtke, Singh ’95]
  - Focussing on parts of the state that matter
  - Constraints:
    - Same problem solved from several initial positions
    - Decisions have to be fast
  - Idea: Update values along the paths

Changing Granularity

- Generalized Policy Iteration:
  - Partial evaluation *and* partial improvement of policies
  - Multi-step lookahead improvement



- **AsyncPI Theorem**: If both evaluation and improvement happens at every state infinitely often then the process converges to an optimal policy. [Williams & Baird '93]

Variations of a theme [SzeLi99]

- Game against nature [Heger '94]:
  $\inf_w \sum_t \gamma^t R_t(w)$ with $X_0 = x$
- Risk-sensitive criterion:
  $\log (E[\exp(\sum_t \gamma^t R_t) \mid X\_0 = x])$
- Stochastic Shortest Path
- Average Reward
- Markov games
  - Simultaneous action choices (Rock-paper-scissor)
  - Sequential action choices
  - Zero-sum (or not)

References

- [Howard ’60] R.A. Howard: *Dynamic Programming and Markov Processes*, The MIT Press, Cambridge, MA, 1960.
- [Gordon ’95] G.J. Gordon: Stable function approximation in dynamic programming. ICML, pp. 261—268, 1995.
- [Watkins ’90] C.J.C.H. Watkins: *Learning from Delayed Rewards*, PhD Thesis, 1990.
- [McMahan, Gordon ’05] H. B. McMahan and Geoffrey J. Gordon: Fast Exact Planning in Markov Decision Processes. ICAPS.
- [Korf ’90] R. Korf: Real-Time Heuristic Search. *Artificial Intelligence* 42, 189–211, 1990.
- [Barto, Bradtke & Singh, ’95] A.G. Barto, S.J. Bradtke & S. Singh: Learning to act using real-time dynamic programming, *Artificial Intelligence* 72, 81—138, 1995.
- [Williams & Baird, ’93] R.J. Williams & L.C. Baird: *Tight Performance Bounds on Greedy Policies Based on Imperfect Value Functions*. Northeastern University Technical Report NU-CCS-93-14, November, 1993.
- [SzeLi99] Cs. Szepesvári and M.L. Littman: A Unified Analysis of Value-Function-Based Reinforcement-Learning Algorithms, *Neural Computation*, 11, 2017—2059, 1999.
- [Heger ’94] M. Heger: Consideration of risk in reinforcement learning, ICML, 105—111, 1994.

Reinforcement Learning: Approximate Planning

Csaba Szepesvári
University of Alberta

Kioloa, MLSS’08

Slides: http://www.cs.ualberta.ca/~szepesva/MLSS08/

Planning Problem

- The MDP
  - .. is given (p,r can be queried)
  - .. can be sampled from
    - at any state
    - Trajectories

    "Simulation Optimization"
- Goal: Find an optimal policy
- Constraints:
  - Computational efficiency
    - Polynomial complexity
    - O(1) $\equiv$ real-time decisions
  - Sample efficiency ~ computational efficiency

Methods for planning

- Exact solutions (DP)
- Approximate solutions
  - Rollouts ($\equiv$ search)
    - Sparse lookahead trees, UCT
  - Approximate value functions
    - RDM, FVI, LP
  - Policy search
    - Policy gradient (Likelihood Ratio Method), Pegasus [Ng & Jordan '00]
  - Hybrid
    - Actor-critic

Bellman's Curse of Dimensionality

- The state space in many problems is..
  - Continuous
  - High-dimensional
- "Curse of Dimensionality" (Bellman, 57)
  - Running time of algorithms scales *exponentially* with the dimension of the state space.
- Transition probabilities
  - Kernel: $P(dy|x,a)$
  - Density: $p(y|x,a)$ ← !!
    - e.g. $p(y|x,a) \sim \exp(-||y-f(x,a)||^2/(2\sigma^2))$

A Lower Bound

- **Theorem** (Chow, Tsitsiklis ’89)
  - Markovian Decision Problems
  - *d* dimensional state space
  - Bounded transition probabilities, rewards
  - Lipschitz-continuous transition probabilities and rewards

  ➔ *Any* algorithm computing an ϵ-approximation of the optimal value function needs $\Omega(\epsilon^{-d})$ values of *p* and *r*.
- What’s next then??

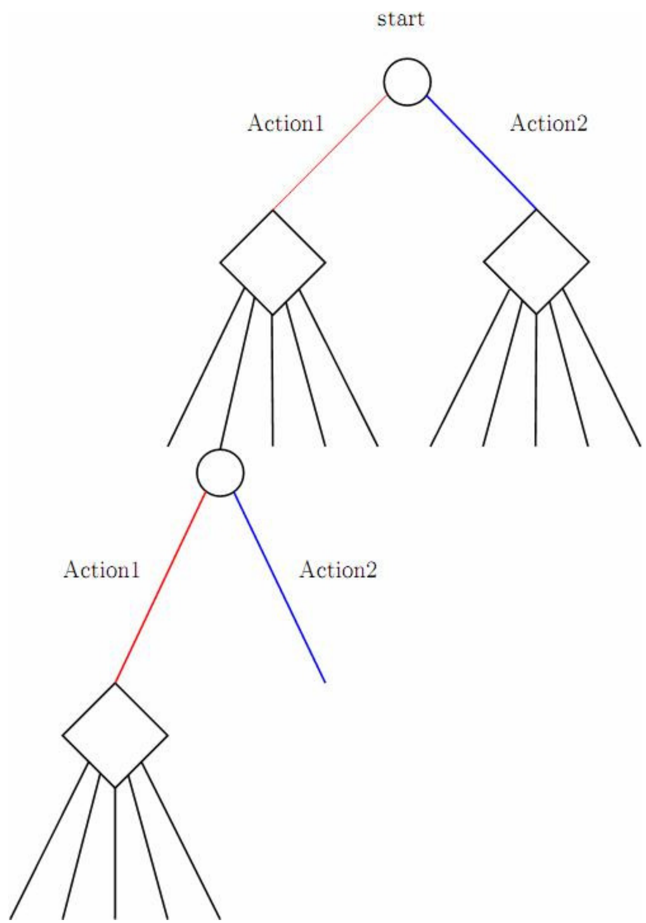
Monte-Carlo Search Methods

Problem:

- Can generate trajectories from an initial state
- Find a good action at the initial state

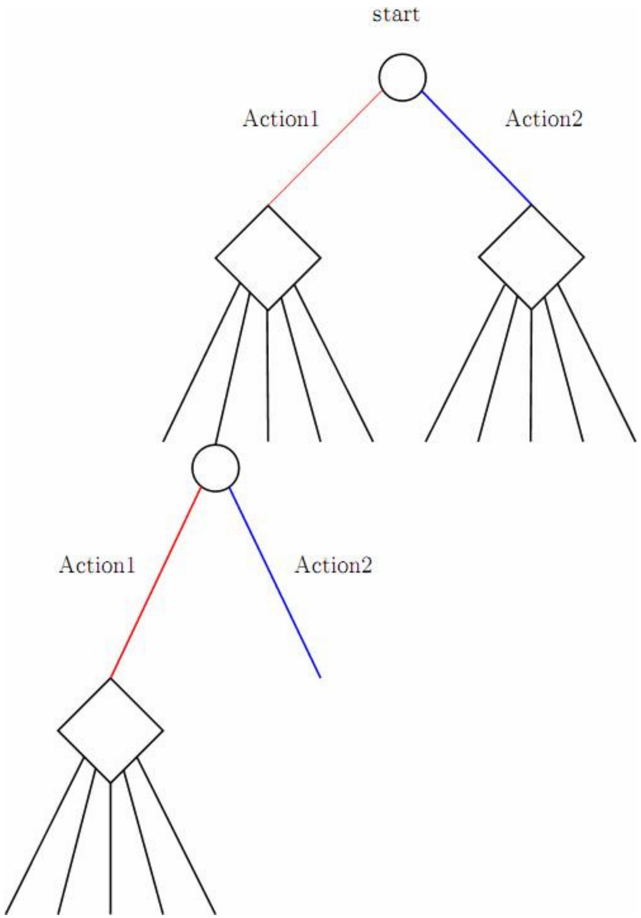
Sparse lookahead trees

- [Kearns et al., ’02]: Sparse lookahead trees
- Effective horizon:
  $H(\epsilon) = K_r/(\epsilon(1-\gamma))$
- Size of the tree:
  $S = c\,|A|^{H(\epsilon)}$ (unavoidable)
- Good news: S is independent of d!
- ..but is exponential in $H(\epsilon)$
- Still attractive: Generic, easy to implement
- Would you use it?



Idea..

- Need to propagate values from good branches as early as possible
- Why sample suboptimal actions at all?
- Breadth-first
  ➔ Depth-first!
- Bandit algorithms ➔ Upper Confidence Bounds



➔ UCT

[KoSze '06]

UCB [Auer et al. '02]

- Bandit with a finite number of actions (a) – called arms here
- $Q_t(a)$: Estimated payoff of action a
- $T_t(a)$: Number of pulls of arm a
- Action choice by UCB:

$$A_t = \operatorname{argmax}_a \left\{ Q_t(a) + \sqrt{\frac{p \log(t)}{2T_t(a)}} \right\}$$

- **Theorem**: The expected loss is bounded by O(log n)
- Optimal rate

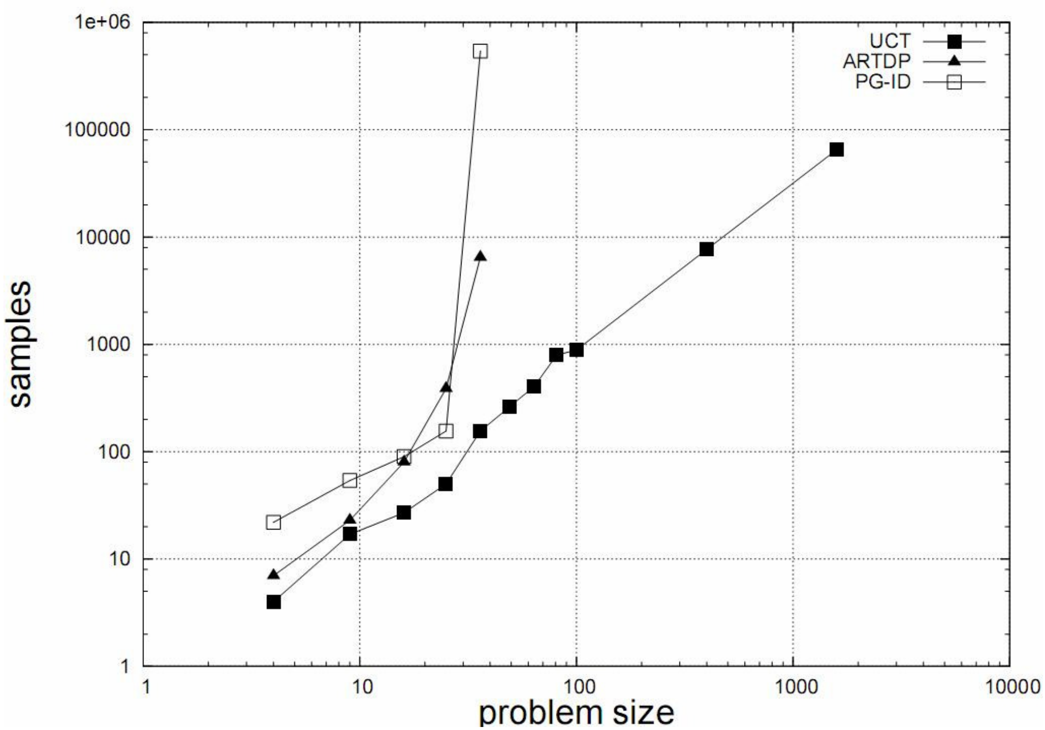
UCT Algorithm [KoSze ’06]

- To decide which way to go play a bandit in each node of the tree
- Extend tree one by one

- Similar ideas:
  - [Peret and Garcia, ’04]
  - [Chang et al., ’05]

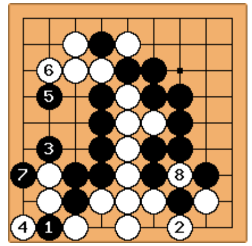
Results: Sailing

- 'Sailing': Stochastic shortest path



- State-space size = 24*problem-size
- Extension to two-player, full information games
- Major advances in go!

Results: 9x9 Go



- Mogo
  - A: Y. Wang, S. Gelly, R. Munos, O. Teytaud, and P-A. Coquelin, D. Silver
  - 100-230K simulations/move
  - Around since 2006 aug.
- CrazyStone
  - A: Rémi Coulom
  - Switched to UCT in 2006
- Steenvreter
  - A: Erik van der Werf
  - Introduced in 2007
- Computer Olympiad (2007 December)
  - 19x19
    1. MoGo
    2. CrazyStone
    3. GnuGo
  - 9x9
    1. Steenvreter
    2. Mogo
    3. CrazyStone
- Guo Jan (5 dan), 9x9 board
  - Mogo black: 75% win
  - Mogo white: 33% win

CGOS: 1800 ELO → 2600 ELO

Random Discretization Method

Problem:

- Continuous state-space
- Given p,r, find a good policy!
- Be efficient!

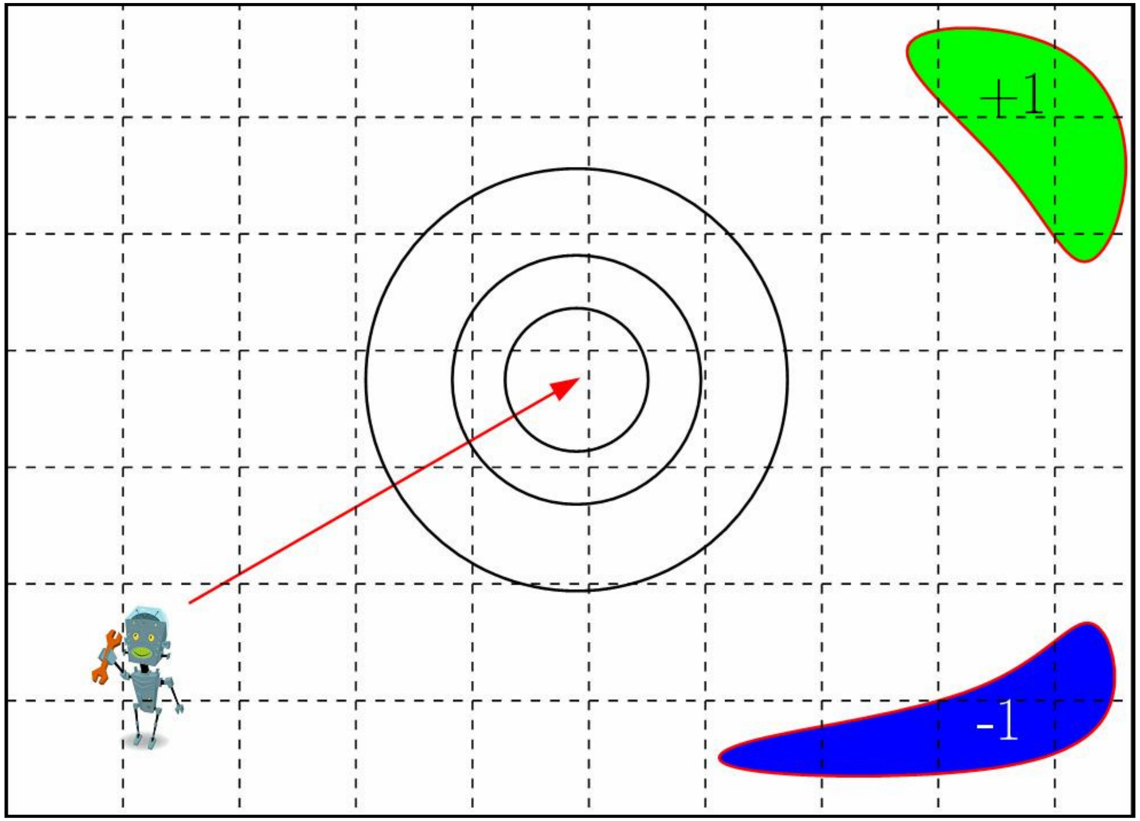
Value Iteration in Continuous Spaces

- Value Iteration:

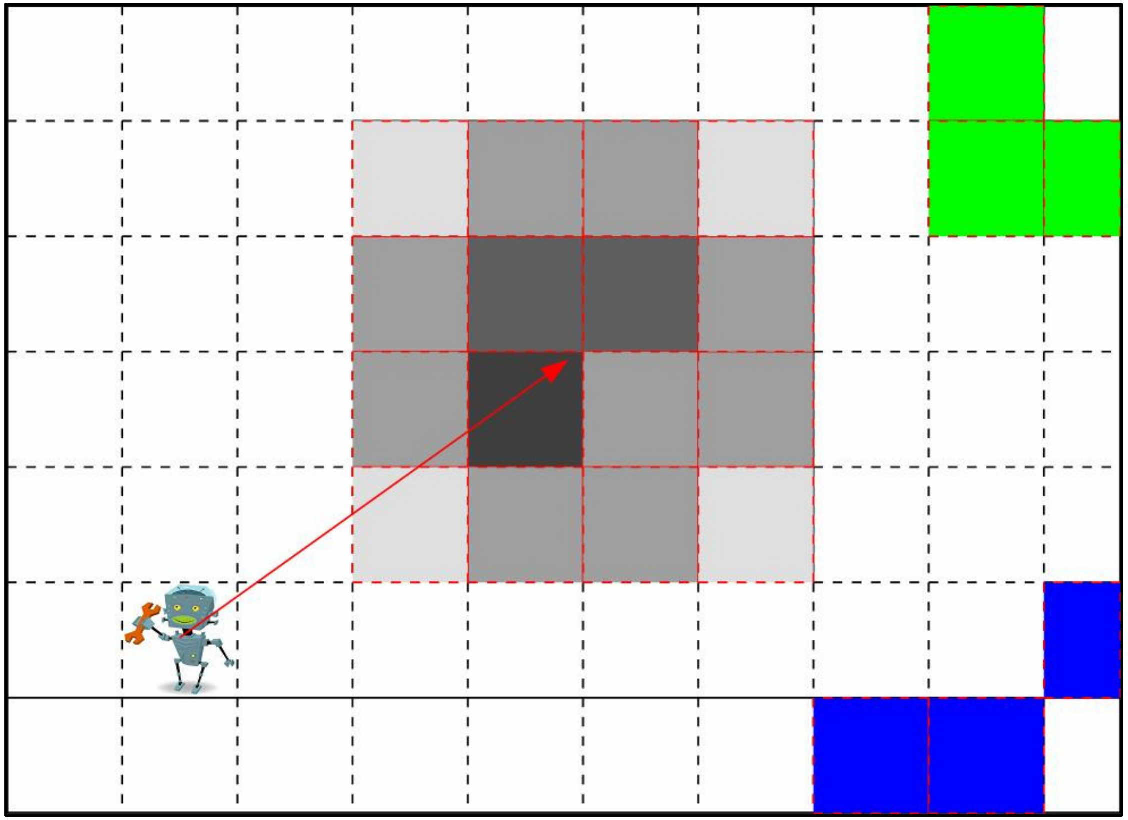
$$V_{k+1}(x) = \max_{a \in \mathcal{A}} \{r(x,a) + \gamma \int_{\mathcal{X}} p(y|x,a)\, V_k(y)\, dy\}$$

- How to compute the integral?
- How to represent value functions?

Discretization



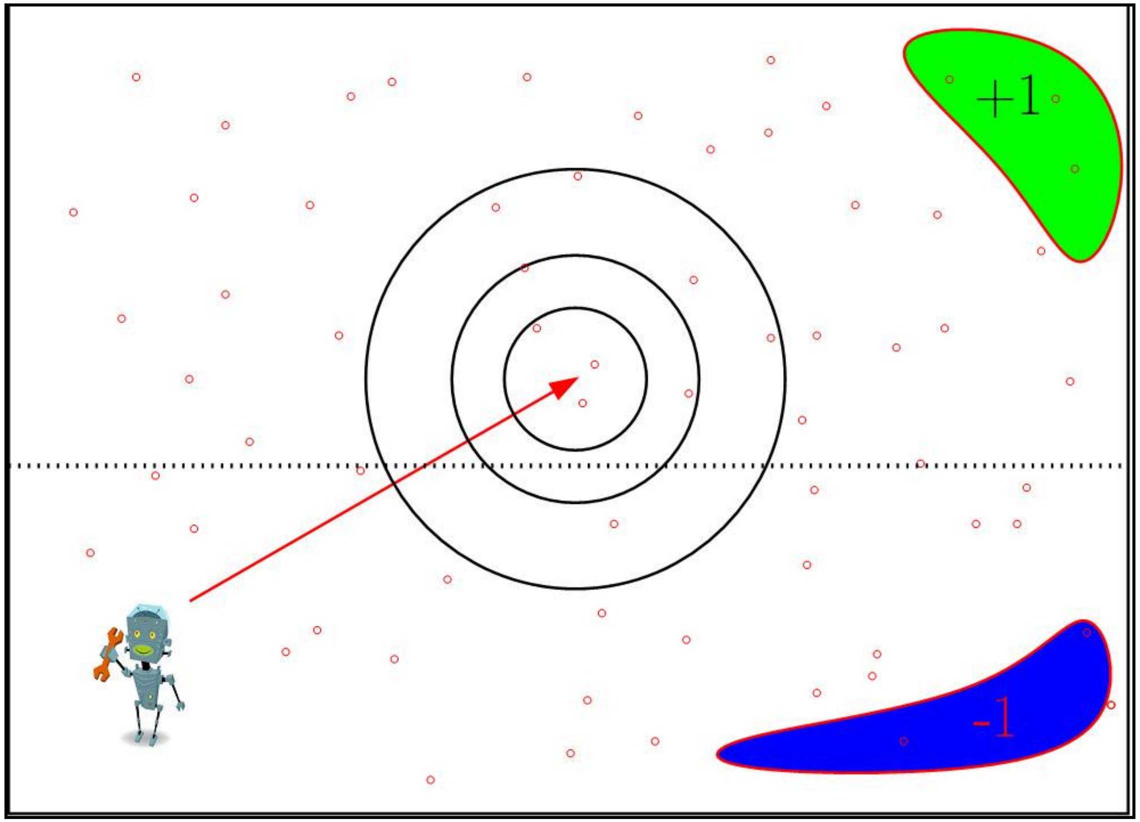
Discretization



Can this work?

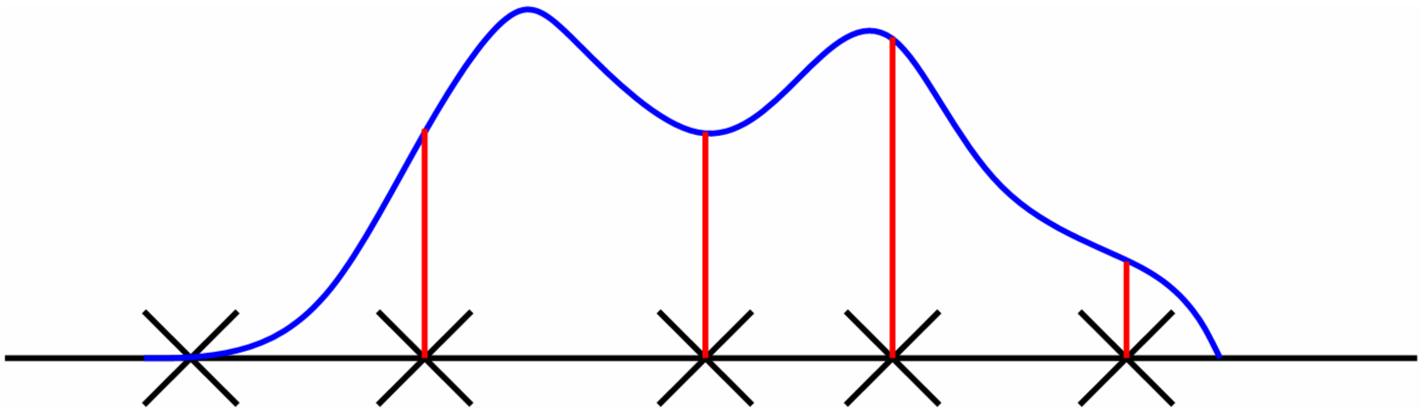
- No!
- The result of [Chow and Tsitsiklis, 1989] says that methods like this can not scale well with the dimensionality

Random Discretization [Rust '97]



Weighted Importance Sampling

- How to compute ∫ p(y|x,a) V(y) dy?



$$Y_i \sim U_X(\cdot) \Rightarrow$$

$$\frac{\sum_{i=1}^{N} p(Y_i|x,a)V(Y_i)}{\sum_{i=1}^{N} p(Y_i|x,a)} \rightarrow \int p(y|x,a)V(y)dy \ \text{ w.p.1}$$

The Strength of Monte-Carlo

- Goal: Compute $I(f) = \int f(x)\, p(x)\, dx$
- Draw $X_1, \ldots, X_N \sim p(.)$
- Compute $I_N(f) = 1/N \sum_i f(X_i)$
- **Theorem:**
  - $E[\ I_N(f)\ ] = I(f)$
  - $Var[\ I_N(f)\ ] = Var[f(X_1)]/N$
  - Rate of convergence is independent of the dimensionality of x!

The Random Discretization Method

function RDM-prepare(p, r, γ)
draw $X_1, \ldots, X_N$ randomly, uniformly over $\mathcal{X}$
$\hat{p}(i|j, a) \leftarrow \frac{p(X_i|X_j,a)}{\sum_{k=1}^{N} p(X_k|X_j,a)}$, $\hat{r}(i, a) \leftarrow r(X_i, a)$
$v \leftarrow VI(K; \hat{p}, \hat{r}, \gamma)$ // call value iteration
return v

function RDM-estimate(x, v, p, r, γ) // $x \in \mathcal{X}$
return $\max_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{j=1}^{n} \frac{p(X_j|x,a)}{\sum_{k=1}^{N} p(X_k|x,a)} v(j) \right\}$

Guarantees

- State space: $[0,1]^d$
- Action space: finite
- p(y|x,a), r(x,a) Lipschitz continuous, bounded
- **Theorem** [Rust '97]:

$$E\left[\|V_N(x) - V^*(x)\|_\infty\right] \leq \frac{Cd|A|^{5/4}}{(1-\gamma)^2 N^{1/4}}$$

- No curse of dimensionality!
- Why??
- Can we have a result for planning??

Planning

[Sze '01]

- Replace $\max_a$ with argmax_a in procedure RDM-estimate:

> function **plan0**(x, v, p, r, γ)
> **return** $\operatorname{argmax}_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{j=1}^{n} \frac{p(X_j | x, a)}{\sum_{k=1}^{N} p(X_k | x, a)} v(j) \right\}$

- Reduce the effect of unlucky samples by using a fresh set:

> function **plan1**(x, p, r, γ)
> $v \leftarrow$prepare(p, r, γ)
> **return** plan0(x, v, p, r, γ)

Results for Planning

- p(y|x,a):
  - Lipschitz continuous (L_p) and bounded (K_p)
- r(x,a) :
  - bounded (K_r)
- $H(\epsilon) = K_r/(\epsilon(1-\gamma))$
- **Theorem** [Sze '01]: If $N=\text{poly}(d,\log(|A|),H(\epsilon),K_p,\log(L_p),\log(1/\delta))$, then
  - with probability $1-\delta$, the policy implemented by **plan0** is ϵ-optimal.
  - with probability 1, the policy implemented by **plan1** is ϵ-optimal.
- Improvements:
  - Dependence on $\log(L_p)$ not L_p; $\log(|A|)$ not $|A|$, no dependence on L_r!

A multiple-choice test..

- Why is not there a curse of dimensionality for RDM?

  A. Randomization is the cure to everything
  B. Class of MDPs is too small
  C. Expected error is small, variance is huge
  D. The result does not hold for control
  E. The hidden constants blow up anyway
  F. Something else

Why no curse of dimensionality??

- RDM uses a computational model different than that of Chow and Tsitsiklis!
  - One is allowed to use p,r at the time of answering "$V^*(x) = ?$, $\pi^*(x) = ?$"
- Why does this help?
  - $V_\pi = r_\pi + \gamma P_\pi \sum_t \gamma^t P_\pi^t r_\pi = r_\pi + \gamma P_\pi V_\pi$
  - Also explains why smoothness of the reward function is not required

Possible Improvements

- Reduce distribution mismatch
  - Once a good policy is computed, follow it to generate new points
    - How to do weighted importance sampling then??
    - Fit distribution & generate samples from the fitted distribution(?)
  - Repeat Z times
- Decide adaptively when to stop adding new points

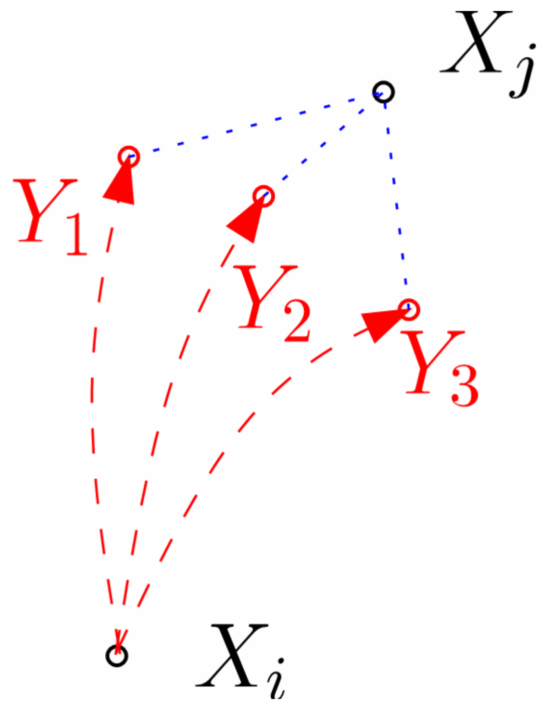
Planning with a Generative Model

Problem:

- Can generate transitions from anywhere
- Find a good policy!
- Be efficient!

Sampling based fitted value iteration

- Generative model
  - Cannot query p(y|x,a)
  - Can generate Y~p(.|x,a)
- Can we generalize RDM?
- Option A: Build model
- Option B: Use function approximation to propagate values
- [Samuel, 1959], [Bellman and Dreyfus, 1959], [Reetz,1977], [Keane and Wolpin, 1994],..



Single-sample version

Sampling based fitted value iteration – single sample

1: function **SFVI-SINGLE**$(N, M, K, \mu, \mathcal{F}, P, S)$
for $i = 1$ to N **do**
  Draw $X_i \sim \mu$, $Y_j^{X_i,a} \sim P(\cdot|X_i, a)$, $R_j^{X_i,a} \sim S(\cdot|X_i, a)$, $(j = 1, \ldots, M, a \in \mathcal{A})$
end for
$V \leftarrow 0$ // approximate value function
for $k = 1$ to K **do**
  $\hat{V}_i \leftarrow \max_{a \in \mathcal{A}} \left\{ \frac{1}{M} \sum_{j=1}^{M} \left(R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right) \right\}$
  $V \leftarrow \mathrm{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{N} (f(X_i) - \hat{V}_i)^2$ // fitting
end for
return V

[SzeMu '05]

Multi-sample version

Sampling based fitted value iteration – multi-sample variant

function **SFVI-MULTI**$(N, M, K, \mu, \mathcal{F}, P, S)$
$V \leftarrow 0$ // approximate value function
for $k = 1$ to K **do**
  for $i = 1$ to N **do**
    Draw $X_i \sim \mu$, $Y_j^{X_i,a} \sim P(\cdot|X_i, a)$, $R_j^{X_i,a} \sim S(\cdot|X_i, a)$, $(j = 1, \ldots, M,\ a \in \mathcal{A})$
  end for
  $\hat{V}_i \leftarrow \max_{a \in \mathcal{A}} \left\{ \frac{1}{M} \sum_{j=1}^{M} \left(R_j^{X_i,a} + \gamma V(Y_j^{X_i,a}) \right) \right\}$
  $V \leftarrow \mathrm{argmin}_{f \in \mathcal{F}} \sum_{i=1}^{N} (f(X_i) - \hat{V}_i)^2$ // fitting
end for
return V

[SzeMu ’05]

Assumptions

- $C(\mu) = ||dP(.|x,a)/d\mu||_\infty < +\infty$
  - μ uniform: $dP/d\mu = p(.|x,a)$; density kernel
  - This was used by the previous results
  - Rules out deterministic systems and systems with jumps

Loss bound

$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq$$

$$\frac{2\gamma}{(1-\gamma)^2} \Bigg\{ C(\mu)^{1/p} \Bigg[d(T\mathcal{F}, \mathcal{F}) +$$

$$c_1 \left(\frac{\mathcal{E}}{N} (\log(N) + \log(K/\delta)) \right)^{1/2p} +$$

$$c_2 \left(\frac{1}{M} (\log(N|A|) + \log(K/\delta)) \right)^{1/2} \Bigg] +$$

$$c_3 \gamma^K K_{\max} \Bigg\}$$

[SzeMu ’05]

The Bellman error of function sets

- Bound is in temrs of the "distance of the functions sets $\mathcal{F}$ and T$\mathcal{F}$:
  - $d(T\mathcal{F}, \mathcal{F}) = \inf_{f\in \mathcal{F}} \sup_{V\in \mathcal{F}} ||TV-f||_{p,\mu}$
- "Bellman error on $\mathcal{F}$"
- $\mathcal{F}$ should be large to make $d(T\mathcal{F}, \mathcal{F})$ small
- If MDP is "smooth", TV is smooth for any bounded(!) V
- Smooth functions can be well-approximated
- → Assume MDP is smooth

Metric Entropy

- The bound depends on the metric entropy, $\mathcal{E}=\mathcal{E}(\mathcal{F})$.
  - Metric entropy: ‘capacity measure’, similar to VC-dimension
- Metric entropy increases with $\mathcal{F}$!
- Previously we concluded that $\mathcal{F}$ should be big
- ???
  - Smoothness
  - RKHS

RKHS Bounds

- Linear models (~RKHS):
  $\mathcal{F} = \{ w^T \phi : ||w||_1 \leq A \}$
- [Zhang, '02]: $\mathcal{E}(\mathcal{F}) = O(\log N)$
- This is independent of dim(ϕ)!

- **Corollary**: Sample complexity of FVI is polynomial for "sparse" MDPs
  - Cf. [Chow and Tsitsiklis '89]
- Extension to control? Yes

Improvements

- Model selection
  - How to choose $\mathcal{F}$?
  - Choose as large an $\mathcal{F}$ as needed!
    - Regularization
    - Model-selection
    - Aggregation
    - ..
- Place base-points better
  - Follow policies
  - No need to fit densities to them!

References

- [Ng & Jordan ’00] A.Y. Ng and M. Jordan: PEGASUS: A policy search method for large MDPs and POMDPs, UAI 2000.
- [R. Bellman ’57] R. Bellman: *Dynamic Programming*. Princeton Univ. Press, 1957.
- [Chow & Tsitsiklis ’89] C.S. Chow and J.N. Tsitsiklis: The complexity of dynamic programming, *Journal of Complexity*, 5:466—488, 1989.
- [Kearns et al. ’02] M.J. Kearns, Y. Mansour, A.Y. Ng: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning* 49: 193—208, 2002.
- [KoSze ’06] L. Kocsis and Cs. Szepesvári: Bandit based Monte-Carlo planning. ECML, 2006.
- [Auer et al. ’02] P. Auer, N. Cesa-Bianchi and P. Fischer: Finite time analysis of the multiarmed bandit problem, *Machine Learning*, 47:235—256, 2002.
- [Peret and Garcia ’04] L. Peret & F. Garcia: On-line search for solving Markov decision processes via heuristic sampling. ECAI, 2004.
- [Chang et al. ’05] H.S. Chang, M. Fu, J. Hu, and S.I. Marcus: An adaptive sampling algorithm for solving Markov decision processes. *Operations Research*, 53:126—139, 2005.
- [Rust ’97] J. Rust, 1997, Using randomization to break the curse of dimensionality, *Econometrica*, 65:487—516, 1997.
- [Sze ’01] Cs. Szepesvári: Efficient approximate planning in continuous space Markovian decision problems, *AI Communications*, 13:163 - 176, 2001.
- [SzeMu ’05] Cs. Szepesvári and R. Munos: Finite time bounds for sampling based fitted value iteration, ICML, 2005.
- [Zhang ’02] T. Zhang: Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2:527–550, 2002.

Reinforcement Learning: Learning Algorithms

Csaba Szepesvári
University of Alberta

Kioloa, MLSS'08

Slides: http://www.cs.ualberta.ca/~szepesva/MLSS08/

Contents

- Defining the problem(s)
- Learning optimally
- Learning a good policy
  - Monte-Carlo
  - Temporal Difference (bootstrapping)
  - Batch – fitted value iteration and relatives

The Learning Problem

- The MDP is unknown but the agent can interact with the system
- Goals:
  - Learn an optimal policy
    - Where do the samples come from?
      - Samples are generated externally
      - The agent interacts with the system to get the samples (“active learning”)
    - Performance measure: What is the performance of the policy obtained?
  - Learn optimally: Minimize regret while interacting with the system
    - Performance measure: loss in rewards due to not using the optimal policy from the beginning
    - Exploration vs. exploitation

Learning from Feedback

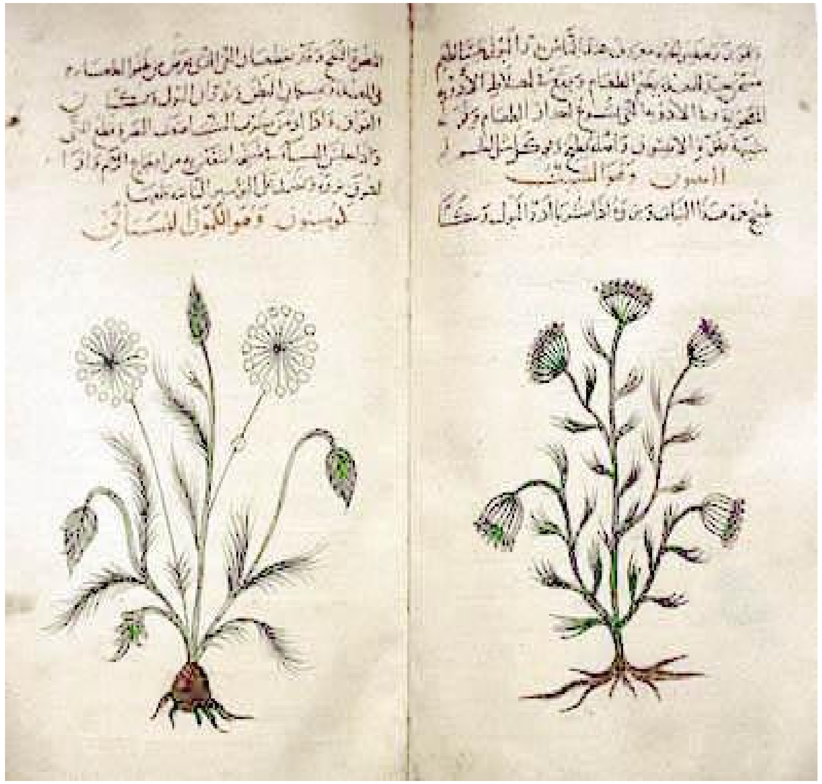
- A protocol for prediction problems:
  - x_t – situation (observed by the agent)
  - $y_t \in Y$ – value to be predicted
  - $p_t \in Y$ – predicted value (can depend on all past values $\Rightarrow$ learning!)
  - $r_t(x_t,y_t,y)$ – value of predicting y
    loss of learner: $\lambda_t = r_t(x_t,y_t,y) - r_t(x_t, y_t,p_t)$
- Supervised learning:
  agent is told y_t, $r_t(x_t,y_t,.)$
  - Regression: $r_t(x_t,y_t,y) = -(y-y_t)^2 \rightarrow \lambda_t = (y_t - p_t)^2$
- Full information prediction problem:
  $\forall y \in Y$, $r_t(x_t,y)$ is communicated to the agent, but not y_t
- Bandit (partial information) problem:
  $r_t(x_t,p_t)$ is communicated to the agent only

Learning Optimally

- Explore or exploit?
- Bandit problems
  - Simple schemes
  - Optimism in the face of uncertainty (OFU) → UCB
- Learning optimally in MDPs with the OFU principle

Learning Optimally: Exploration vs. Exploitation

- Two treatments
- Unknown success probabilities
- Goal:
  - find the best treatment while loosing few patients
- Explore or exploit?



Exploration vs. Exploitation: Some Applications

- Simple processes:
  - Clinical trials
  - Job shop scheduling (random jobs)
  - What ad to put on a web-page
- More complex processes (memory):
  - Optimizing production
  - Controlling an inventory
  - Optimal investment
  - Poker
  - ..

Bernoulli Bandits

- Payoff is 0 or 1

- Arm 1:

  0 , 1 , 0 , 0 , …

- Arm 2:

  1 , 1 , 0 , 1 , …

Some definitions

Now: t=9
$T_1(t\text{-}1) = 4$
$T_2(t\text{-}1) = 4$
$A_1 = 1$, $A_2 = 2$, …

- Payoff is 0 or 1
- Arm 1:
  0 , 1 , 0 , 0 , …
- Arm 2:
  1 , 1 , 0 , 1 , …

$$\hat{L}_T \stackrel{\text{def}}{=} \sum_{t=1}^{T} R_t(k^*) - \sum_{t=1}^{T} R_{T_{A_t}(t)}(A_t)$$

The Exploration/Exploitation Dilemma

- Action values: $Q^*(a) = E[R_t(a)]$
- Suppose you form estimates

$$Q_t(a) \approx Q^*(a)$$

- The greedy action at t is:

$$A_t^* = \mathrm{argmax}_a \, Q_t(a)$$

- Exploitation: When the agent chooses to follow A_t^*
- Exploration: When the agent chooses to do something else

Action-Value Methods

- Methods that adapt action-value estimates and nothing else
- How to estimate action-values?
- Sample average:

$$Q_t(a) = \frac{R_1(a) + \ldots + R_{T_t(a)}(a)}{T_t(a)}$$

- Claim: $\lim_{t\to\infty} Q_t(a) = Q^*(a)$, if $n_t(a) \to \infty$
- Why??

ε-Greedy Action Selection

- Greedy action selection:

$$A_t = A_t^* = \text{argmax}_a \, Q_t(a)$$

- ε-Greedy:

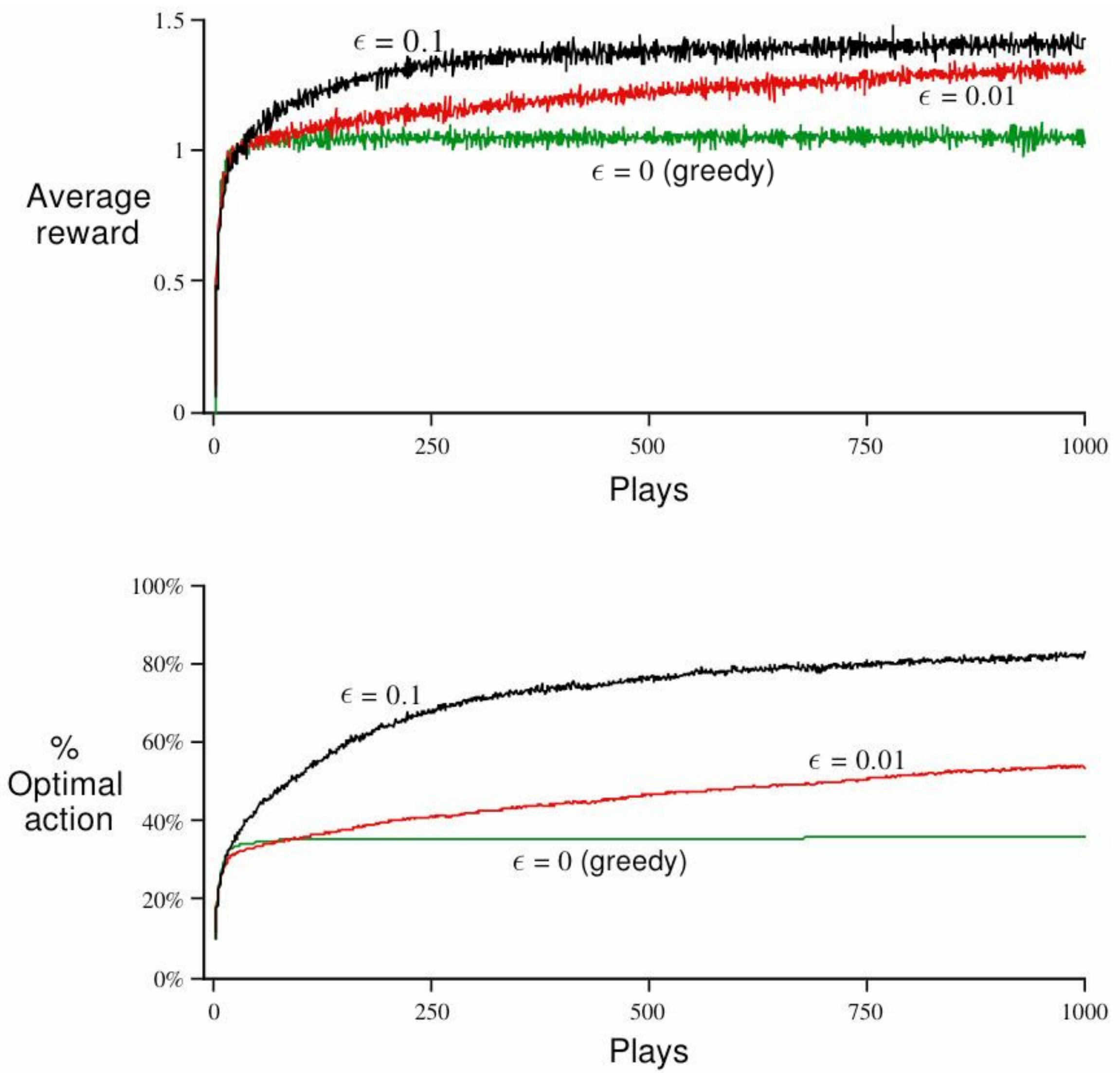
$$A_t = \begin{cases} A_t^* & \text{with probability } 1 - \varepsilon \\ \text{random action} & \text{with probability } \quad \varepsilon \end{cases}$$

. . . the simplest way to “balance” exploration and exploitation

10-Armed Testbed

- n = 10 possible actions
- Repeat 2000 times:
  - $Q^*(a) \sim N(0,1)$
  - Play 1000 rounds
    - $R_t(a) \sim N(Q^*(a),1)$

ε-Greedy Methods on the 10-Armed Testbed



Softmax Action Selection

- Problem with ϵ-greedy: Neglects action values
- Softmax idea: grade action probs. by estimated values.
- Gibbs, or Boltzmann action selection, or exponential weights:

$$\mathbb{P}\left(A_t = a \middle| H_t\right) = \frac{e^{Q_t(a)/\tau_t}}{\sum_b e^{Q_t(b)/\tau_t}}$$

  - $\tau = \tau_t$ is the "computational temperature"

Incremental Implementation

- Sample average:

$$Q_t(a) = \frac{R_1(a) + \ldots + R_{T_t(a)}(a)}{T_t(a)}$$

- Incremental computation:

$$Q_{t+1}(A_t) = Q_t(A_t) + \frac{1}{t+1}\left(R_{t+1} - Q_t(A_t)\right)$$

- Common update rule form:

NewEstimate **=** *OldEstimate*
+ *StepSize*[*Target* **–** *OldEstimate*]

[Auer et al. ’02]

UCB: Upper Confidence Bounds

- Principle: Optimism in the face of uncertainty
- Works when the environment is not adversary
- Assume rewards are in [0,1]. Let

$$A_t = \mathrm{argmax}_a \left\{ Q_t(a) + \sqrt{\frac{p \log(t)}{2T_t(a)}} \right\} \quad (p>2)$$

- For a stationary environment, with iid rewards this algorithm is hard to beat!
- Formally: regret in T steps is $O(\log T)$
- Improvement: Estimate variance, use it in place of p [AuSzeMu ’07]
- This principle can be used for achieving small regret in the full RL problem!

UCRL2: UCB Applied to RL

- [Auer, Jaksch & Ortner '07]
- **Algorithm UCRL2**(δ):
  - Phase initialization:
    - Estimate mean model p_0 using maximum likelihood (counts)
    - $C := \{ p \mid \|p(.|x,a) - p_0(.|x,a)\| \leq c\, |X| \log(|A|T/\text{delta}) / N(x,a) \}$
    - $p' := \text{argmax}_p\, \rho^*(p)$, $\pi := \pi^*(p')$
    - $N_0(x,a) := N(x,a),\ \forall\, (x,a) \in X \times A$
  - Execution
    - Execute π until some (x,a) have been visited at least $N_0(x,a)$ times in this phase

UCRL2 Results

- **Def**: Diameter of an MDP M:
  $D(M) = \max_{x,y} \min_{\pi} E[\, T(x \rightarrow y; \pi)\,]$
- **Regret bounds**
  - Lower bound:
    $E[L_n] = \Omega((D\, |X|\, |A|\, T)^{1/2})$
  - Upper bounds:
    - w.p. $1-\delta/T$,
      $L_T \leq O(D\, |X|\, (|A|\, T \log(|A|T/\delta)^{1/2})$
    - w.p. $1-\delta$,
      $L_T \leq O(D^2\, |X|^2\, |A| \log(|A|T/\delta)/\Delta)$

      Δ =performance gap between best and second best policy

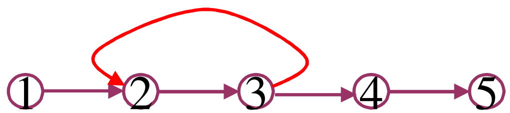
Learning a Good Policy

- Monte-Carlo methods
- Temporal Difference methods
  - Tabular case
  - Function approximation
- Batch learning

Learning a good policy

- Model-based learning
  - Learn p,r
  - “Solve” the resulting MDP
- Model-free learning
  - Learn the optimal action-value function and (then) act greedily
  - Actor-critic learning
  - Policy gradient methods
- Hybrid
  - Learn a model and mix planning and a model-free method; e.g. Dyna

Monte-Carlo Methods



- Episodic MDPs!
- Goal: Learn $V^\pi(.)$
- $V^\pi(x)$
  $= E_\pi[\ \sum_t \gamma^t R_t | X_0 = x]$
- (X_t, A_t, R_t):
  -- trajectory of π
- Visits to a state
  - $f(x) = \min \{t | X_t = x\}$
    - First visit
  - $E(x) = \{ t \mid X_t = x \}$
    - Every visit
- Return:

$S(t) = \gamma^0 R_t + \gamma^1 R_{t+1} + \ldots$

- K independent trajectories ➔ $S^{(k)}, E^{(k)}, f^{(k)}$, $k=1..K$
- **First-visit MC:**
  - Average over
  $\{ S^{(k)}(f^{(k)}(x)) : k=1..K \}$
- **Every-visit MC:**
  - Average over
  $\{ S^{(k)}(t) : k=1..K, t \in E^{(k)}(x) \}$
- **Claim**: Both converge to $V^\pi(.)$
- From now on $S_t = S(t)$

[Singh & Sutton ’96]

Learning to Control with MC

- Goal: Learn to behave optimally
- Method:
  - Learn $Q^{\pi}(x,a)$
  - ..to be used in an approximate policy iteration (PI) algorithm
- Idea/algorithm:
  - Add randomness
    - Goal: all actions are sampled eventually infinitely often
    - e.g., ϵ-greedy or exploring starts
  - Use the first-visit or the every-visit method to estimate $Q^{\pi}(x,a)$
  - Update policy
    - Once values converged
      .. or ..
    - Always at the states visited

Monte-Carlo: Evaluation

- Convergence rate: $Var(S(0)|X=x)/N$
- Advantages over DP:
  - Learn from interaction with environment
  - No need for full models
  - No need to learn about ALL states
  - Less harm by Markovian violations (no bootstrapping)
- Issue: maintaining sufficient exploration
  - exploring starts, soft policies

[Samuel, '59], [Holland '75], [Sutton '88]

Temporal Difference Methods

- Every-visit Monte-Carlo:
  - $V(X_t) \leftarrow V(X_t) + \alpha_t(X_t)\ (S_t - V(X_t))$
- Bootstrapping
  - $S_t = R_t + \gamma\ S_{t+1}$
  - $S_t' = R_t + \gamma\ V(X_{t+1})$
- TD(0):
  - $V(X_t) \leftarrow V(X_t) + \alpha_t(X_t)\ (\ S_t' - V(X_t)\ )$
- Value iteration:
  - $V(X_t) \leftarrow E[\ S_t' \mid X_t\ ]$
- **Theorem:** Let V_t be the sequence of functions generated by TD(0). Assume $\forall$ x, w.p.1 $\sum_t \alpha_t(x)=\infty$, $\sum_t \alpha_t^2(x)<+\infty$. Then $V_t \rightarrow V_\pi$ w.p.1
- **Proof:** Stochastic approximations: $V_{t+1}=T_t(V_t,V_t)$, $U_{t+1}=T_t(U_t,V_\pi) \rightarrow TV_\pi$. [Jaakkola et al., '94, Tsitsiklis '94, SzeLi99]

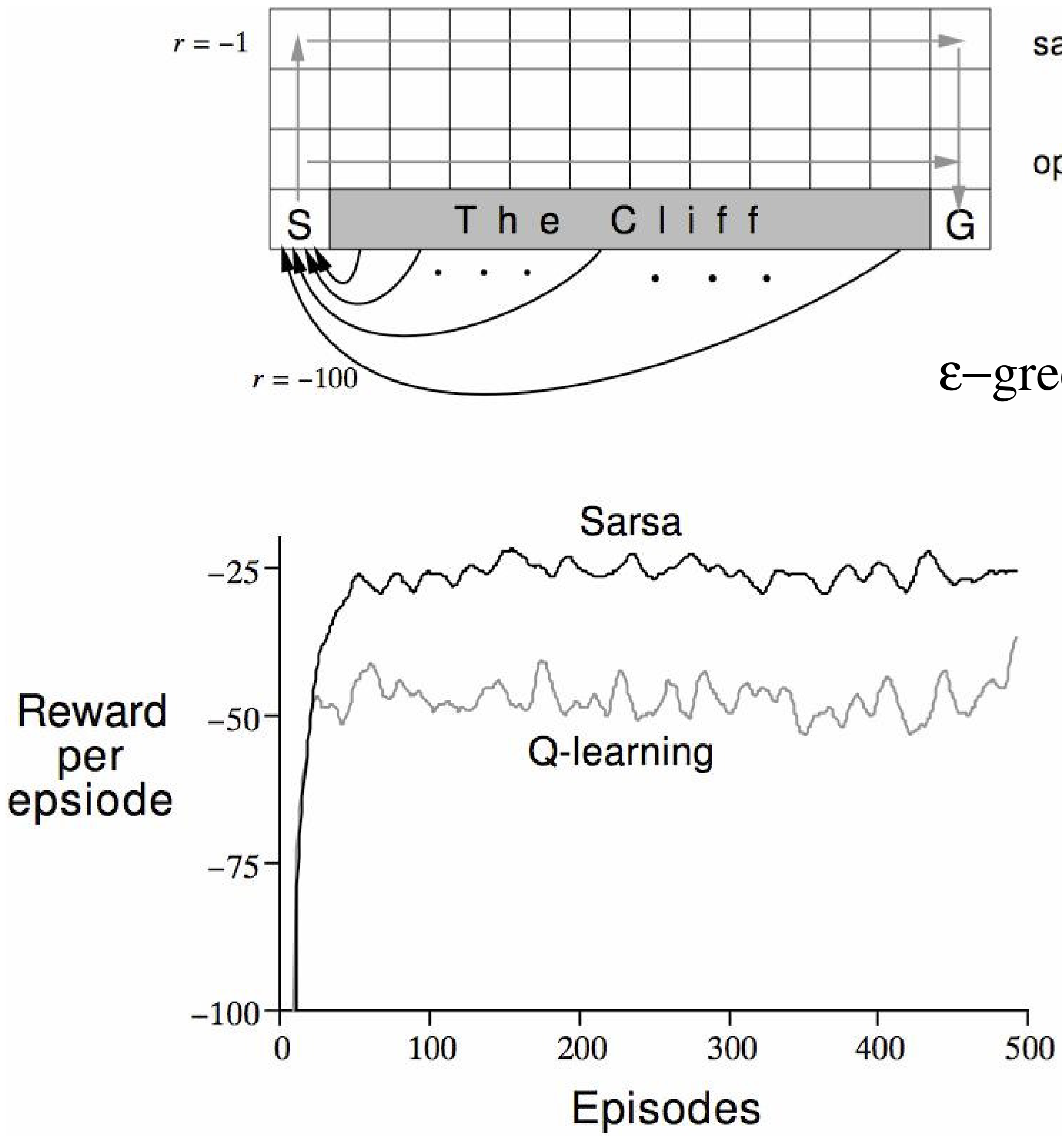
TD or MC?

- TD advantages:
  - can be fully incremental, i.e.,
    learn before knowing the final outcome
    - Less memory
    - Less peak computation
  - learn without the final outcome
    - From incomplete sequences
- MC advantage:
  - Less harm by Markovian violations
- Convergence rate?
  - Var(S(0)|X=x) decides!

Learning to Control with TD

- Q-learning [Watkins '90]:
  $Q(X_t,A_t) \leftarrow Q(X_t,A_t) + \alpha_t(X_t,A_t)\,\{R_t+\gamma \max_a Q(X_{t+1},a)-Q(X_t,A_t)\}$
- **Theorem**: Converges to Q^* [JJS'94, Tsi'94,SzeLi99]
- SARSA [Rummery & Niranjan '94]:
  - $A_t \sim \text{Greedy}_\epsilon(Q,X_t)$
  - $Q(X_t,A_t) \leftarrow Q(X_t,A_t) + \alpha_t(X_t,A_t)\,\{R_t+\gamma Q(X_{t+1},A_{t+1})-Q(X_t,A_t)\}$
- Off-policy (Q-learning) vs. on-policy (SARSA)
- Expecti-SARSA
- Actor-Critic [Witten '77, Barto, Sutton & Anderson '83, Sutton '84]

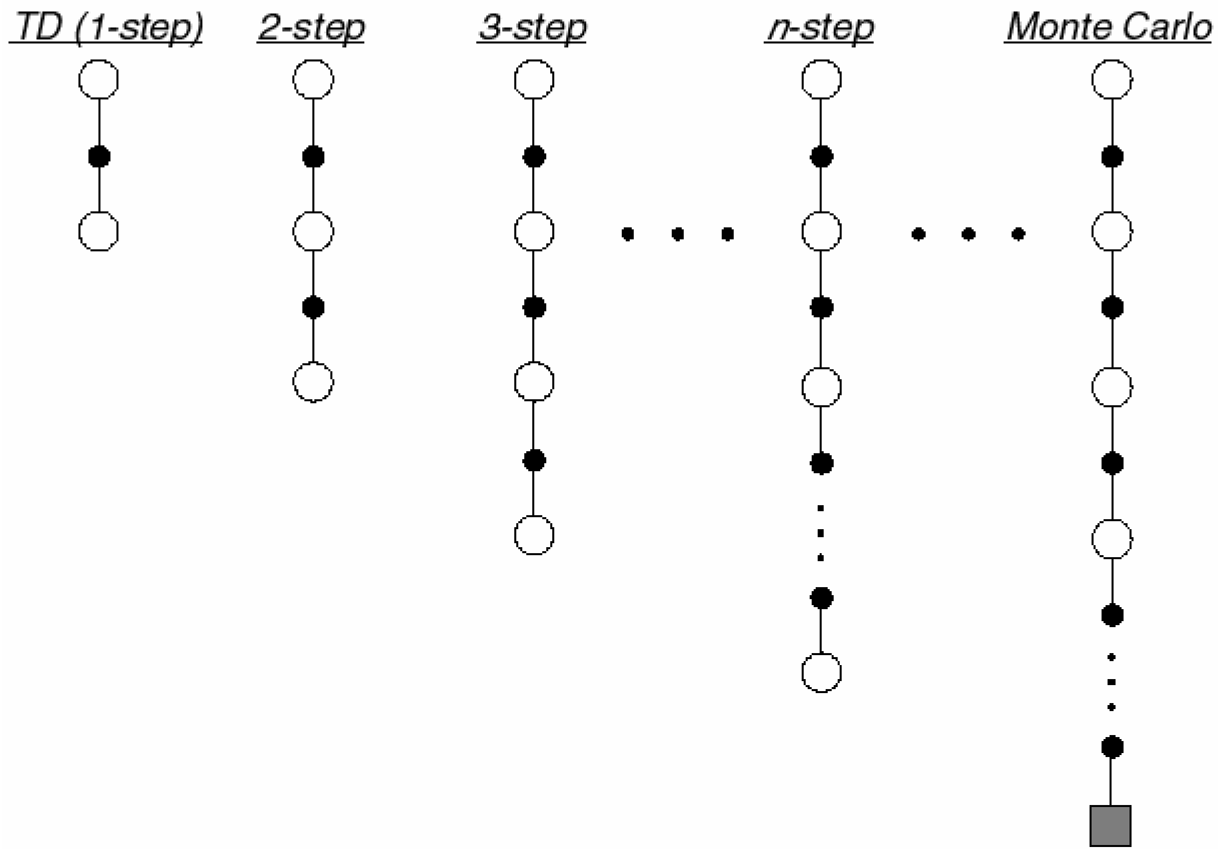
Cliffwalking



ε−greedy, $\varepsilon = 0.1$

N-step TD Prediction

- Idea: Look farther into the future when you do TD backup (1, 2, 3, …, n steps)



N-step TD Prediction

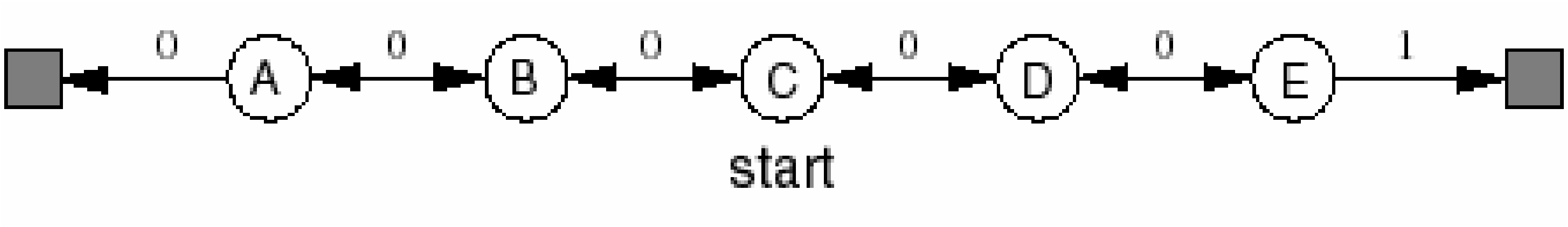
- Monte Carlo:
  - $S_t = R_t + \gamma R_{t+1} + .. \gamma^{T-t} R_T$

- TD: $S_t^{(1)} = R_t + \gamma V(X_{t+1})$
  - Use V to estimate remaining return
- n-step TD:
  - 2 step return:
    - $S_t^{(2)} = R_t + \gamma R_{t+1} + \gamma^2 V(X_{t+2})$
  - n-step return:
    - $S_t^{(n)} = R_t + \gamma R_{t+1} + \ldots + \gamma^n V(X_{t+n})$

Learning with n-step Backups

- Learning with n-step backups:
  - $V(X_t) \leftarrow V(X_t) + \alpha_t(S_t^{(n)} - V(X_t))$
- n: controls how much to bootstrap

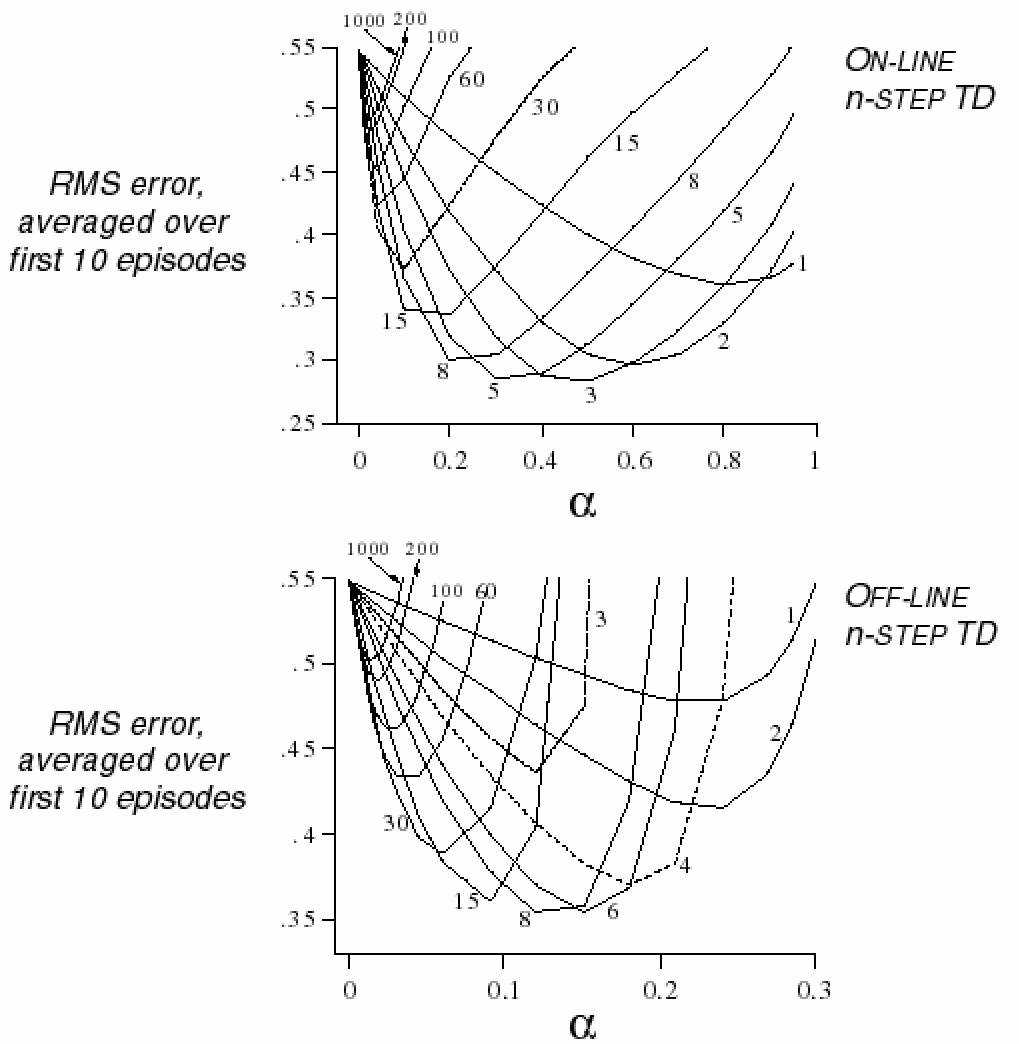
Random Walk Examples



- How does 2-step TD work here?
- How about 3-step TD?

A Larger Example

- Task: 19 state random walk
- Do you think there is an optimal n? for everything?

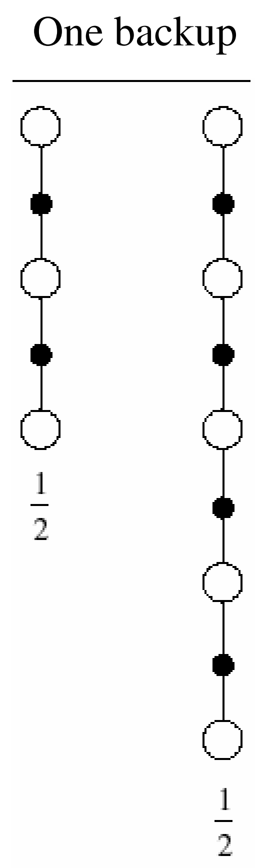


Averaging N-step Returns

- Idea: backup an average of several returns
  - e.g. backup half of 2-step and half of 4-step:

$$\overline{R_t} = \frac{1}{2}R_t^{(2)} + \frac{1}{2}R_t^{(4)}$$

- "complex backup"



Forward View of TD(λ)

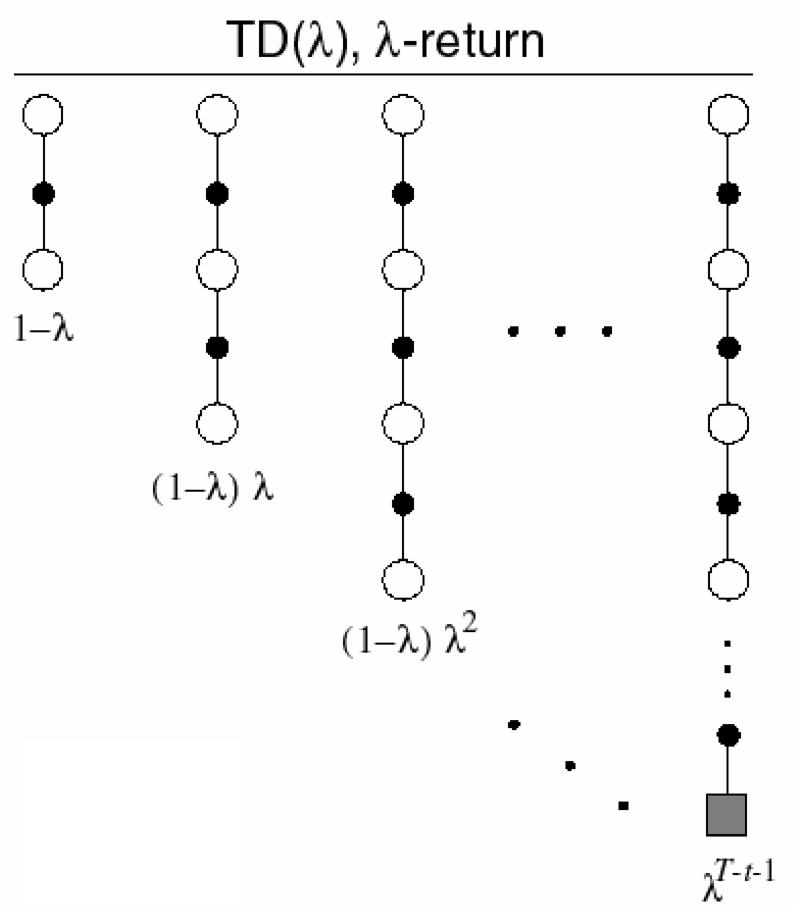
- Idea: Average over multiple backups
- λ-return:

$S_t^{(\lambda)} = (1-\lambda) \sum_{n=0..\infty} \lambda^n S_t^{(n+1)}$

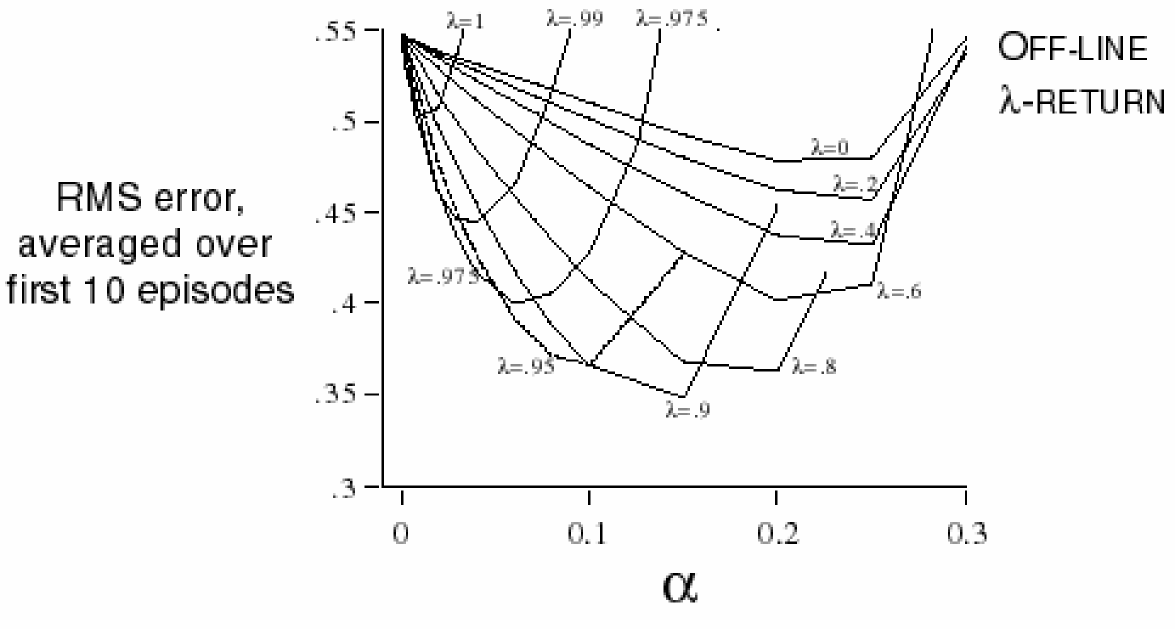
- TD(λ):

$\Delta V(X_t) = \alpha_t(S_t^{(\lambda)} - V(X_t))$

- Relation to TD(0) and MC
  - $\lambda=0$ ➔ TD(0)
  - $\lambda=1$ ➔ MC



λ-return on the Random Walk



- Same 19 state random walk as before
- Why intermediate values of λ are best?

[Sutton '88, Singh & Sutton '96]

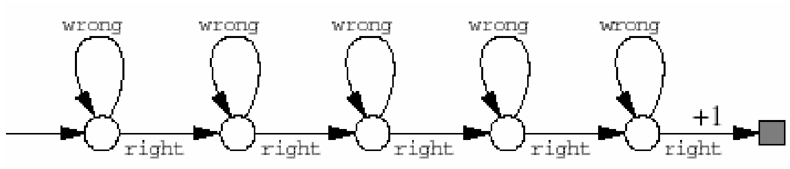
Backward View of TD(λ)

$$\delta_t = R_t + \gamma V(X_{t+1}) - V(X_t)$$

$$V(x) \leftarrow V(x) + \alpha_t \delta_t e(x)$$

$$e(x) \leftarrow \gamma \lambda e(x) + I(x=X_t)$$

- Off-line updates ➔Same as FW TD(λ)
- e(x): eligibility trace
  - Accumulating trace
  - Replacing traces speed up convergence:
    - $e(x) \leftarrow \max(\gamma\lambda e(x), I(x=X_t))$



Function Approximation with TD

Gradient Descent Methods

transpose

$$\theta_t = (\theta_t(1), \ldots, \theta_t(n))^T$$

- Assume V_t is a differentiable function of θ:

$$V_t(x) = V(x;\theta).$$

- Assume, for now, training examples of the form:

$$\{ (X_t, V^\pi(X_t)) \}$$

Performance Measures

- Many are applicable but…
- a common and simple one is the mean-squared error (MSE) over a distribution *P*:

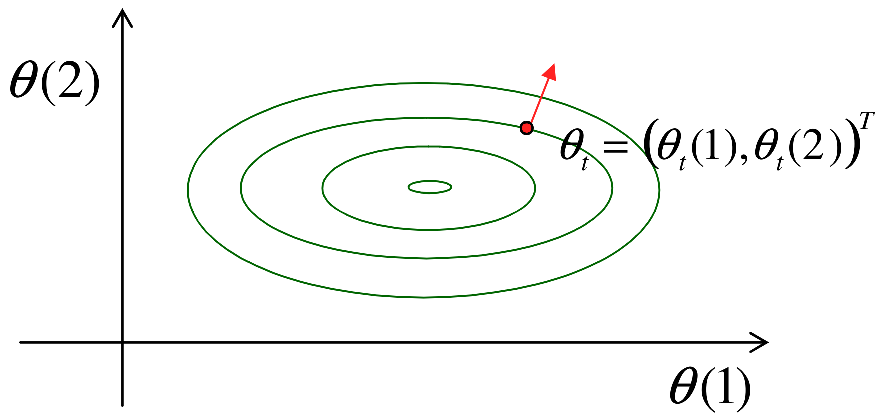
$$L(\theta) = \sum_{x \in X} P(x) \left(V^{\pi}(x) - V(x;\theta)\right)^2$$

- Why *P*?
- Why minimize MSE?
- Let us assume that *P* is always the distribution of states at which backups are done.
- The **on-policy distribution**: the distribution created while following the policy being evaluated. Stronger results are available for this distribution.

Gradient Descent

- Let L be any function of the parameters. Its gradient at any point θ in this space is:

$$\nabla_\theta L = \left(\frac{\partial L}{\partial \theta(1)}, \frac{\partial L}{\partial \theta(2)}, \ldots, \frac{\partial L}{\partial \theta(n)} \right)^T$$



- Iteratively move down the gradient:

$$\theta_{t+1} = \theta_t - \alpha_t \left(\nabla_\theta L \right) \big|_{\theta=\theta_t}$$

Gradient Descent in RL

- Function to descent on:

$$L(\theta) = \sum_{x \in X} P(x) \left(V^{\pi}(x) - V(x;\theta)\right)^2$$

- Gradient:

$$\nabla_\theta L(\theta) = -2 \sum_{x \in X} P(x) \left(V^{\pi}(x) - V(x;\theta)\right) \nabla_\theta V(x;\theta)$$

- Gradient descent procedure:

$$\theta_{t+1} = \theta_t + \alpha_t \left(V^{\pi}(X_t) - V(X_t;\theta_t)\right) \nabla_\theta V(X_t;\theta_t)$$

- Bootstrapping with S_t'

$$\theta_{t+1} = \theta_t + \alpha_t \left(S_t' - V(X_t;\theta_t)\right) \nabla_\theta V(X_t;\theta_t)$$

- TD(λ) (forward view):

$$\theta_{t+1} = \theta_t + \alpha_t \left(S_t^{\lambda} - V(X_t;\theta_t)\right) \nabla_\theta V(X_t;\theta_t)$$

[Sutton '84, '88, Tsitsiklis & Van Roy '97]

Linear Methods

- Linear FAPP: $V(x;\theta) = \theta^T \phi(x)$
- $\nabla_\theta V(x;\theta) = \phi(x)$
- Tabular representation:

  $$\phi(x)_y = I(x=y)$$
- Backward view:

  $$\delta_t = R_t + \gamma V(X_{t+1}) - V(X_t)$$
  $$\theta \leftarrow \theta + \alpha_t \delta_t e$$
  $$e \leftarrow \gamma \lambda e + \nabla_\theta V(X_t;\theta)$$
- **Theorem** [TsiVaR'97]: V_t converges to V s.t. $||V-V_\pi||_{D,2} \leq ||V_\pi-\Pi V_\pi||_{D,2}/(1-\gamma)$.

[Rummery & Niranjan '94]

Control with FA

- Learning state-action values

  Training examples:

  $$\{((X_t, A_t), Q^*(X_t, A_t) + \text{noise}_t)\}$$

- The general gradient-descent rule:

  $$\theta_{t+1} = \theta_t + \alpha_t \left(S_t - Q(X_t, A_t; \theta_t)\right) \nabla_\theta Q(X_t, A_t; \theta_t)$$

- Gradient-descent Sarsa(λ)

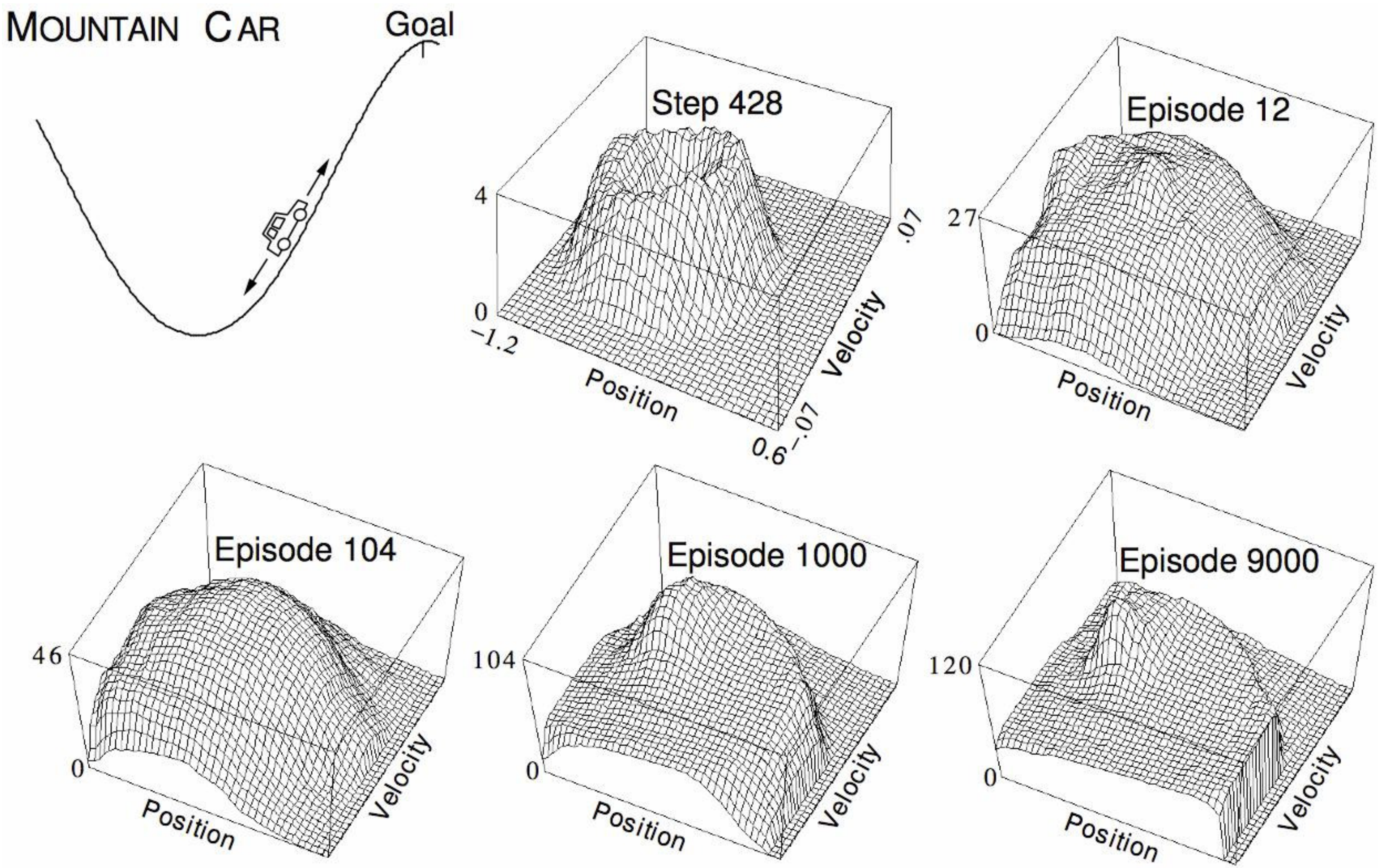
$$\theta_{t+1} = \theta_t + \alpha_t \delta_t e_t$$

where

$$\delta_t = R_t + \gamma Q(X_{t+1}, A_{t+1}; \theta_t) - Q_t(X_t, A_t; \theta_t)$$

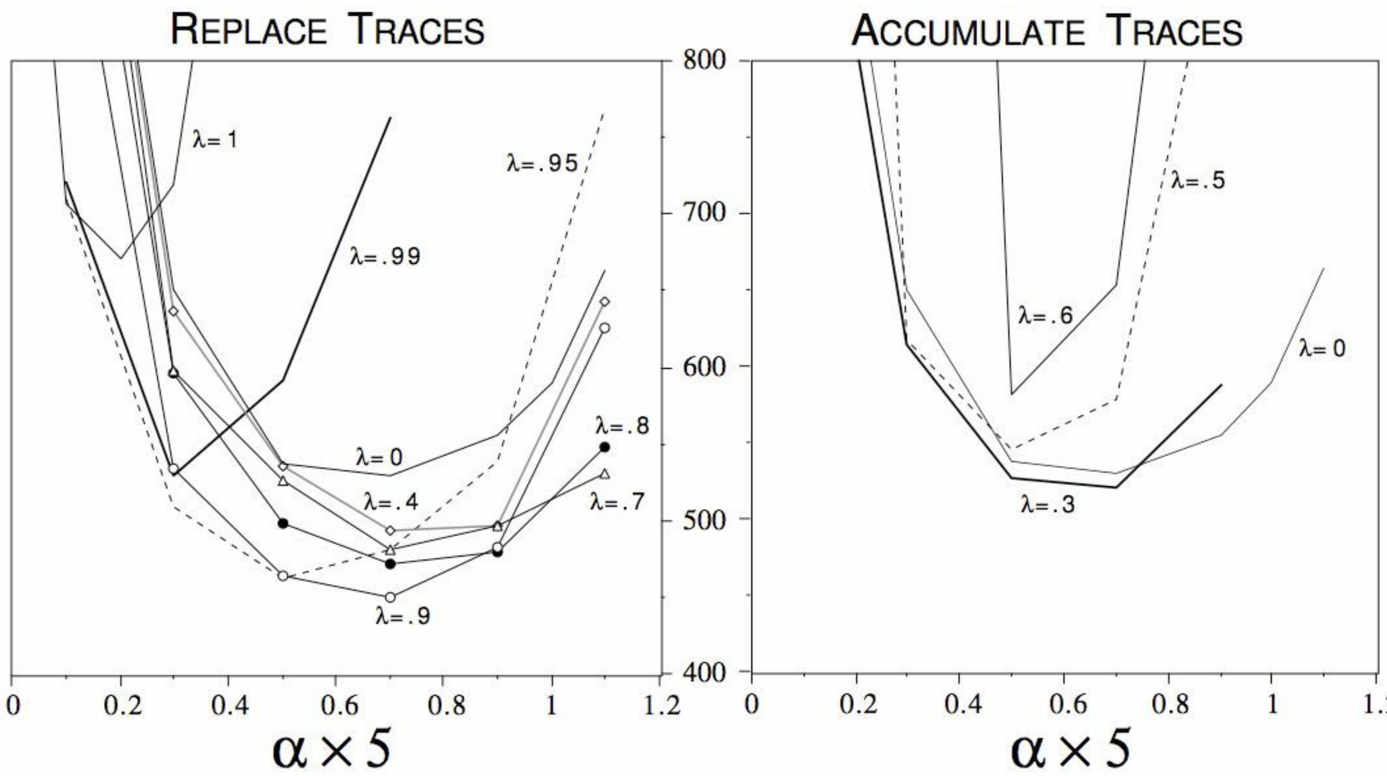
$$e_t = \gamma \lambda e_{t-1} + \nabla_\theta Q(X_t, A_t; \theta)$$

Mountain-Car Task



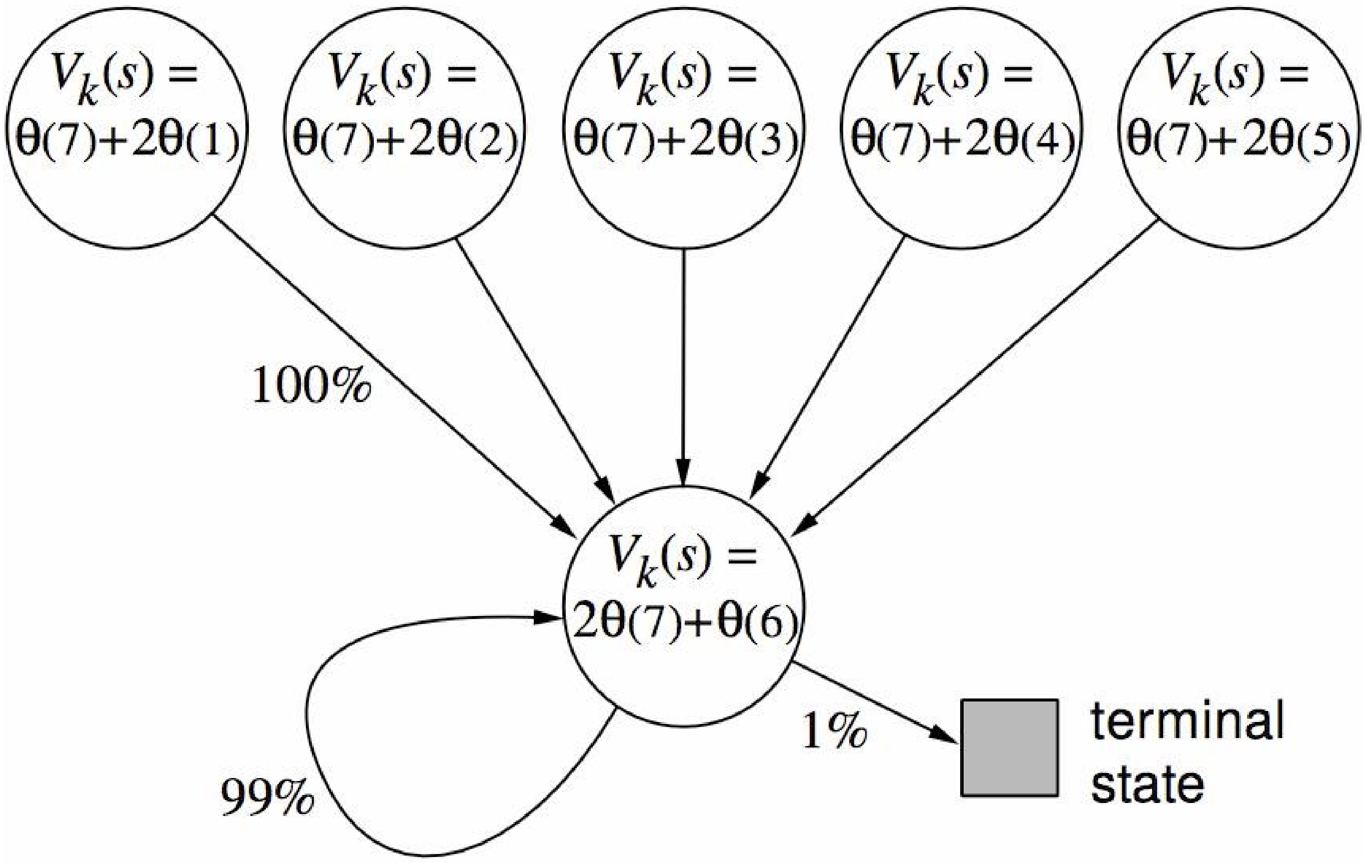
Mountain-Car Results

Steps per episode
averaged over
first 20 trials
and 30 runs

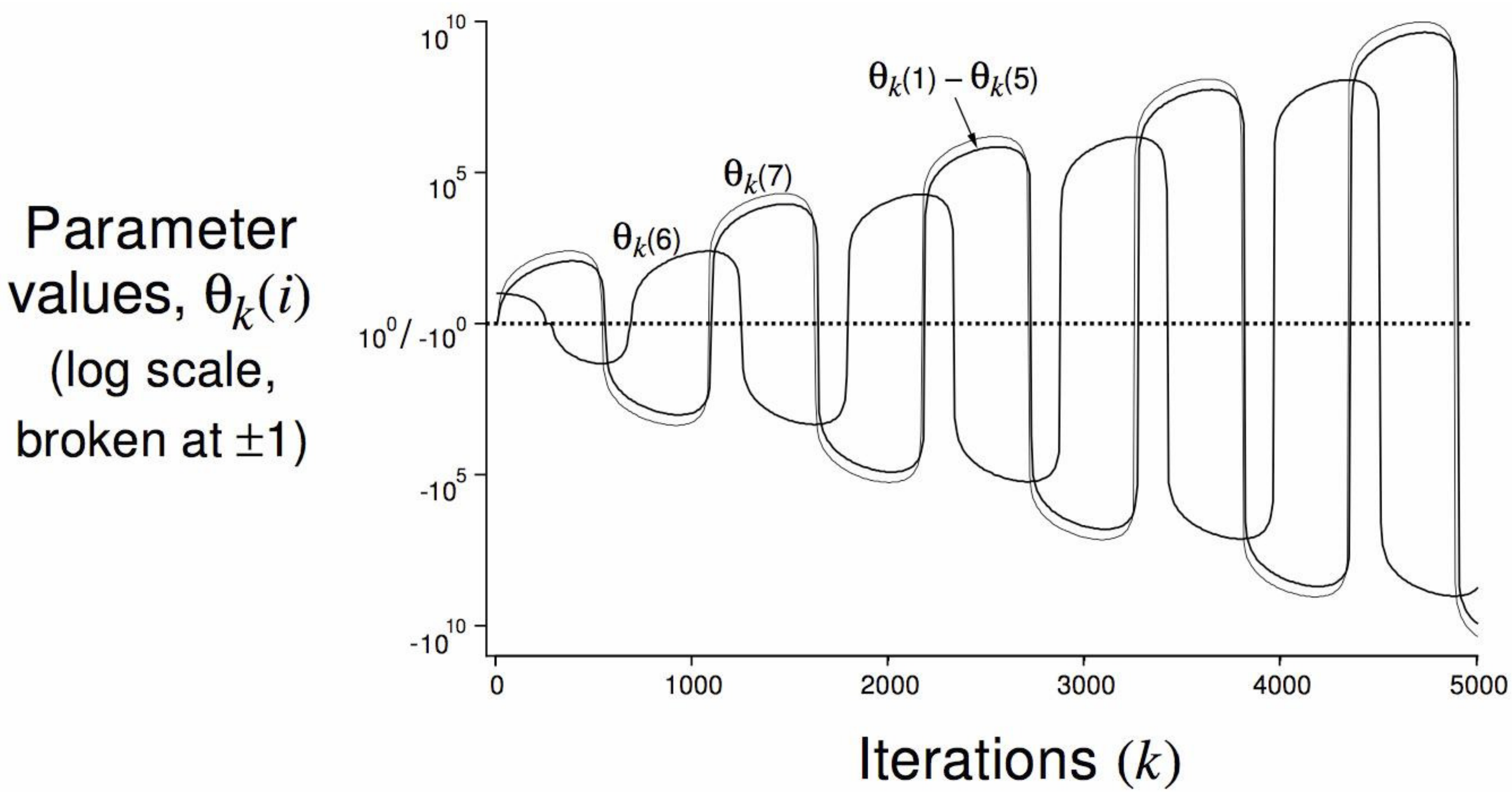


[Baird '95]

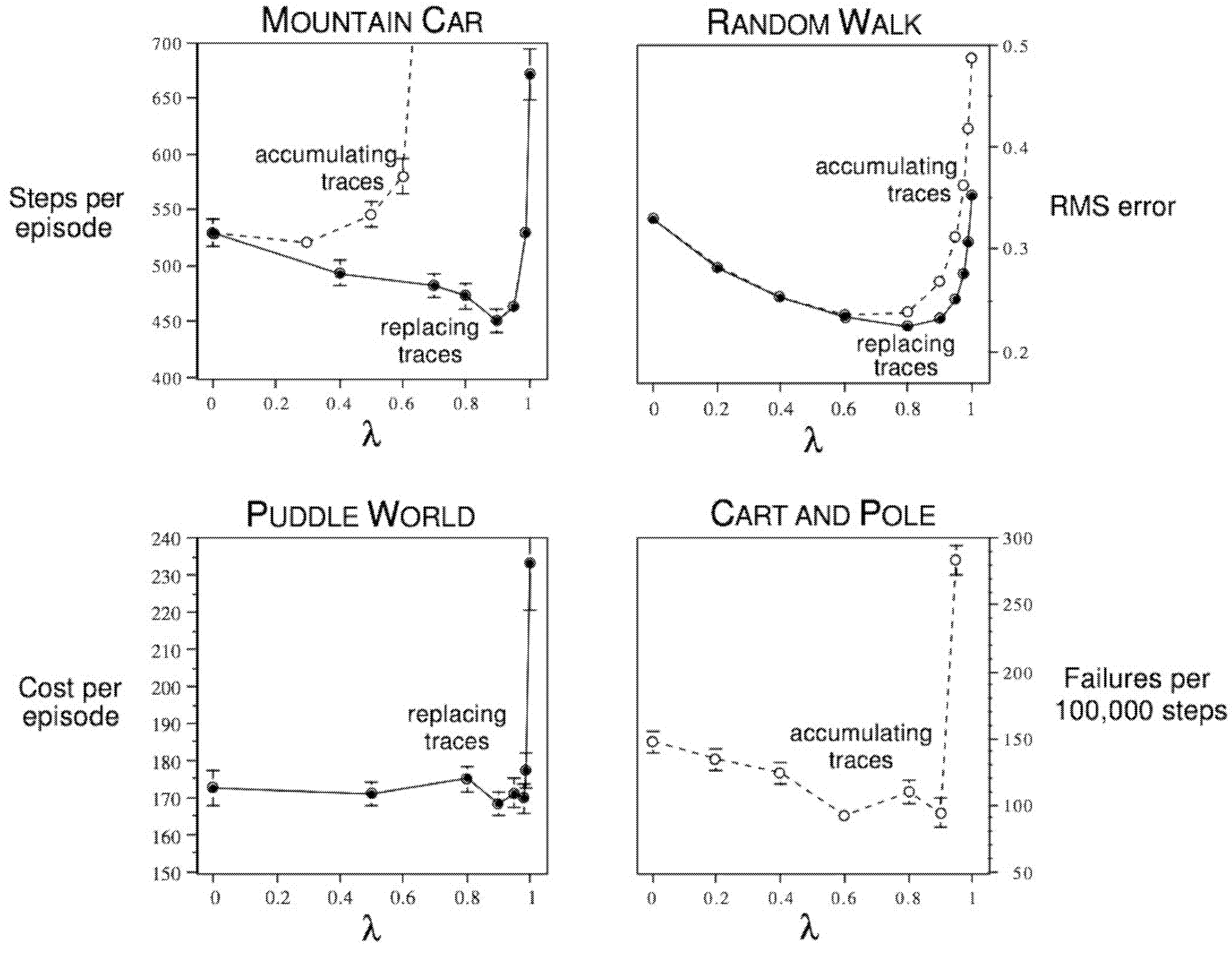
Baird's Counterexample: Off-policy Updates Can Diverge



Baird's Counterexample Cont.



Should We Bootstrap?



Batch Reinforcement Learning

Batch RL

- Goal: Given the trajectory of the behavior policy π_b

  $X_1, A_1, R_1, \ldots, X_t, A_t, R_t, \ldots, X_N$

  compute a good policy!
- "Batch learning"
- Properties:
  - Data collection is not influenced
  - Emphasis is on the quality of the solution
  - Computational complexity plays a secondary role
- Performance measures:
  - $||V^*(x) - V_\pi(x)||_\infty = \sup_x |V^*(x) - V_\pi(x)|$
    $= \sup_x V^*(x) - V_\pi(x)$
  - $||V^*(x) - V_\pi(x)||^2 = \int (V^*(x) - V_\pi(x))^2 \, d\mu(x)$

[Bradtke, Barto ’96], [Lagoudakis, Parr ’03], [AnSzeMu ’07]

Solution methods

- Build a model
- Do not build a model, but find an approximation to Q^*
  - using value iteration => fitted Q-iteration
  - using policy iteration =>
    - Policy evaluated by approximate value iteration Policy evaluated by Bellman-residual minimization (BRM)
    - Policy evaluated by least-squares temporal difference learning (LSTD) => LSPI
- Policy search

Evaluating a policy: Fitted value iteration

- Choose a function space *F*.
- Solve for *i=1,2,…,M* the LS (regression) problems:

$$Q_{i+1} = \mathrm{argmin}_{Q \in F} \sum_{t=1}^{T} (R_t + \gamma Q_i(X_{t+1}, \pi(X_{t+1})) - Q(X_t, A_t))^2$$

- Counterexamples?!?!?
  [Baird '95, Tsitsiklis and van Roy '96]
- When does this work??
- Requirement: If *M* is big enough and the number of samples is big enough Q_M should be close to Q^{π}
- We have to make some assumptions on *F*

Least-squares vs. gradient

- Linear least squares (ordinary regression):
  $$y_t = w_*^T x_t + \epsilon_t$$
  (x_t, y_t) jointly distributed r.v.s., iid, $E[\epsilon_t | x_t] = 0$.
- Seeing (x_t, y_t), $t=1,\ldots,T$, find out w_*.
- Loss function: $L(w) = E[\, (y_1 - w^T x_1\,)^2\,]$.
- Least-squares approach:
  - $w_T = \mathrm{argmin}_w \sum_{t=1}^T (y_t - w^T x_t)^2$
- Stochastic gradient method:
  - $w_{t+1} = w_t + \alpha_t\,(y_t - w_t^T x_t)\, x_t$
- Tradeoffs
  - Sample complexity: How good is the estimate
  - Computational complexity: How expensive is the computation?

Fitted value iteration: Analysis

- *Goal:* Bound $||Q_M - Q_\pi||_\mu^2$ in terms of
  $\max_m ||\epsilon_m||_\nu^2$, $||\epsilon_m||_\nu^2 = \int \epsilon_m^2(x,a)\, \nu(dx,da)$,
  where $Q_{m+1} = T_\pi Q_m + \epsilon_m$, $\epsilon_{-1} = Q_0 - Q_\pi$
- $U_m = Q_m - Q_\pi$

$$\begin{aligned} U_{m+1} &= Q_{m+1} - Q_\pi \\ &= T^\pi Q_m - Q_\pi + \epsilon_m \\ &= T^\pi Q_m - T^\pi Q_\pi + \epsilon_m \\ &= \gamma P_\pi U_m + \epsilon_m. \end{aligned}$$

$$U_M = \sum_{m=0}^{M} (\gamma P_\pi)^{M-m} \epsilon_{m-1}.$$

Analysis/2

Jensen applied to operators, $\mu \le C_1 \nu$ and: $\forall \rho: \ \rho P_\pi \le C_1 \nu$

Legend:

- $\rho f = \int f(x) \rho(dx)$
- $(Pf)(x) = \int f(y) P(dy|x)$

$$
\begin{aligned}
U_M &= \sum_{m=0}^{M} (\gamma P_\pi)^{M-m} \epsilon_{m-1}. \\
\mu |U_M|^2 &\le \left(\frac{1}{1-\gamma}\right)^2 \frac{1-\gamma}{1-\gamma^{M+1}} \sum_{m=0}^{M} \gamma^m \mu \left((P_\pi)^m \epsilon_{M-m-1} \right)^2 \\
&\le C_1 \left(\frac{1}{1-\gamma}\right)^2 \frac{1-\gamma}{1-\gamma^{M+1}} \sum_{m=0}^{M} \gamma^m \nu |\epsilon_{M-m-1}|^2 \\
&\le C_1 \left(\frac{1}{1-\gamma}\right)^2 \frac{1-\gamma}{1-\gamma^{M+1}} \left(\gamma^M \nu |\epsilon_{-1}|^2 + \sum_{m=0}^{M} \gamma^m \epsilon^2 \right) \\
&= C_1 \left(\frac{1}{1-\gamma}\right)^2 \epsilon^2 + C_1 \frac{\gamma^M \nu |\epsilon_{-1}|^2}{1-\gamma^{M+1}}.
\end{aligned}
$$

Jensen

Summary

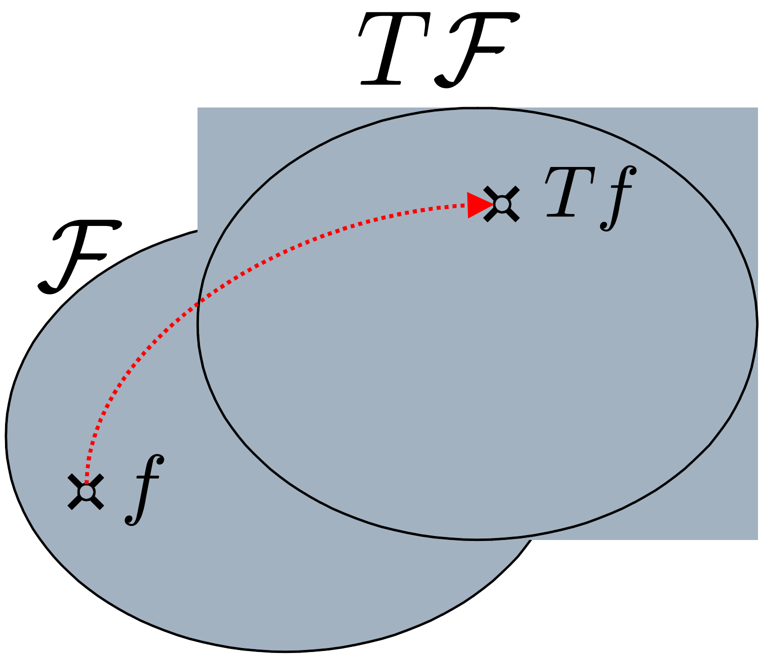
- If the regression errors are all small and the system is noisy ($\forall\ \pi,\rho,\ \rho\, P^{\pi} \leq C_1\, \nu$) then the final error will be small.
- How to make the regression errors small?
- Regression error decomposition:

Estimation error

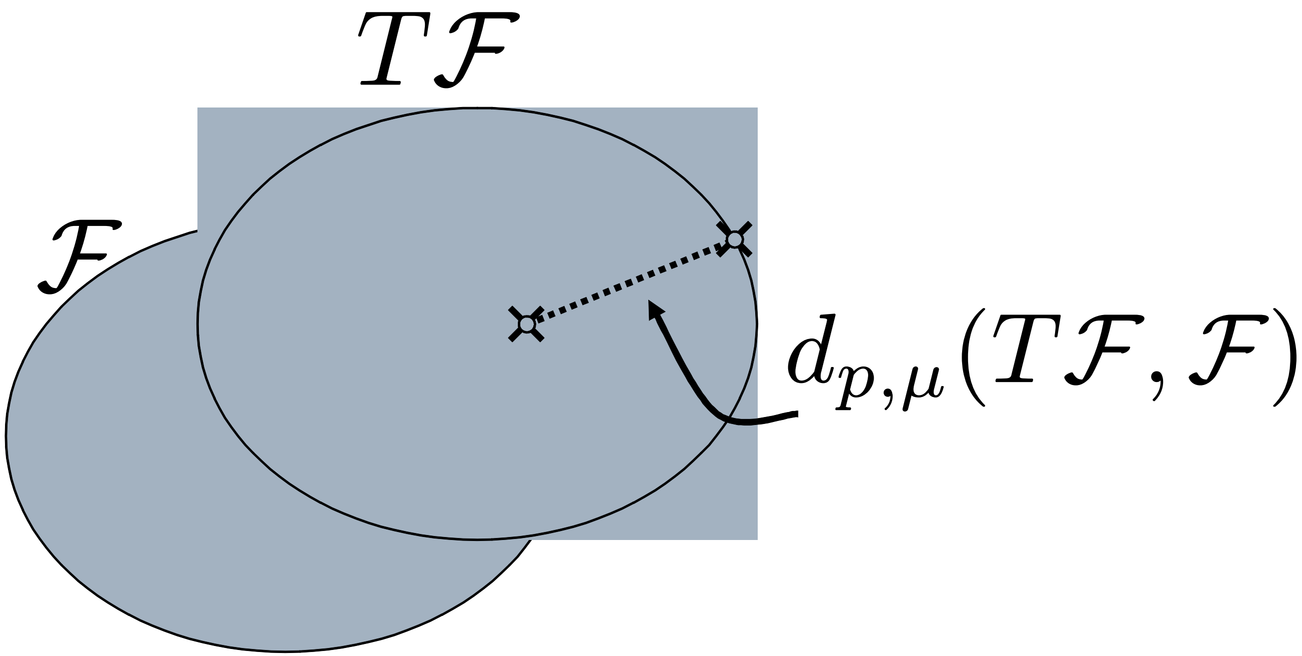
$$\begin{aligned} \|Q_{m+1} - T_\pi Q_m\|_2 \quad &\leq \quad \|Q_{m+1} - \Pi_F T_\pi Q_m\|_2 \\ &\quad + \|\Pi_F T_\pi Q_m - T_\pi Q_m\|_2 \end{aligned}$$

Approximation error

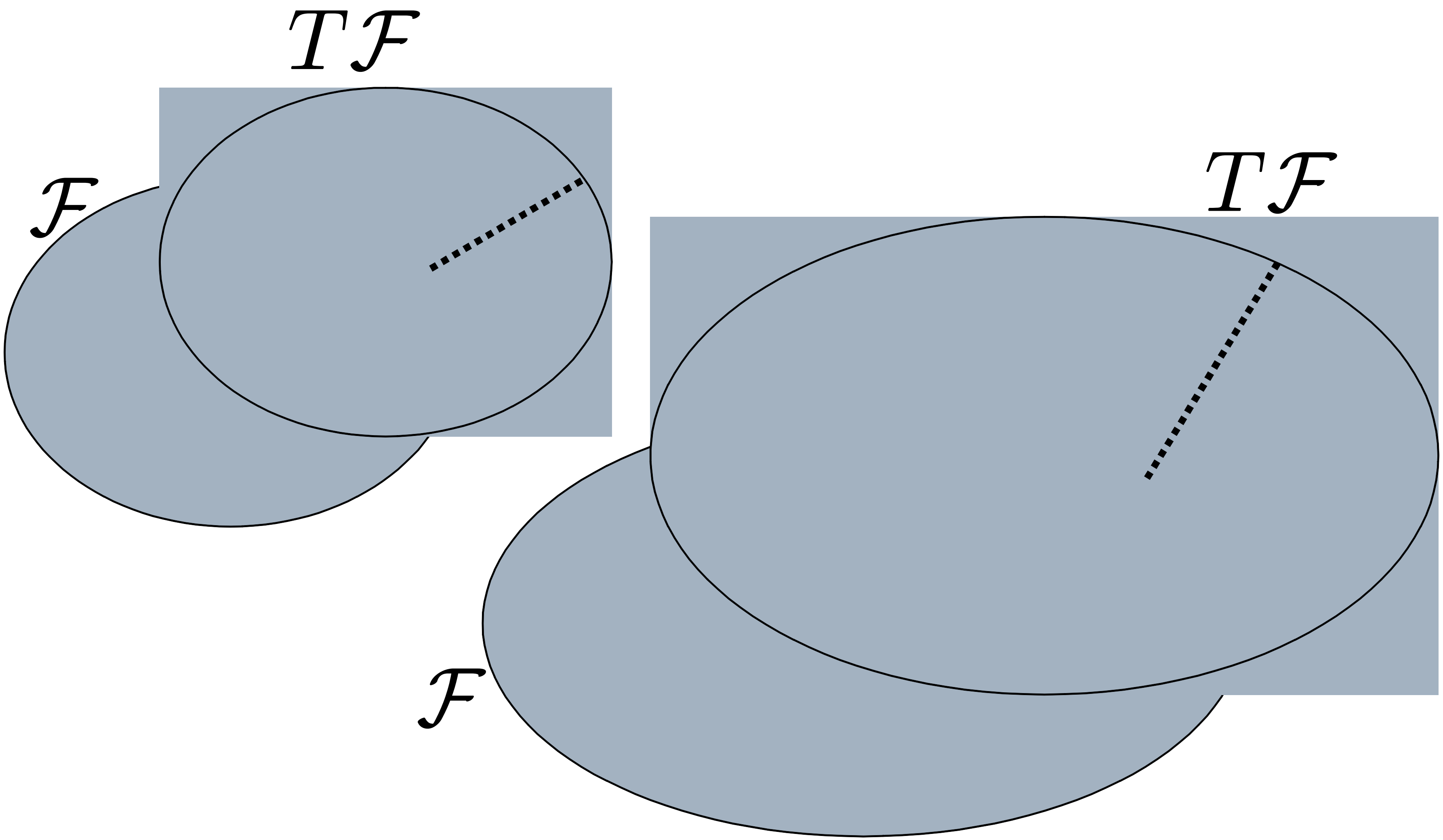
Controlling the approximation error



Controlling the approximation error

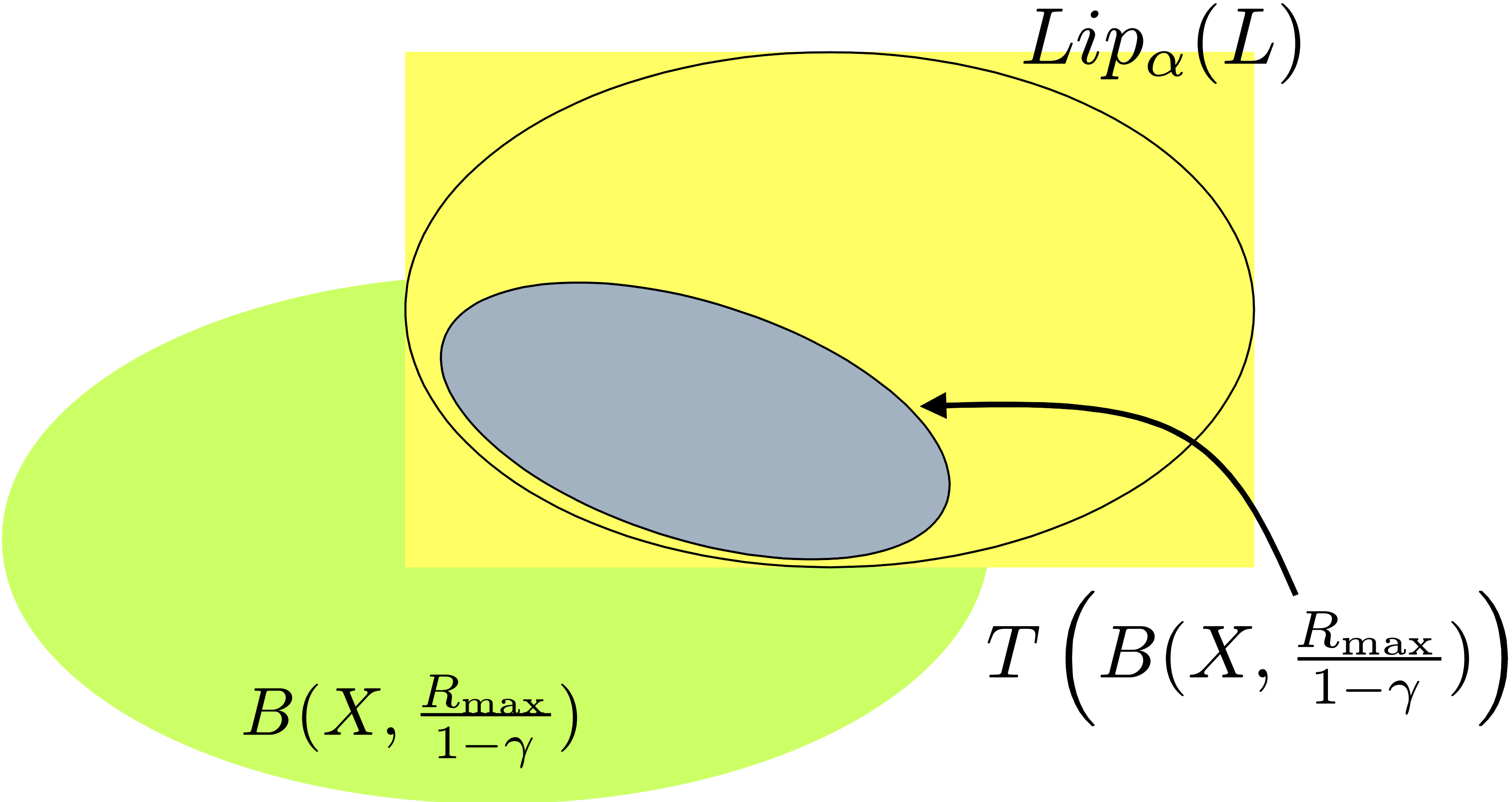


Controlling the approximation error



Controlling the approximation error

- Assume smoothness!



Learning with (lots of) historical data

- Data: A long trajectory of some exploration policy
- Goal: Efficient algorithm to learn a policy
- Idea: Use fitted action-values
- Algorithms:
  - Bellman residual minimization, FQI [AnSzeMu '07]
  - LSPI [Lagoudakis, Parr '03]
- Bounds:
  - Oracle inequalities (BRM, FQI and LSPI)
  - $\Rightarrow$ consistency

[AnSzeMu '07]

BRM insight

- TD error: $\Delta_t = R_t + \gamma\, Q(X_{t+1}, \pi(X_{t+1})) - Q(X_t, A_t)$
- Bellman error: $E[E[\, \Delta_t \mid X_t, A_t \,]^2]$
- What we can compute/estimate: $E[E[\, \Delta_t^2 \mid X_t, A_t]]$
- They are different!
- However:

$$\mathbb{E}[\Delta_t | X_t, A_t]^2 = \mathbb{E}[\Delta_t^2 | X_t, A_t] - \mathrm{Var}[\Delta_t | X_t, A_t]$$

$$\mathbb{E}[\Delta_t|X_t, A_t]^2 = \mathbb{E}[\Delta_t^2|X_t, A_t] - \mathrm{Var}[\Delta_t|X_t, A_t]$$

Loss function

$$L_{N,\pi}(Q,h) =$$

$$\frac{1}{N}\sum_{t=1}^{N} w_t \Big\{ (R_t + \gamma Q(X_{t+1}, \pi(X_{t+1})) - Q(X_t, A_t))^2$$

$$- (R_t + \gamma Q(X_{t+1}, \pi(X_{t+1})) - h(X_t, A_t))^2 \Big\}$$

$$w_t = 1/\mu(A_t|X_t)$$

Algorithm (BRM++)

1. Choose π_0, $i := 0$
2. While $(i \leq K)$ do:
3. Let $Q_{i+1} = \text{argmin}_{Q \in \mathcal{F}^{\mathcal{A}}} \sup_{h \in \mathcal{F}^{\mathcal{A}}} L_{N,\pi_i}(Q, h)$
4. Let $\pi_{i+1}(x) = \text{argmax}_{a \in \mathcal{A}} Q_{i+1}(x, a)$
5. $i := i + 1$

Do we need to reweight or throw away data?

- NO!
- WHY?
- Intuition from regression:
  - $m(x) = E[Y|X=x]$ can be learnt no matter what $p(x)$ is!
  - $\pi^*(a|x)$: the same should be possible!
- BUT..
  - Performance might be poor! => YES!
  - Like in supervised learning when training and test distributions are different

Bound

$$\|Q^* - Q_{\pi_K}\|_{2,\rho} \leq$$
$$\frac{2\gamma}{(1-\gamma)^2} C_{\rho,\nu}^{1/2} \left(\tilde{E}(\mathcal{F}) + E(\mathcal{F}) + S_{N,x}^{1/2} \right) + (2\gamma^K)^{1/2} R_{\max},$$

$$S_{N,x} = c_2 \frac{\left((\frac{V}{2}+1)\ln(N) + \ln(c_1) + \frac{1}{1+\kappa}\ln(\frac{bc_2^2}{4}) + x \right)^{\frac{1+\kappa}{2\kappa}}}{(b^{1/\kappa} N)^{1/2}}$$

The concentration coefficients

- Lyapunov exponents

$$y_{t+1} = P_t y_t$$
$$\hat{\gamma}_{\text{top}} = \limsup_{t\to\infty} \frac{1}{t} \log^+(\|y_t\|_\infty)$$

- Our case:
  - y_t is infinite dimensional
  - P_t depends on the policy chosen
  - If top-Lyap exp. ≤ 0, we are good☺

Open question

- Abstraction:

$$f(i_1, \ldots, i_m) = \log(||P_{i_1} P_{i_2} \ldots P_{i_m}||),\ i_k \in \{0, 1\}.$$

- Let

$$f : \{0,1\}^* \to \mathbb{R}^+,\ f(x+y) \le f(x) + f(y),$$
$$\limsup_{m\to\infty} \frac{1}{m} f([x]_m) \le \beta.$$

- True?

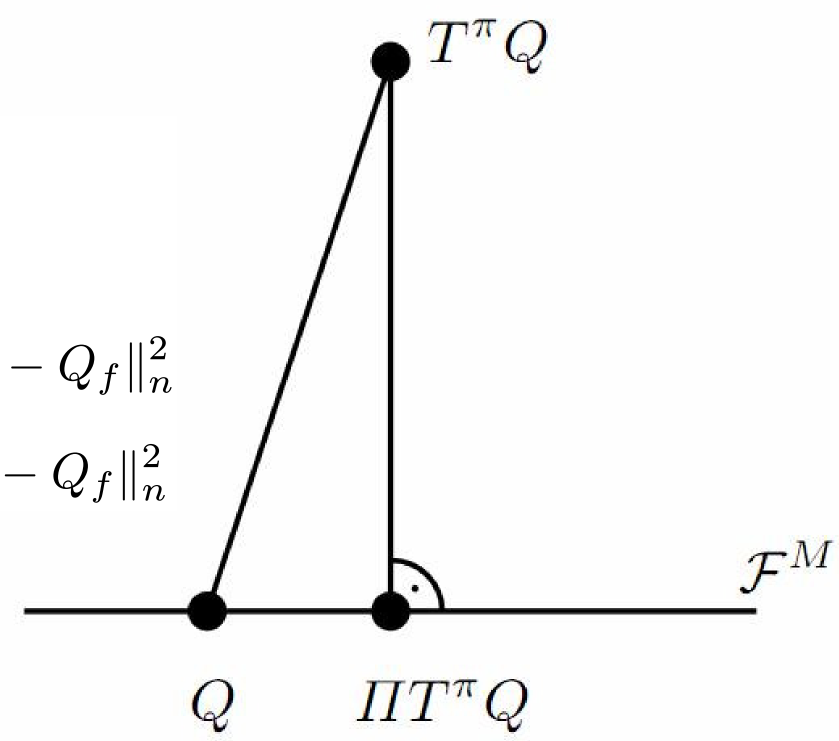
$$\forall \{y_m\}_m,\ y_m \in \{0,1\}^m,$$

$$\limsup_{m\to\infty} \frac{1}{m} \log f(y_m) \le \beta$$

Relation to LSTD

- LSTD:
  - Linear function space
  - Bootstrap the "normal equation"

$$
\begin{aligned}
h^*(f) &= \inf_{h \in \mathcal{F}} \|h - Q_f\|_n^2 \\
Q_{\mathrm{LSTD}} &= \inf_{f \in \mathcal{F}} \|f - h^*(f)\|_n^2 \\
Q_{\mathrm{BRM}} &= \inf_{f \in \mathcal{F}} \|f - Q_f\|_n^2 - \|h^*(f) - Q_f\|_n^2 \\
\|Q - Q_f\|_n^2 &= \|Q - h^*(Q)\|_n^2 + \|h^*(Q) - Q_f\|_n^2
\end{aligned}
$$



Open issues

- Adaptive algorithms to take advantage of regularity when present to address the "curse of dimensionality"
  - Penalized least-squares/aggregation?
  - Feature relevance
  - Factorization
  - Manifold estimation
- Abstraction – build automatically
- Active learning
- Optimal on-line learning for infinite problems

References

- [Auer et al. ’02] P. Auer, N. Cesa-Bianchi and P. Fischer: Finite time analysis of the multiarmed bandit problem, *Machine Learning*, 47:235—256, 2002.
- [AuSzeMu ’07] J.-Y. Audibert, R. Munos and Cs. Szepesvári: Tuning bandit algorithms in stochastic environments, ALT, 2007.
- [Auer, Jaksch & Ortner ’07] P. Auer, T. Jaksch and R. Ortner: Near-optimal Regret Bounds for Reinforcement Learning, (2007), available at http://www.unileoben.ac.at/~infotech/publications/ucrlrevised.pdf
- [Singh & Sutton ’96] S.P. Singh and R.S. Sutton: Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123—158, 1996.
- [Sutton ’88] R.S. Sutton: Learning to predict by the method of temporal differences. *Machine Learning*, 3:9—44, 1988.
- [Jaakkola et al. ’94] T. Jaakkola, M.I. Jordan, and S.P. Singh: On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6: 1185—1201, 1994.
- [Tsitsiklis, ’94] J.N. Tsitsiklis: Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185—202, 1994.
- [SzeLi99] Cs. Szepesvári and M.L. Littman: A Unified Analysis of Value-Function-Based Reinforcement-Learning Algorithms, *Neural Computation*, 11, 2017—2059, 1999.
- [Watkins ’90] C.J.C.H. Watkins: *Learning from Delayed Rewards*, PhD Thesis, 1990.
- [Rummery and Niranjan ’94] G.A. Rummery and M. Niranjan: On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
- [Sutton ’84] R.S. Sutton: *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 1984.
- [Tsitsiklis & Van Roy ’97] J.N. Tsitsiklis and B. Van Roy: An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674—690, 1997.
- [Sutton ’96] R.S. Sutton: Generalization in reinforcement learning: Successful examples using sparse coarse coding. NIPS, 1996.
- [Baird ’95] L.C. Baird: Residual algorithms: Reinforcement learning with function approximation, ICML, 1995.
- [Bradtke, Barto ’96] S.J. Bradtke and A.G. Barto: Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33—57, 1996.
- [Lagoudakis, Parr ’03] M. Lagoudakis and R. Parr: Least-squares policy iteration, *Journal of Machine Learning Research*, 4:1107—1149, 2003.
- [AnSzeMu ’07] A. Antos, Cs. Szepesvari and R. Munos: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path, Machine Learning Journal, 2007.