More Reinforcement Learning

CMPSCI 383
Dec 1, 2011

Today's lecture

- Quick review of passive RL
  - More on TD learning
  - TD learning of action values
- Active RL
  - Key fact
  - Exploration
  - SARSA
  - Q-Learning
  - TD and Dopamine
  - Generalization: function approximation
  - Eligibility traces

Check this out

MIT's Technology Review has an in-depth interview with Peter Norvig, Google'sDirector of Research, and Eric Horvitz, a Distinguished Scientist at MicrosoftResearch about their optimism for the futureof AI.

http://www.technologyreview.com/computing/39156/

Passive RL: review

- Agent has fixed policy and learns utilities
  - Barto: “not really RL: it is prediction”
- Direct utility estimation
  - Collect samples of quantity to be estimated
  - Average them
  - Or use an incremental method….
  - Does not take advantage of relationship between utilities of different states
- Adaptive Dynamic Programming (ADP)
  - Learn a model
  - Do DP on it
  - Can interleave these (modified policy iteration)
- Temporal Difference (TD) Learning
  - Use an error to make estimates adhere to constraint
  - Does not need a model

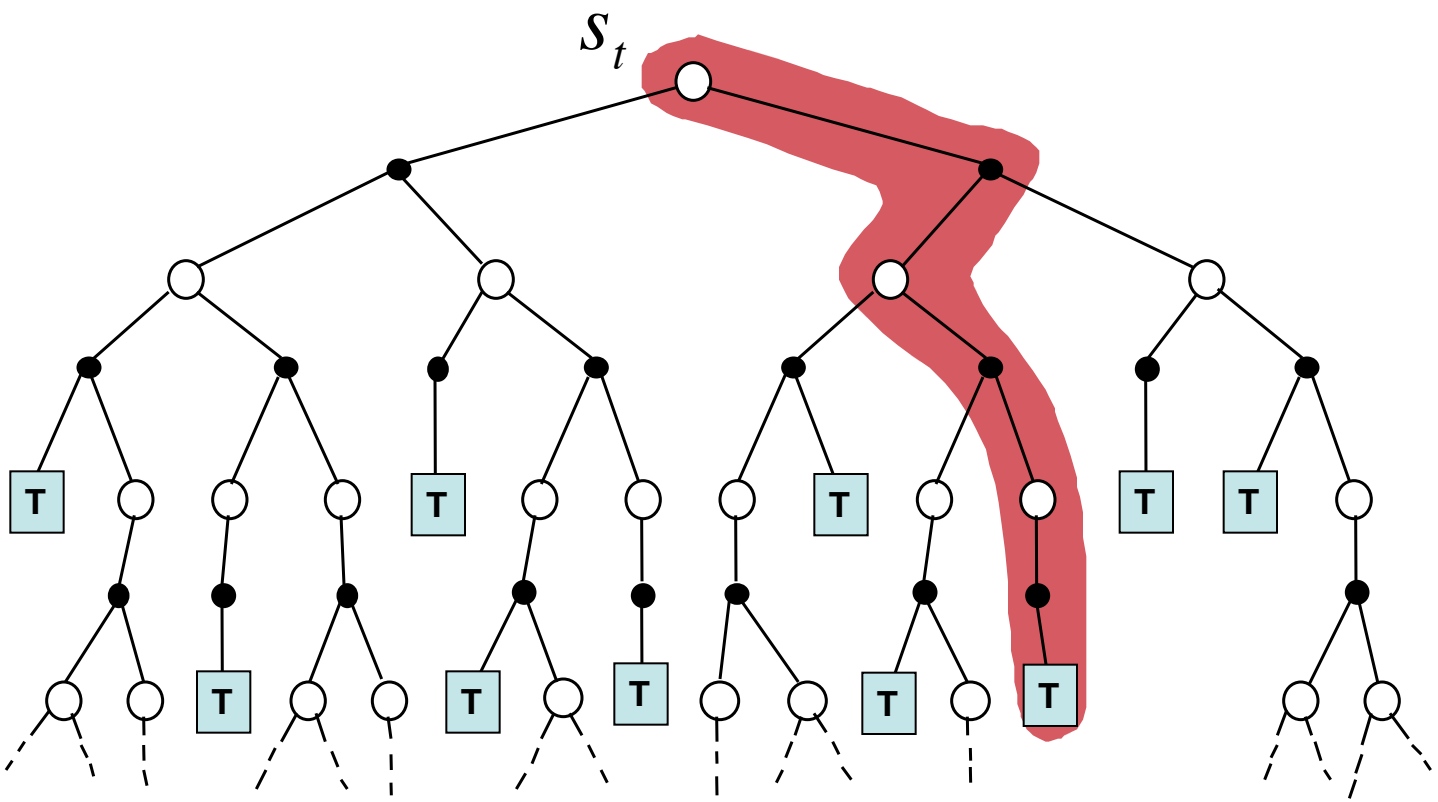
A Bit of Terminology

- Utilities (U) = Values (V)
- Return: discounted sum of rewards
  - Return from a state: the discounted sum of rewards accumulated after visiting that state
  - Same as “reward-to-go”
  - Utility (or value) of a state is the expected return from that state

Direct Utility Estimation (incremental)

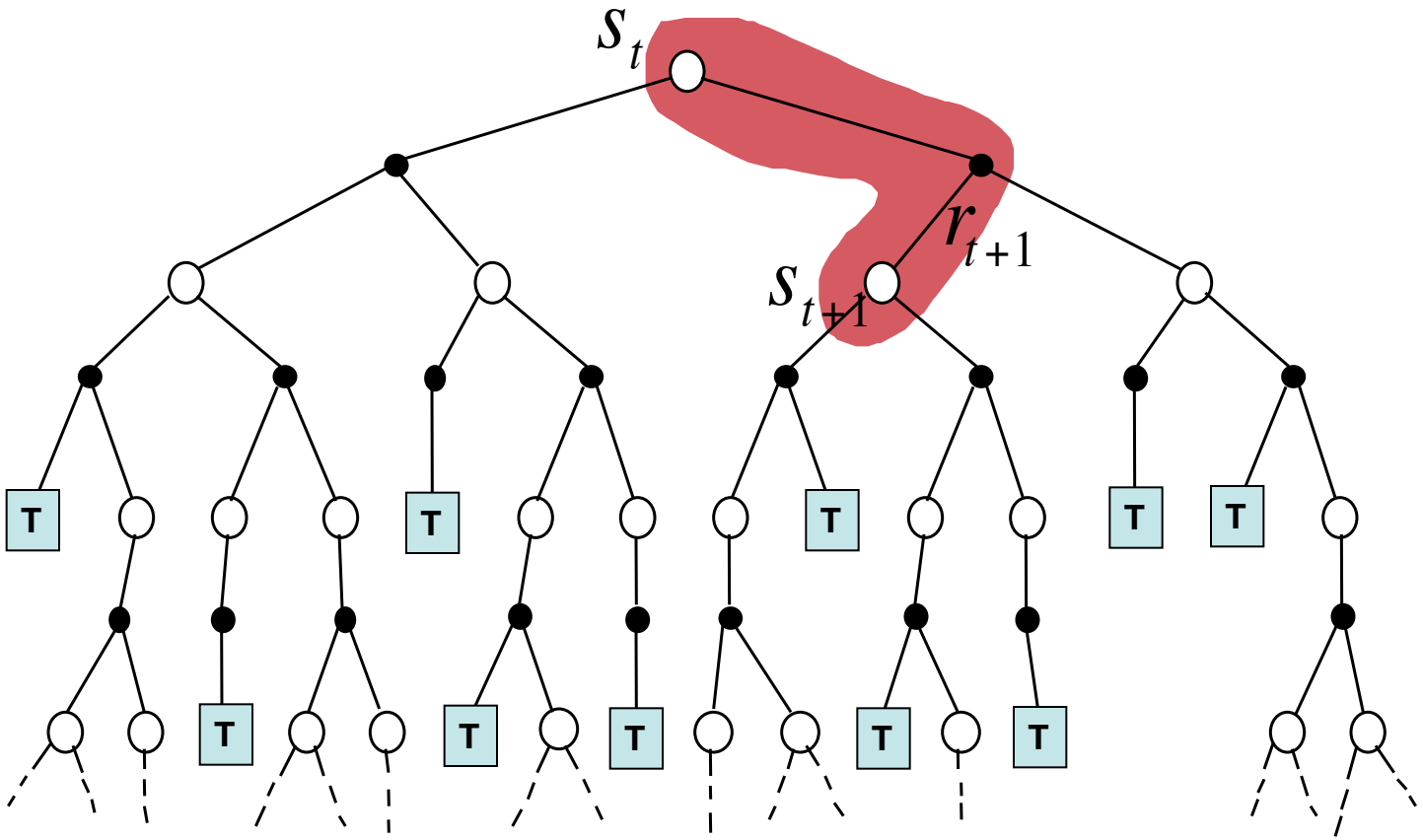
$$V(s_t) \leftarrow V(s_t) + \alpha\left[R_t - V(s_t)\right]$$

where R_t is the actual return following state s_t.



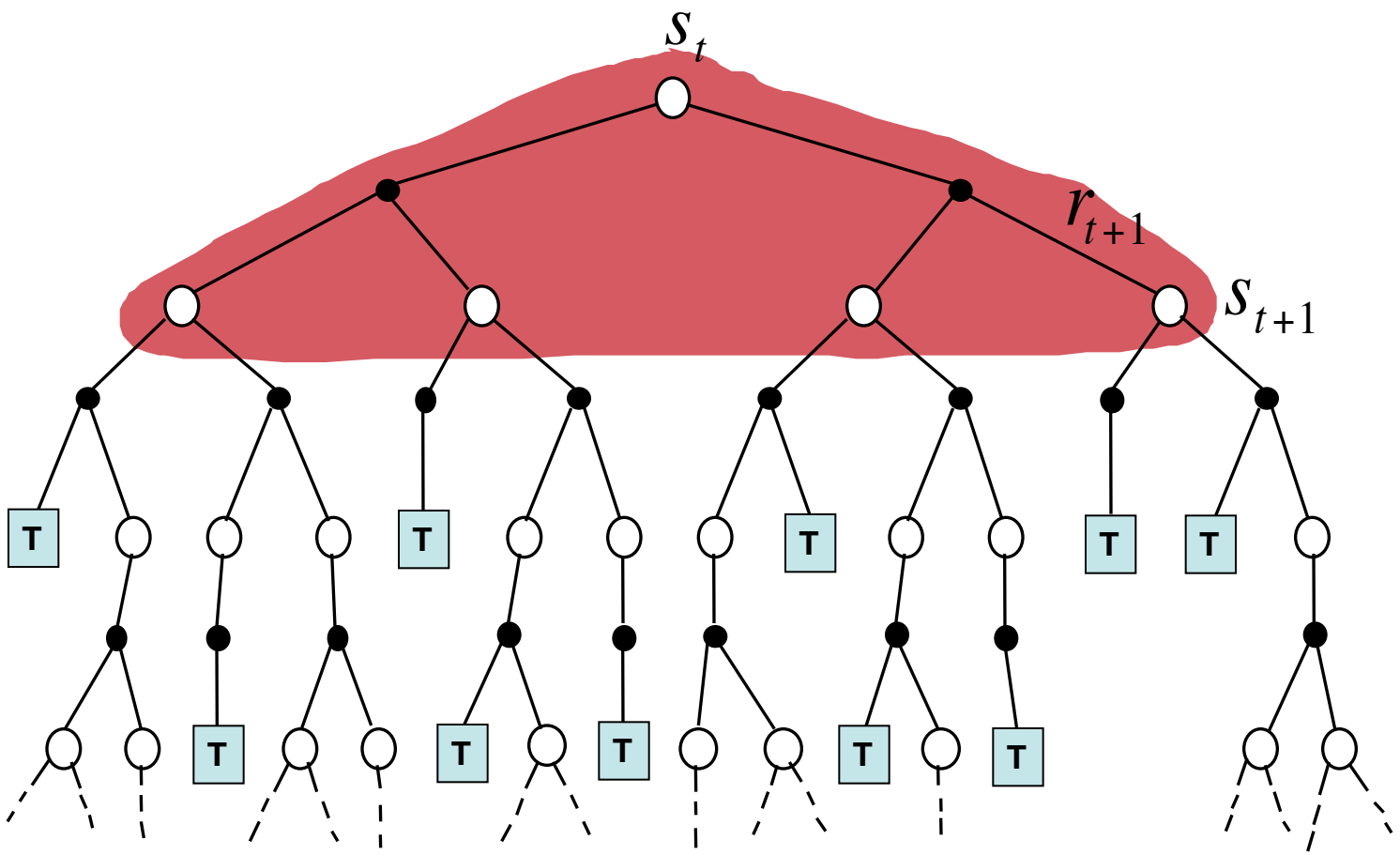
Simplest TD Method

$$V(s_t) \leftarrow V(s_t) + \alpha\left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right]$$



cf. Dynamic Programming

$$V(s_t) \leftarrow E_\pi \left\{ r_{t+1} + \gamma \, V(s_t) \right\}$$



More on TD Learning

- TD methods do not require a model of the environment, only experience
- You can learn before knowing the final outcome
  - Less memory
  - Less peak computation
  - You can learn without the final outcome from incomplete sequences

You are the Predictor

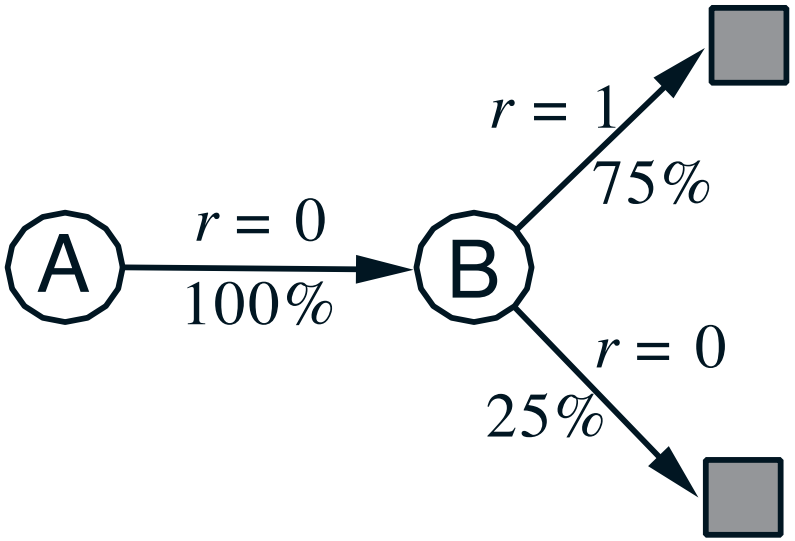
Suppose you observe the following 8 episodes:

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

$V(\mathrm{A})?$

$V(\mathrm{B})?$

You are the Predictor



$V(\mathrm{A})?$

You are the Predictor

- The prediction that best matches the training data is V(A)=0
  - This minimizes the mean-square-error on the training set
  - This is what direct utility estimation gets
- If we consider the sequentiality of the problem, then we would set V(A)=.75
  - This is correct for the maximum likelihood estimate of a Markov model generating the data
  - i.e, if we do a best fit Markov model, and assume it is exactly correct, and then compute what it predicts (how?)
  - This is called the certainty-equivalence estimate
  - This is what TD gets

12

TD learning of Action-Values (for a given policy)

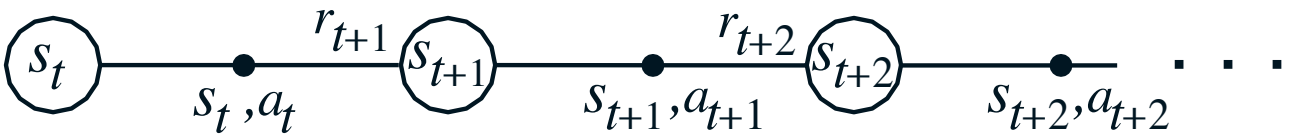
$Q^{\pi}(s,a) =$ Utility of doing action a in state s
i.e.: Total amount of reward expected over the future if you do action *a* in state s and thereafter follow policy π

The utility of a state is the utility of doing the best action from that state:

$$U^{\pi}(s) = \max_{a} Q^{\pi}(s,a)$$

TD Learning of Action-Values (for a given policy)

Estimate Q^{π} for the current behavior policy π.



After every transition from a nonterminal state s_t, do this:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

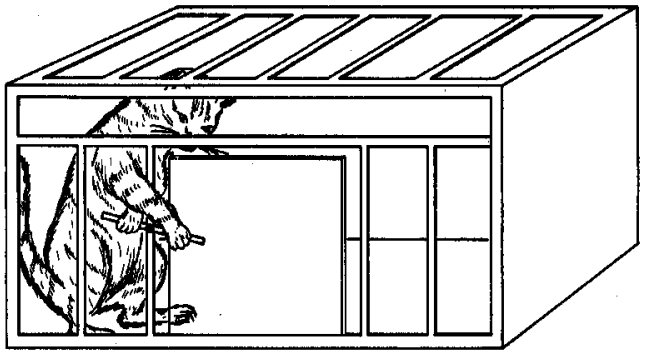
If s_{t+1} is terminal, then $Q(s_{t+1}, a_{t+1}) = 0$.

Active RL

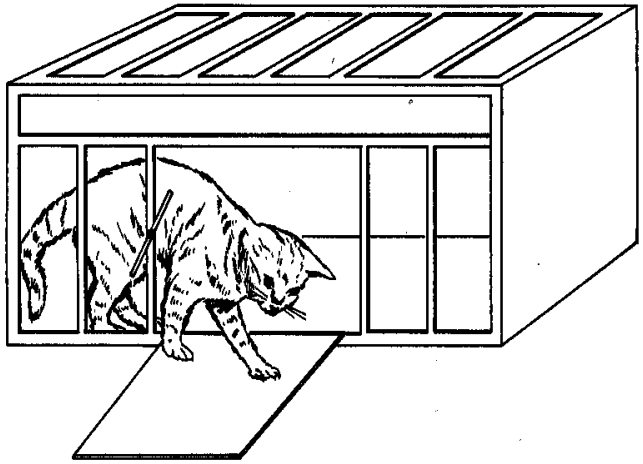
- Passive agent follows a fixed policy, estimates expected utilities
- Active agent needs to decide on what actions to perform to maximize expected utility

Passive agent: faces a **prediction** problem
Active agent: faces a **control** problem

Edward L. Thorndike (1874-1949)



(a)



Learning by “Trial-and-Error”

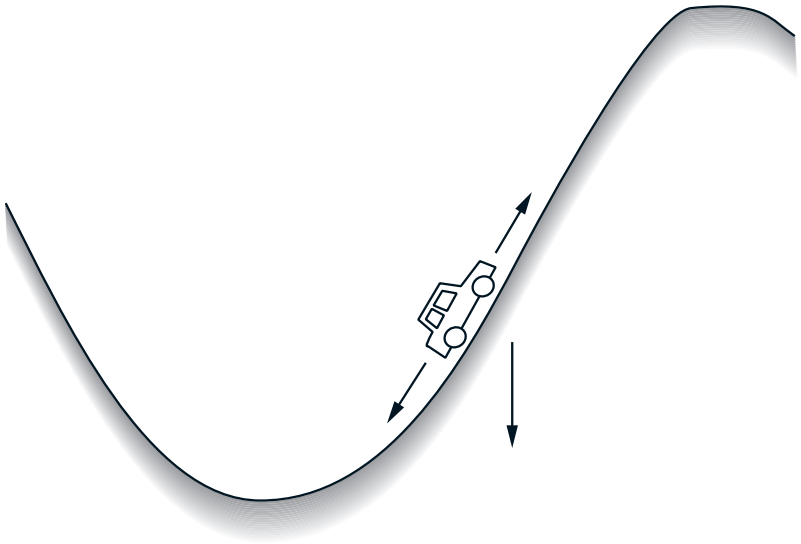
puzzle box

Key Fact

- A greedy agent is one that always takes the action that maximizes its current utility estimates

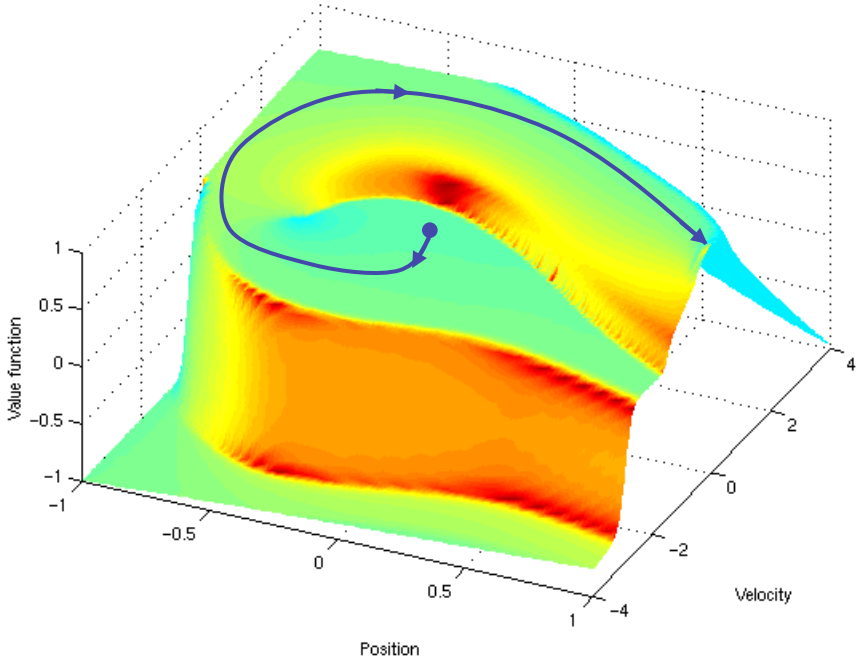
- If its utility estimates are correct, i.e., it has learned the true utility function (or optimal value function), then a greedy agent acts optimally.

Value-Guided Optimal Control



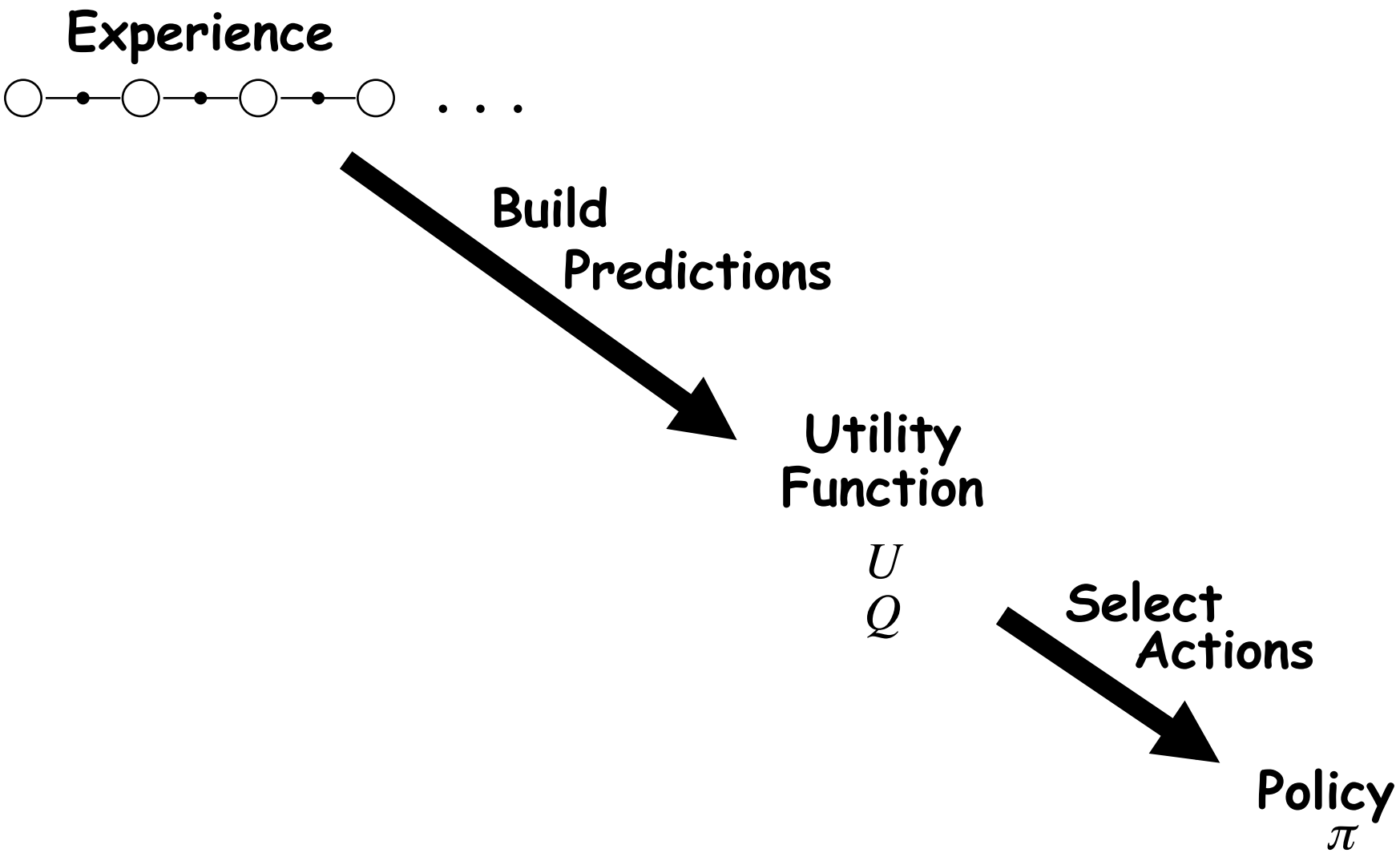
Get to the top of the hill as quickly as possible (roughly)

Predicted minimum time to goal (negated)

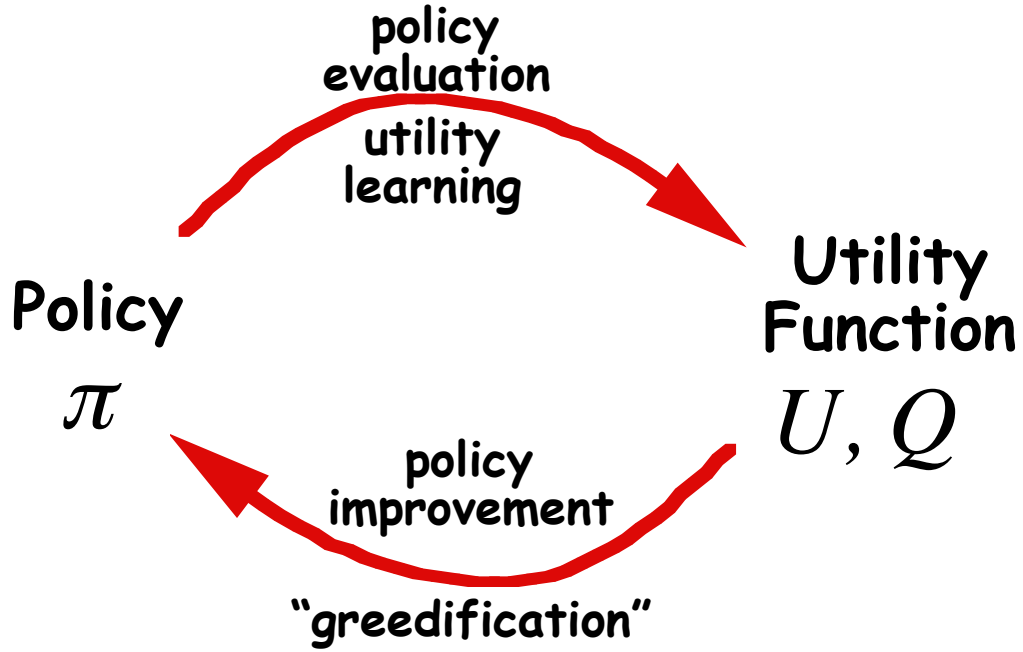


Munos & Moore "Variable resolution discretization for high-accuracy solutions of optimal control problems", IJCAI 99.

Active RL Agents



Interaction of policy and utility



Exploration

A greedy agent very rarely learns to act optimally

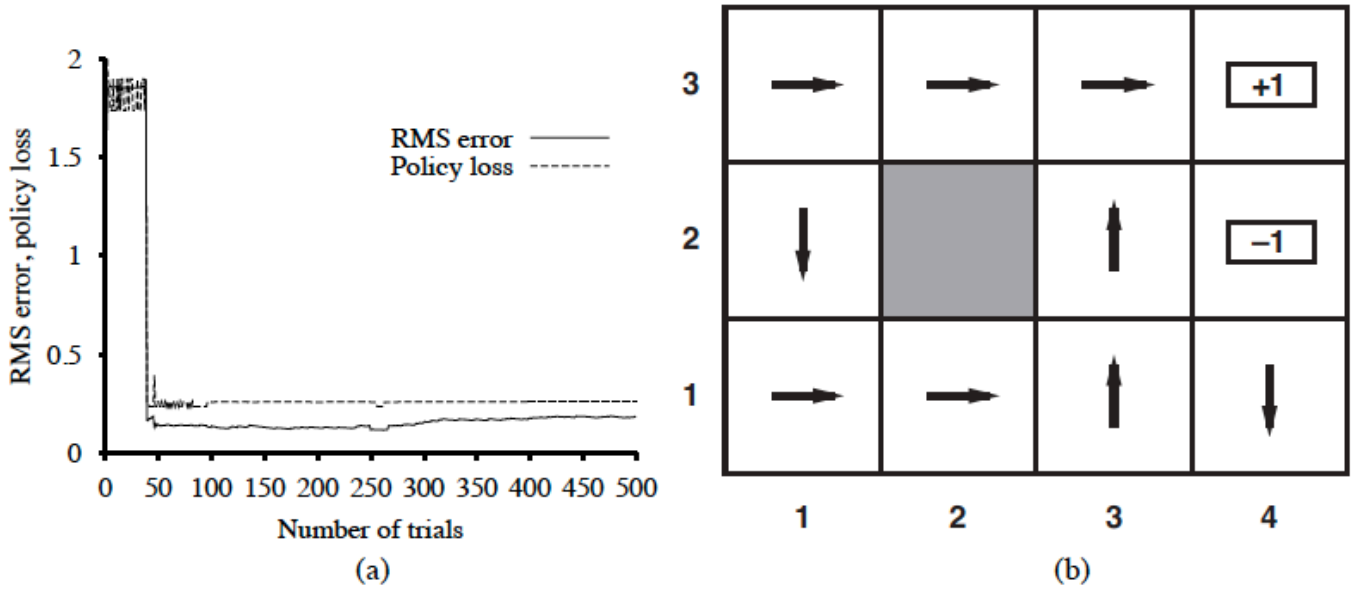


Figure 21.6 FILES: figures/4x3-greedy-adp-policy.eps (Tue Nov 3 16:22:10 2009). Performance of a greedy ADP agent that executes the action recommended by the optimal policy for the learned model. (a) RMS error in the utility estimates averaged over the nine nonterminal squares. (b) The suboptimal policy to which the greedy agent converges in this particular sequence of trials.

Exploration/Exploitation Dilemma

- Exploitation: act according to your current estimates (exploit current “knowledge”).
- Exploration: do something else!

- You can’t do both at the same time.
- How do you handle the tradeoff?

What's the best exploration policy?

Assume you've learned a utility function,
How do you select actions?

Current estimate

Greedy Action Selection:

Always select the action that looks best:

$$\pi(s) = \arg\max_a Q(s,a)$$

ε-Greedy Action Selection:

Be greedy most of the time

Occasionally take a random action

The simplest possible thing!

Other Methods:

Boltzmann distribution, keep track of confidence intervals, etc.

ε-Greedy Action Selection

- Greedy action selection:

$$a_t = a_t^* = \arg\max_a Q_t(a)$$

- ε-Greedy:

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

. . . the simplest way to try to balance exploration and exploitation

More on Exploration

- GLIE schemes: “Greedy in the Limit of Infinite Exploration”
  - Simplest maybe: ε-Greedy with decreasing ε
  - Optimistic initial estimates, fading out with increasing visitations
- “Exploration Bonuses”
- Approximations to optimal exploration…see box on p. 841

Sarsa

Turn passive learning of action values into an active method by always updating the policy to be greedy with respect to the current estimate:

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right]$
    $s \leftarrow s'; a \leftarrow a';$
  until s is terminal

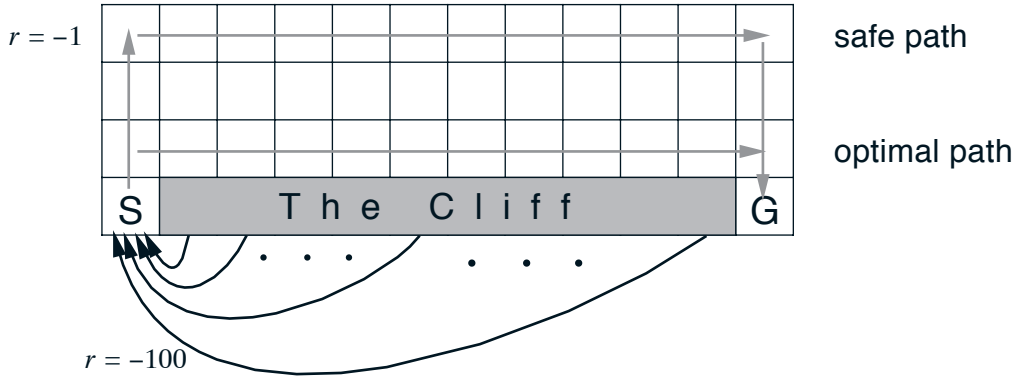
Q-Learning

One - step Q - learning :

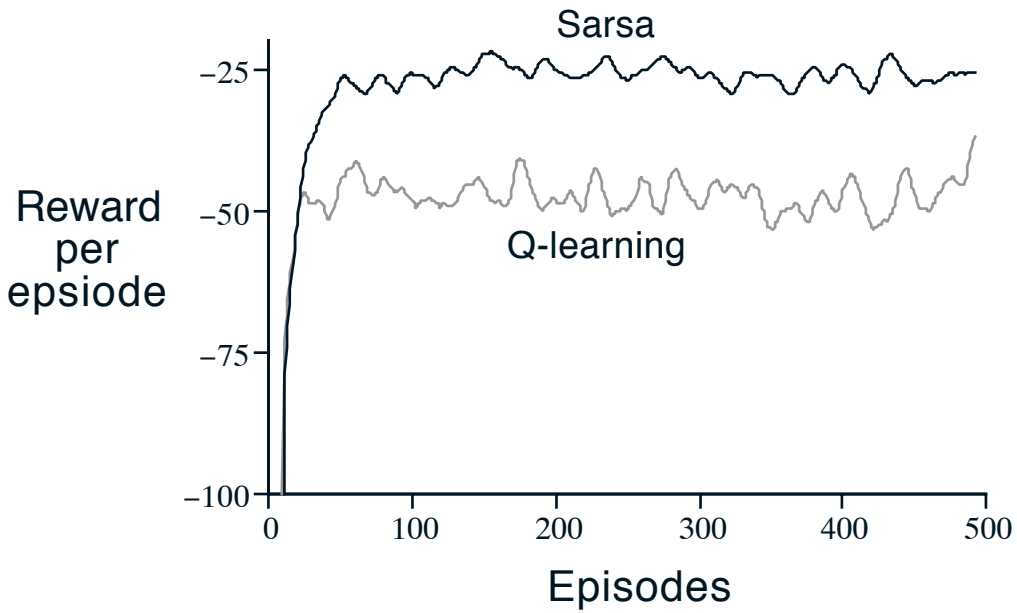
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        Choose a from s using policy derived from Q (e.g., ϵ-greedy)
        Take action a, observe r, s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
        $s \leftarrow s'$;
    until s is terminal

Cliffwalking



ε-greedy, ε = 0.1



On-Policy vs. Off-Policy

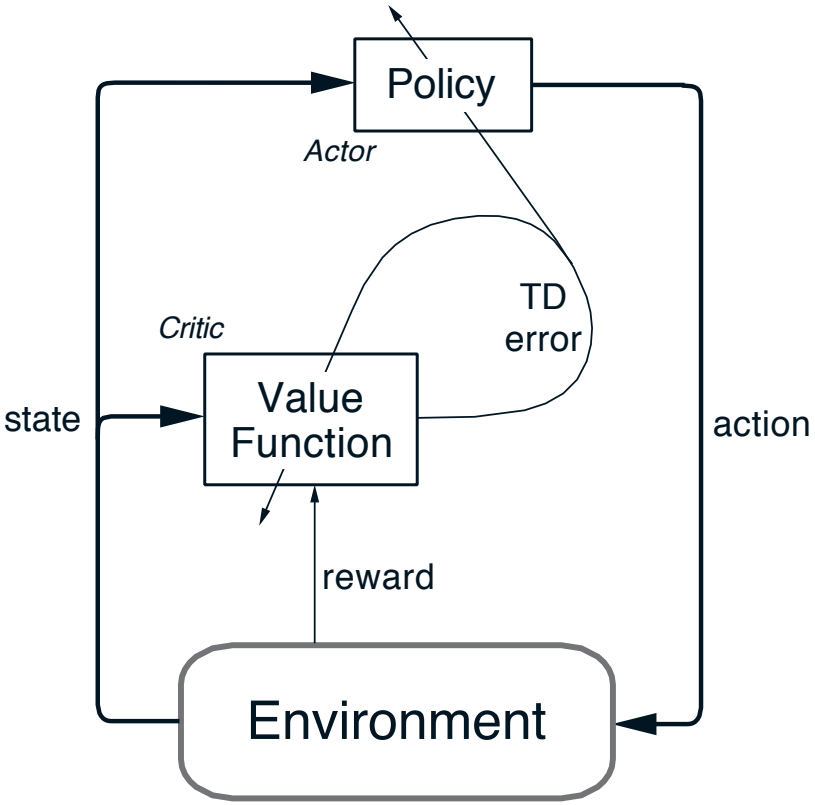
- Behavior policy: the policy the agent is using.
- Estimation policy: the policy the agent is evaluating

SARSA

- On-Policy methods:
  - Estimation policy = Behavior policy
- Off-Policy methods:
  - Estimation policy ≠ Behavior policy

Q-learning

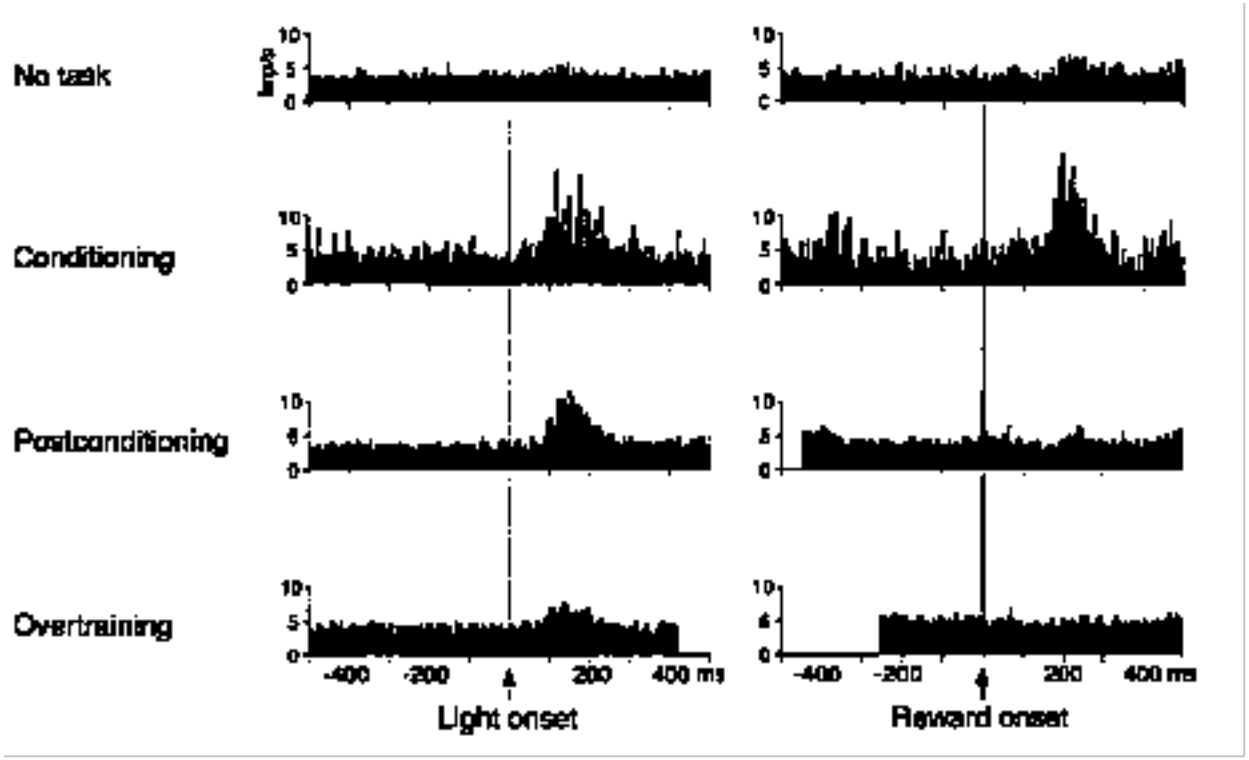
Actor-Critic Methods



- Explicit representation of policy as well as value function
- Minimal computation to select actions
- Can learn an explicit stochastic policy
- Can put constraints on policies
- Appealing as psychological and neural models

Dopamine and TD Error

W. Schultz et al.
Universite de Fribourg



Generalization in RL

- So far only considered lookup table representations of utility functions.
- What if the state set is huge? e.g. Backgammon
- Use function approximation methods

Features or Basis Functions

- E.g., linear function approximation: represent U or Q as a linear combination of features (or basis functions $f_1, \ldots f_n$:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s)$$

$n <<$ number of states

Gradient ascent TD learning

For state-value functions:

$$\theta_i \leftarrow \theta_i + \alpha\left[R(s) + \gamma\,\hat{U}_\theta(s') - \hat{U}_\theta(s)\right]\frac{\partial \hat{U}_\theta(s)}{\partial \theta_i}$$

For action-value functions:

$$\theta_i \leftarrow \theta_i + \alpha\left[R(s) + \gamma \max_{a'} \hat{Q}_\theta(s',a') - \hat{Q}_\theta(s,a)\right]\frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i}$$

For Linear Function Approximation

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s)$$

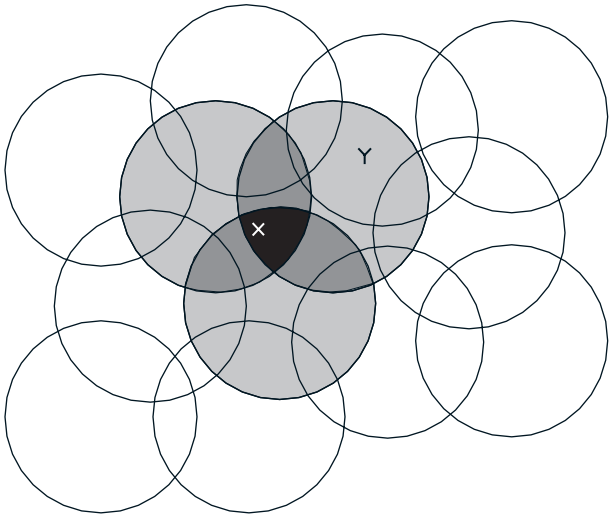
$$\frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} = ?$$

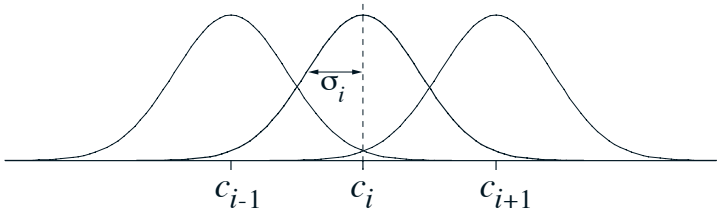
For Linear Function Approximation

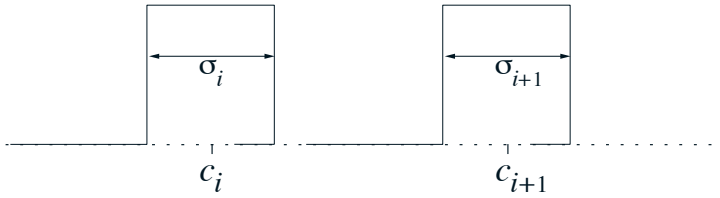
$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s)$$

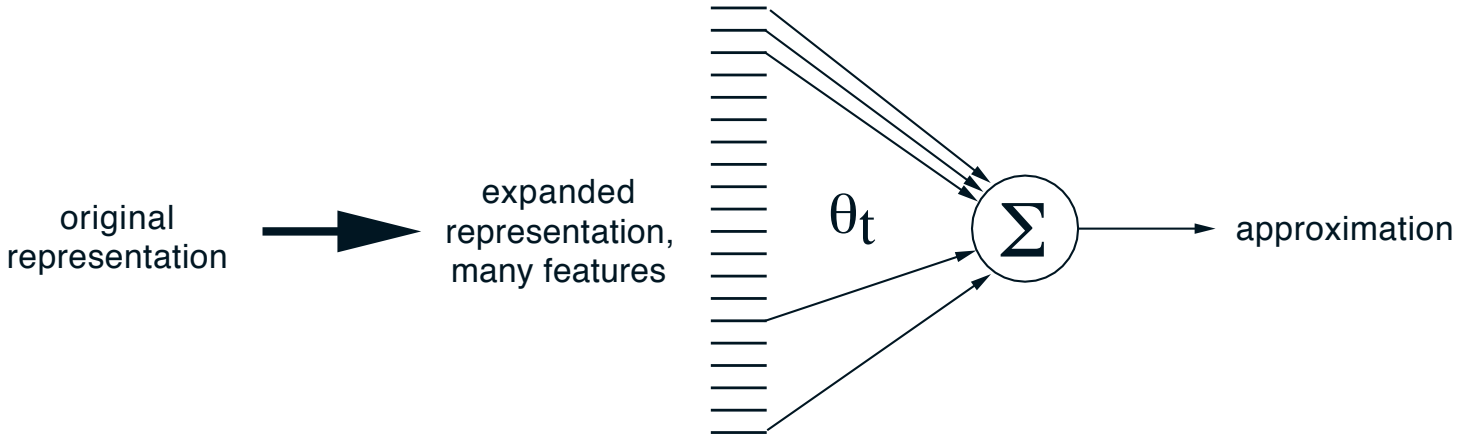
$$\frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} = f_i(s)$$

“Coarse Coding”

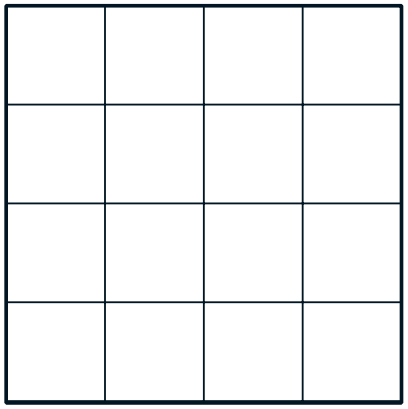




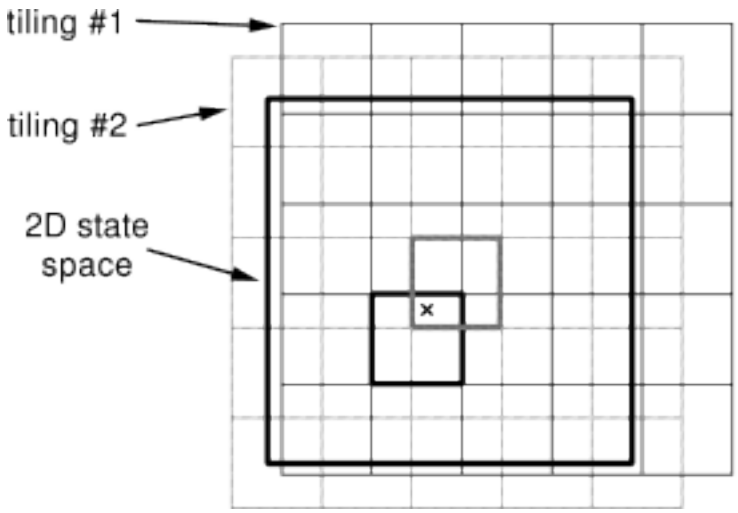




Tile Coding



- Binary feature for each tile
- Number of features present at any one time is constant
- Binary features means weighted sum easy to compute
- Easy to compute indices of the freatures present



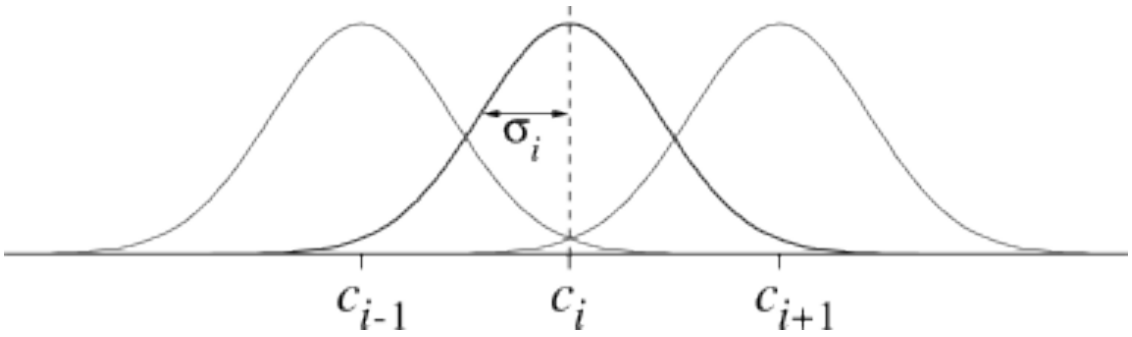
Shape of tiles ⇒ Generalization

#Tilings ⇒ Resolution of final approximation

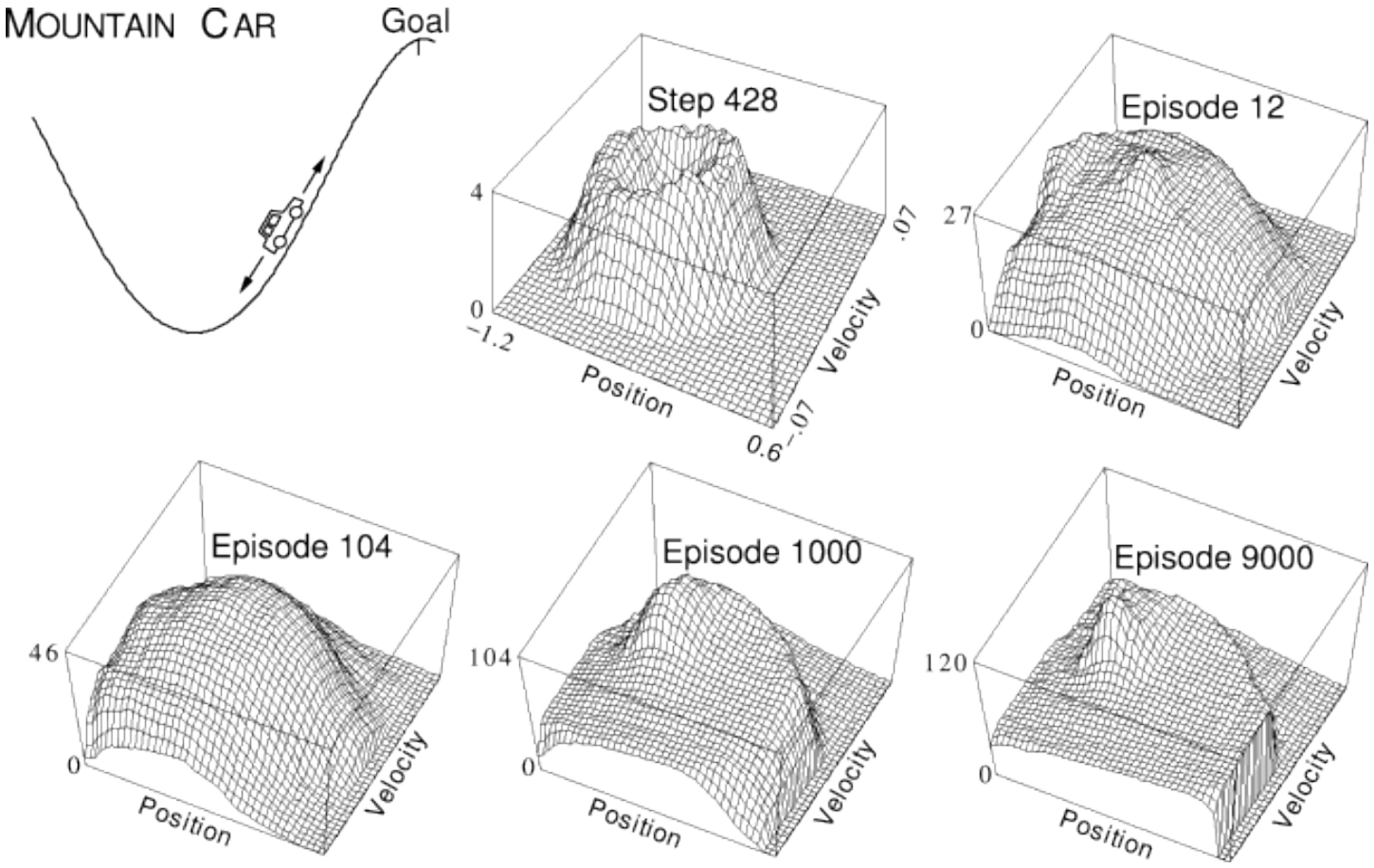
Radial Basis Functions (RBFs)

e.g., Gaussians

$$f_i(s) = \exp\left(-\frac{\|s - c_i\|^2}{2\sigma_i^2}\right)$$

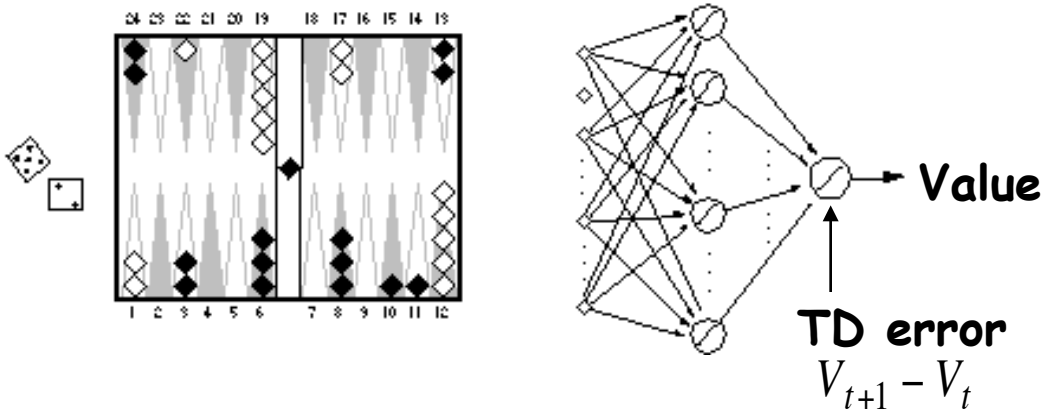


Mountain-Car Task



Nonlinear Function Approx.

TD-Gammon **Tesauro, 1992–1995**



Action selection by 2–3 ply search

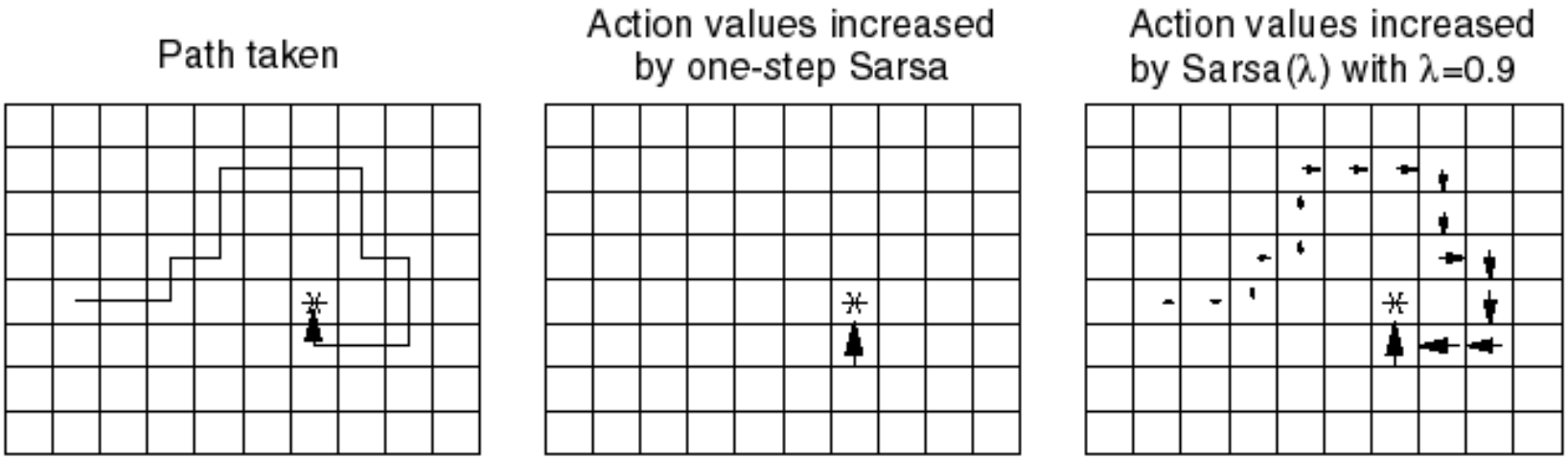
Start with a random network

Play very many games against self

Learn a value function using TD with backpropagation from this simulated experience

This produces arguably the best player in the world

Eligibility traces: Sarsa(λ) Example



- With one trial, the agent has much more information about how to get to the goal
  - not necessarily the *best* way
- Can considerably accelerate learning

Summary

- Quick review of passive RL
  - More on TD learning
  - TD learning of action values
- Active RL
  - Key fact
  - Exploration
  - SARSA
  - Q-Learning
  - TD and Dopamine
  - Generalization: function approximation
  - Eligibility traces

Next Class

- Review for the final