Function Approximation in Reinforcement Learning

Geoff Gordon
ggordon@cs.cmu.edu

November 15, 1999

Overview

Example (TD-Gammon)

Admission

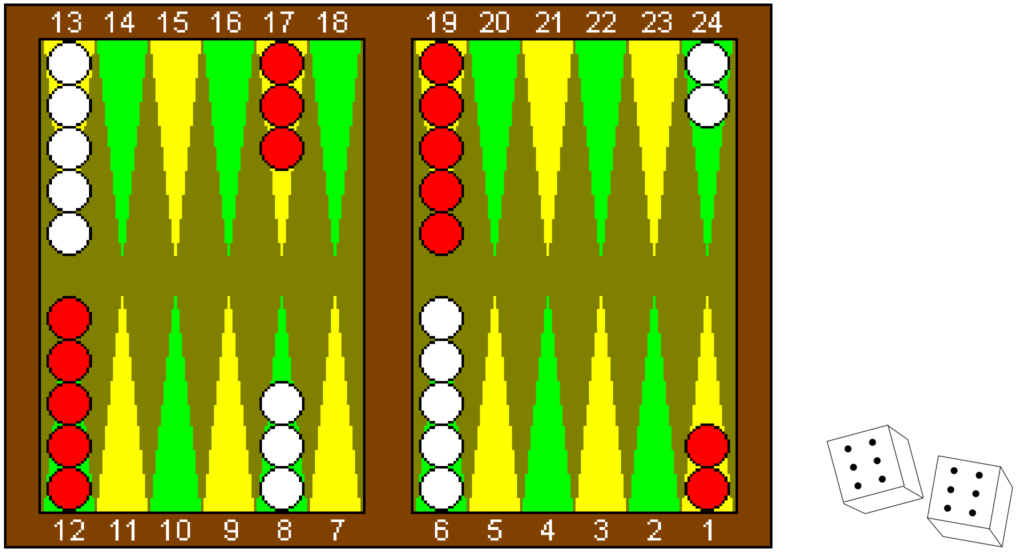
Why approximate RL is hard

TD(λ)

Fitted value iteration (collocation)

Example (k-nn for hill-car)

Backgammon



Object: move all pieces off first

Opponent can block you or send you backwards

Backgammon cont'd

30 pieces (15 each for you and opponent)

24 points + bar + off = 26 locations

2 dice (36 possible rolls)

Rough (over)estimate: $30^{26} \times 36 \times 2 \approx 1.8 \times 10^{40}$ states

Branching factor: $36 \times$ about 20

Complications: gammon, backgammon, doubling cube

TD-Gammon [Tesauro]

Neural network with 1 hidden layer

≈ 200 inputs: board position

40-80 hidden units

1 output: win probability

Sigmoids on all hidden, output units (range $[0, 1]$)

Using RL

Win probability is value function if:

- Reinforcement $= \begin{cases} 1 & \text{win} \\ 0 & \text{loss} \\ 0 & \text{game not over} \end{cases}$
- Discount is $\gamma = 1$

So weights for network should approximately satisfy Bellman equation

Warning: devil is in details!

For now: ignore details, use TD(0)

Review of TD(0)

Write $v(x|w)$ for output of net with inputs x and weights w

Given a transition x, a, r, y

Update $w := w_{\text{old}} + \eta(r + \gamma v(y|w_{\text{old}}) - v(x|w_{\text{old}}))\, \nabla_w v(x|w)|_{w_{\text{old}}}$

Ignores a (assumes it was chosen greedily)

TD(0) is not gradient descent

Max isn't differentiable

Haven't said how to choose transition (won't be independent)

Step direction is not a full gradient

Bellman error

Write $\text{Error}(x, r, y|w) = (r + \gamma v(y|w) - v(x|w))^2$

Write $\text{Error}(x, r, y|w_1, w_2) = (r + \gamma v(y|w_2) - v(x|w_1))^2$

TD uses $\nabla_w \text{Error}(x, r, y|w, w_{\text{old}})|_{w_{\text{old}}}$

GD would use $\nabla_w \text{Error}(x, r, y|w)|_{w_{\text{old}}}$

Representation

Raw board or raw + expert features

Raw:

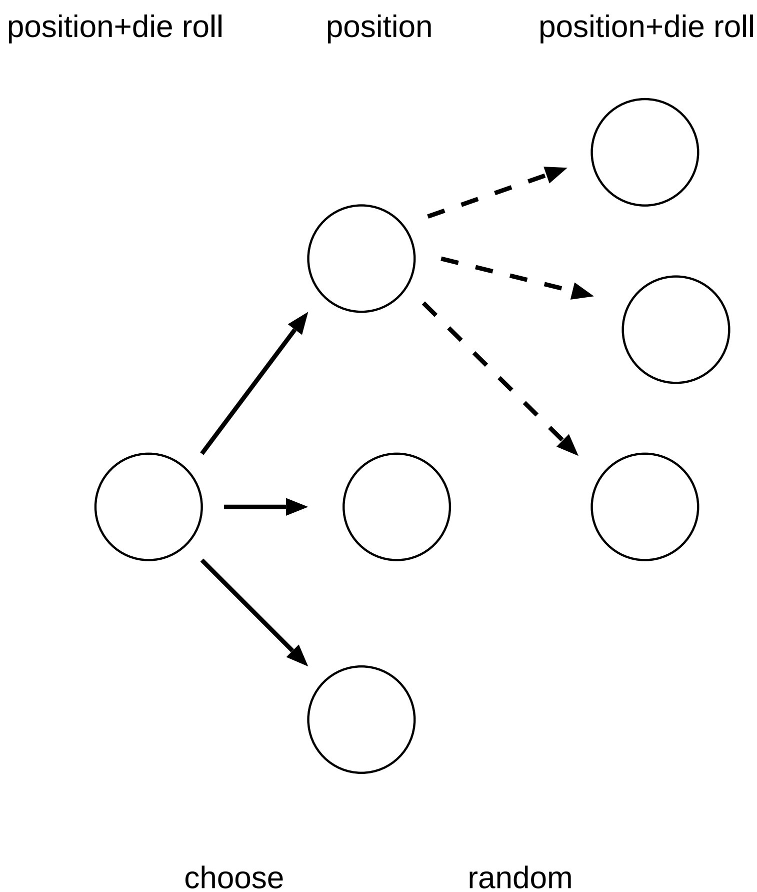
- Are there at least k of my pieces at location x?
- Are there at least k opposing pieces at location x?
- Is it my turn? My opponent's?

For $k > k_{\text{max}}$, extra pieces added to last unit's activation

k_{max} is 4 for points, 1 for bar and off

All inputs scaled to lie approximately in $[0, 1]$

A useful trick



Die rolls aren't represented

Implicit in lookahead

Training

Start from zero knowledge

Self play (up to 1.5 million games)

Data: state, die roll, action, reinforcement, state, . . .

Compress to: state, state, . . . , state, win/loss

Results

Raw features give pretty good player (close to best computer player at time)

Raw+expert features give world-class player

Absolute probabilities not so hot (10% error)

Relative probabilities very good

What's easy about RL in backgammon

Random

- strong effective discount
- kick out of local minima

Guaranteed to terminate

Perfect model, no hidden state

Self-play allows huge training set

What's hard about RL in backgammon

Game (minimax instead of min or max)

Nonlinear function approximator

- usual worry of local minima
- complete divergence is also possible [Van Roy]

Changing data distribution

- changing policy causes change in importance of states
- could get stuck (checkers, go)
- could oscillate forever
- wrong distribution could cause divergence [Gordon, Baird]

Huge variety of approximate RL algorithms

Incremental vs. batch

Model-based vs. direct

Kind of function approximation

Control vs. estimation

State-values vs. action-values (V vs. Q)

Vs. no values (policy only)

One fundamental question

What does "approximate solution to Bellman equations" mean?

Subquestion: how do we find one?

Possible answers

Bellman equations are a red herring

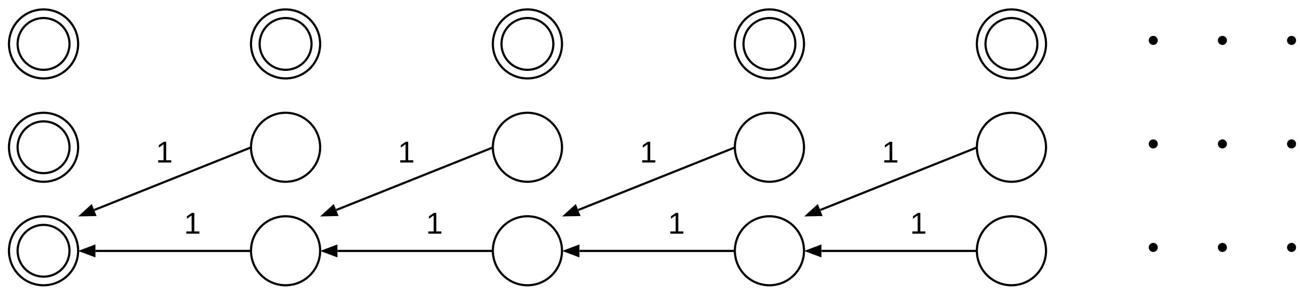
Minimum squared Bellman error

Minimum weighted squared Bellman error

Relaxation of Bellman equations

- satisfy at some states
- other

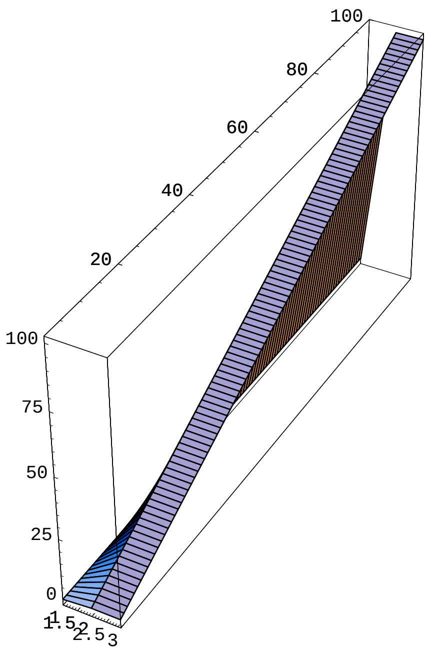
Minimum SBE example



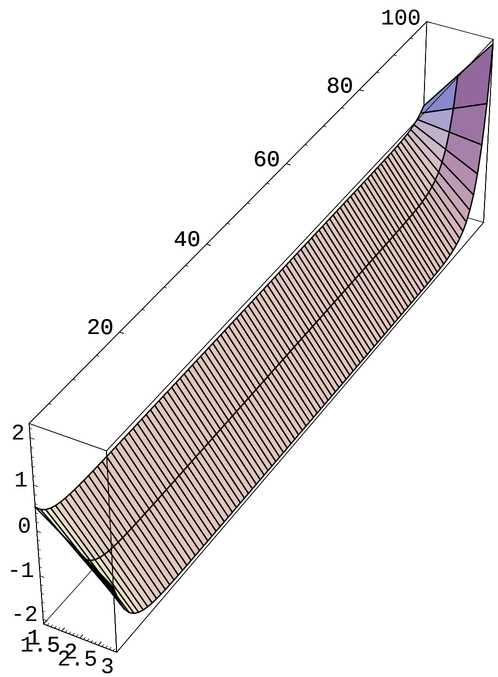
3×100 grid of states

Fit a line to each row: $v(i, j) = a_j + b_j i$

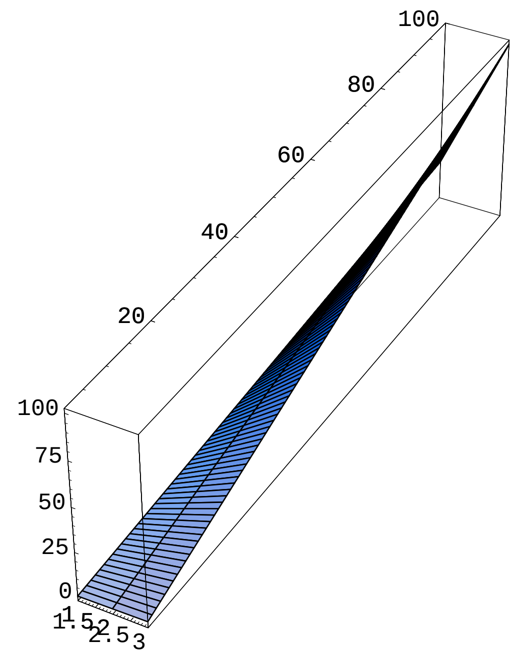
Minimum SBE example cont'd



Exact



Min SBE



Min weighted SBE

Flows

$F_\pi(x, a)$ = expected # of times we visit state x and perform action a

Fixed policy π

Fixed starting frequencies $f_0(x)$

Measures how important (x, a) is to value of π

Flows cont'd

If $\gamma < 1$, use discounted flows

Visit (s, a) at steps 2 and 7, discounted flow is $\gamma^2 + \gamma^7$

For all π, F_π satisfies conservation equation

$$\sum_a F(x, a) = \gamma \left(f_0(x) + \sum_{y,a} F(y, a) P(x|y, a) \right)$$

Optimal flows

$F^* = F_{\pi^*}$ are optimal flows

Optimal flows determine optimal policy

F^* maximizes total expected discounted reinforcement

$$\sum_{x,a} E(r(x,a))F(x,a)$$

Aside:

- Can find optimal flows using nonnegativity, conservation, and maximization of reinforcement
- This is dual (in sense of LP) to solving Bellman equations

Choosing weights

Optimal flows would be good weights

If just one policy, can find flows

Otherwise, chicken and egg problem

Iterative improvement? Yes, but none known to converge.

Research question: EM algorithm?

Matrix form of TD(0)

Assume one policy, n states

Write P for transition probability matrix ($n \times n$)

Write r for reinforcement vector ($n \times 1$)

p_{ij} is probability of going to state j from state i, r_i is expected reinforcement

P_i (ith row of P) is distribution of next state given that current state is i

Matrix form of TD(0) cont'd

Write v for value function ($n \times 1$)

v_i is value of state i

$(\gamma P v + r) - v$ is vector of Bellman errors

Bellman equation is $(\gamma P - I)v + r = 0$

Write $E = \gamma P - I$

Matrix form of TD(0) cont'd

Assume v is linear in parameters w ($k \times 1$)

$\nabla_w v_i$ is a constant k-vector (call it A_i)

$\nabla_w v$ is a constant $n \times k$ matrix (call it A)

Assume $w = 0$ corresponds to $v(x) = 0$ for all x

Expected update

$$
\begin{aligned}
& E((r + \gamma v(y|w) - v(x|w))\nabla_w v(x|w)) \\
= & \sum_x P(x) E((r + \gamma v(y|w) - v(x|w)) a_x | x) \\
= & \sum_x P(x) a_x (E(r|x) + \gamma E(v(y|w, x)) - v(x|w)) \\
= & \sum_x P(x) a_x (r_x + \gamma E(A_y \cdot w | x) - A_x \cdot w) \\
= & \sum_x P(x) a_x (r_x + \gamma w \cdot \sum_y P(y|x) a_y - A_x \cdot w)
\end{aligned}
$$

Expected update cont'd

$$
\begin{aligned}
&\sum_x P(x) a_x (r_x + \gamma w \cdot \sum_y P(y|x) a_y - A_x \cdot w) \\
&= \sum_x P(x) a_x (r_x + \gamma w \cdot (P_x^\top A) - A_x \cdot w) \\
&= \sum_x P(x) a_x (r_x + (\gamma P_x^\top A - A_x) \cdot w) \\
&= \sum_x P(x) a_x (r_x + (\gamma P_x - U_x)^\top A \cdot w) \\
&= (DA)^\top (R + (P - I) A w)
\end{aligned}
$$

TD(λ)

Data $x_1, r_1, x_2, r_2, \ldots$

TD(0) uses 1-step error $(r_1 + \gamma x_2) - x_1$

Could use 2-step error $(r_1 + \gamma r_2 + \gamma^2 x_3) - x_1$

TD(λ) weights k-step error proportional to λ^k

As $\lambda \to 1$, approaches supervised learning

Convergence

Expected update is $A^\top D(EAw + R)$

Can write $w := (I - \eta A^\top DEA)w_{\text{old}} + k + \text{error}$

Sutton (1988) proved that $A^\top DEA$ is positive definite

So $(I - \eta A^\top DEA)$ has radius < 1 for small enough η

So $\exists$ a norm in which $(I - \eta A^\top DEA)$ is a contraction

Convergence of expectation follows; tricky stochastic argument to show that random perturbations don't kill convergence [Dayan, Jaakola, Jordan, Singh, Tsitsiklis, Van Roy]

TD(0) and min weighted SBE

TD(0) does not find minimum of weighted SBE, but close

TD(0) solves $A^\top D(EAw + R) = 0$

Min WSBE satisfies $(DEA)^\top D(EAw + R) = 0$

Note: solving TD eqs in batch mode is called LS-TD (for least squares, even though it's not minimizing squared anything) [Bradtke & Barto]

Collocation

Satisfy Bellman equations exactly at m states

If k parameters, might expect $m = k$

But Bellman equations are nonlinear, so in general

- might need more or fewer than k equations
- solution might not be unique
- hard to find

Averagers

For some function approximators [Gordon, Van Roy]

- $m = k$ works
- solution is unique
- easy to find

Called averagers

Examples: linear interpolation on mesh, multilinear interpolation on grid, k-nearest-neighbor

Fitted value iteration

Pick a sample X

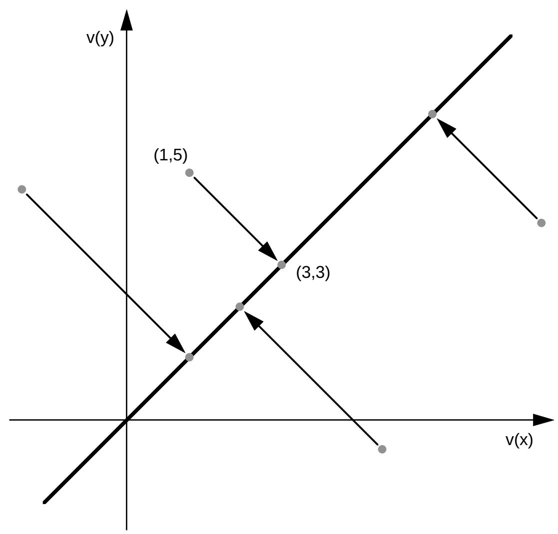
Remember $v(x)$ only for $x \in X$

To find $v(x)$ for $x \notin X$ use k-nearest-neighbor, linear interpolation, ...

One step of fitted VI has two parts:

- Do a step of exact VI
- Replace result with a cheap approximation

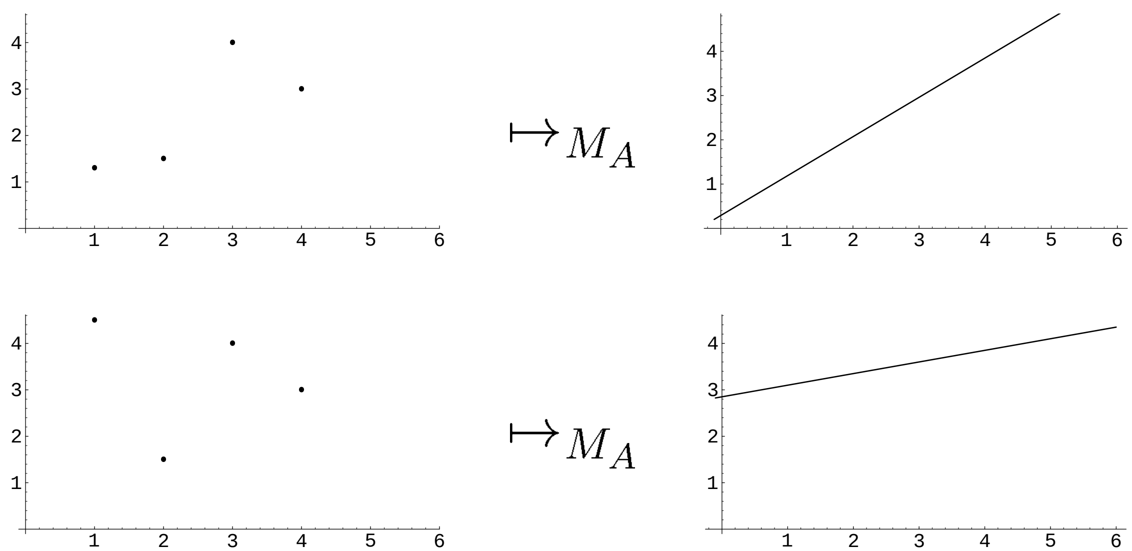
Approximators as mappings



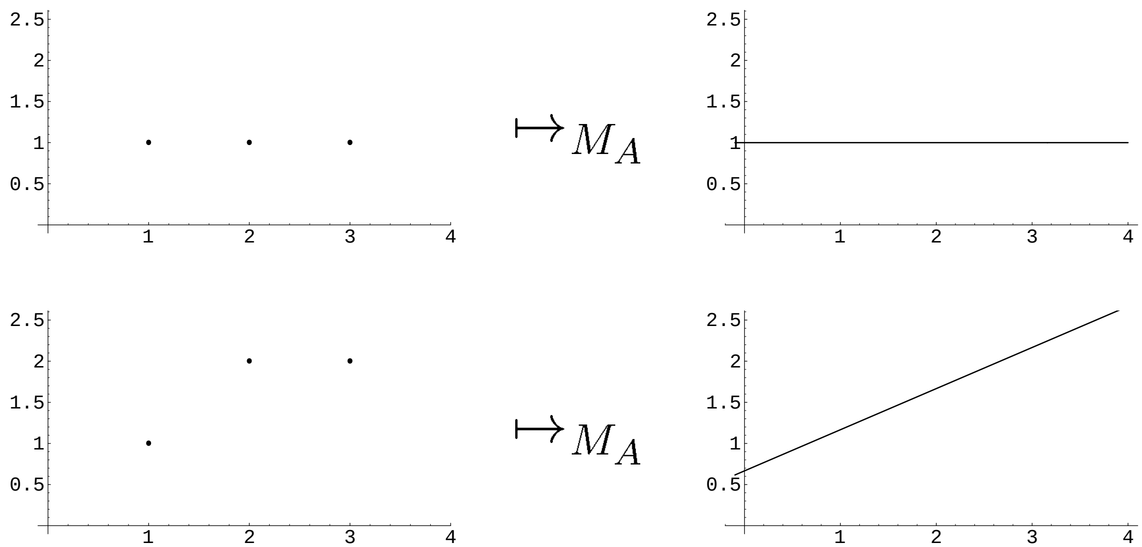
MDP with 2 states, so value fn is 2D: $(v(x), v(y))$

Approximate with $v(x) = v(y) = w$

Approximators as mappings



Exaggeration

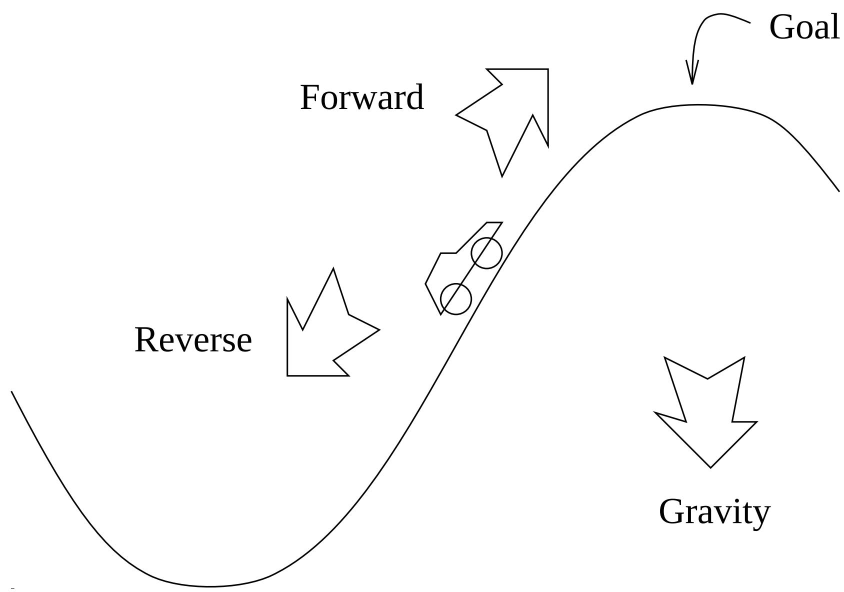


Two similar target value functions

Larger difference between fitted functions

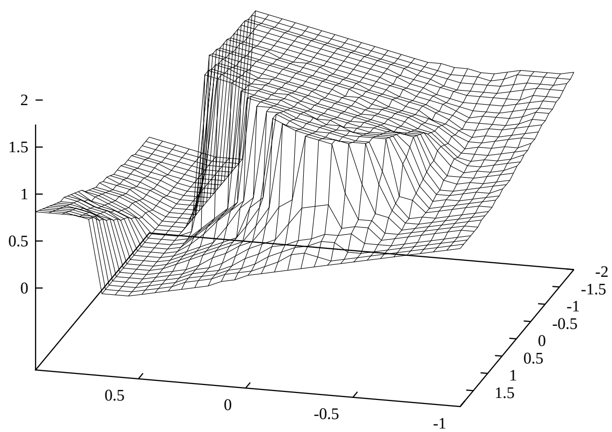
Exaggeration = max-norm expansion

Hill-car

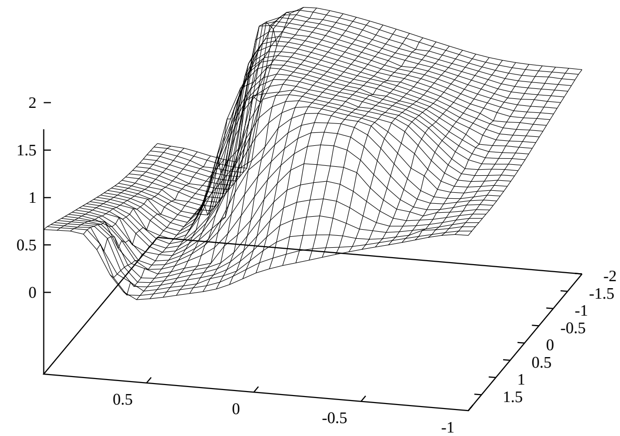


Car tries to drive up hill, but engine is weak so must back up to get momentum

Results

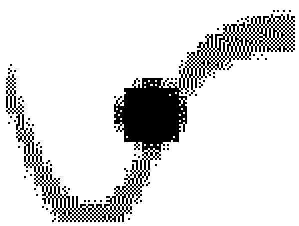


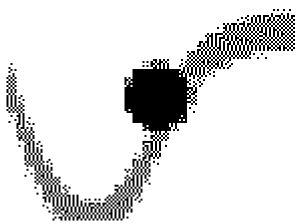
Exact

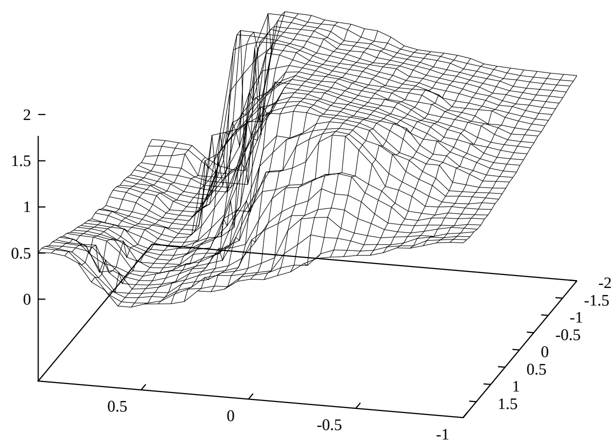


Approximate

Results the hard way







Interesting topics left uncovered

Hidden state (similar problems to fn approx)

Eligibility traces

Policy iteration and λ-PI

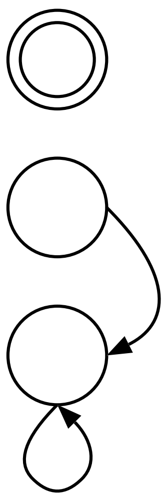
Actor-critic architectures

Adaptive refinement (use flows, policy uncertainty, and value uncertainty to decide where to split) [Munos & Moore]

Duality and RL

Stopping problems

Bonus extra slide: RL counterexample



Start with values $(0, 0, 0)$
Do one backup: $(0, 1, 1)$
Fit a line: $(-\frac{1}{6}, \frac{1}{2}, \frac{7}{6})$
Do another backup: $(0, 1 + \frac{7\gamma}{6}, 1 + \frac{7\gamma}{6})$
For $\gamma \geq \frac{6}{7}$, divergence