RL: Lecture 22—Approximation

Harvey Mudd College

April 20, 2020

Neil Rhodes

Wednesday Guest Lecture
Prof. Erin Talvitie
Model-based RL

Grading up-to-date
except:
A few late submittal PA 6

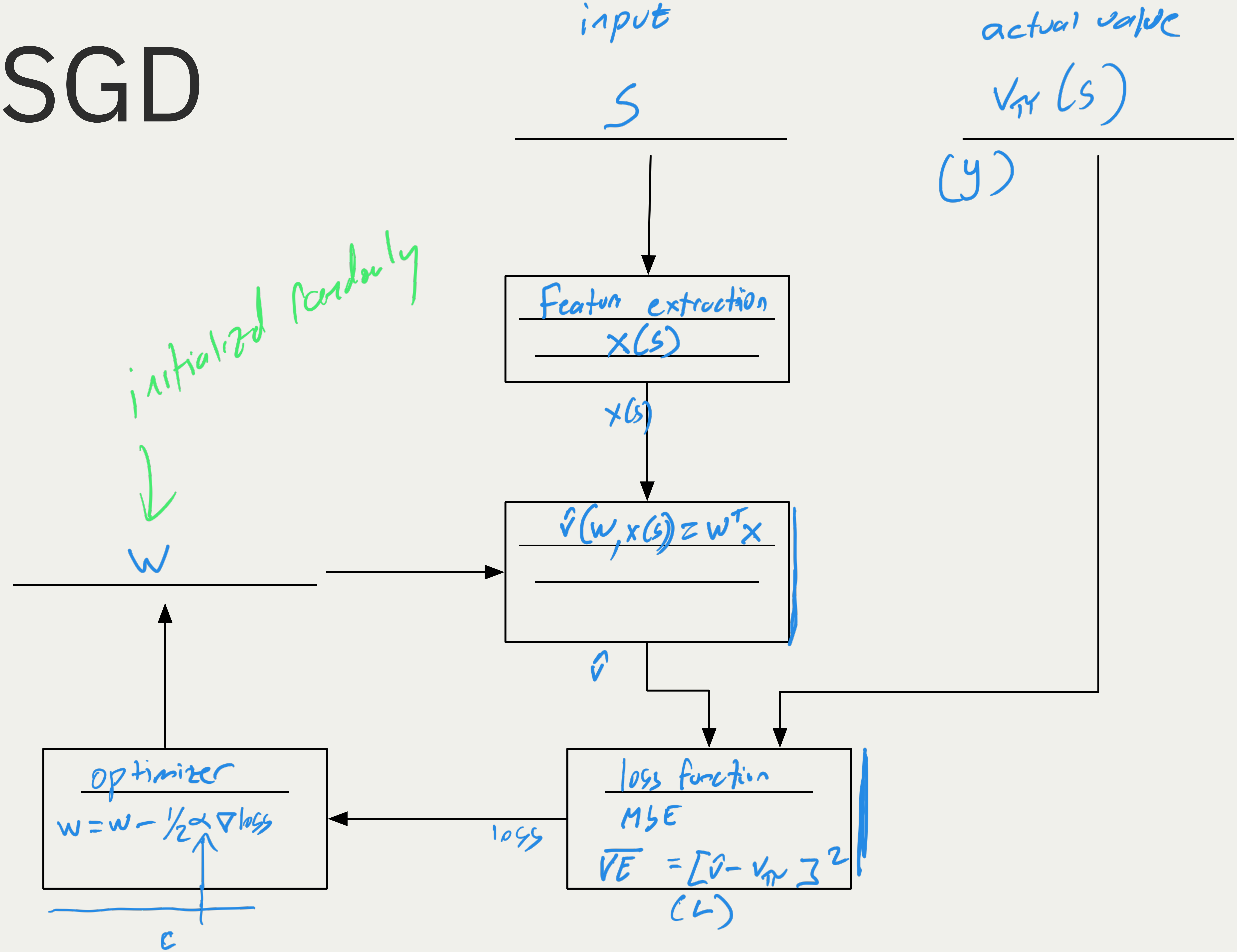
All have an option for P/Nocredit

Quiz today
on
Gradescope

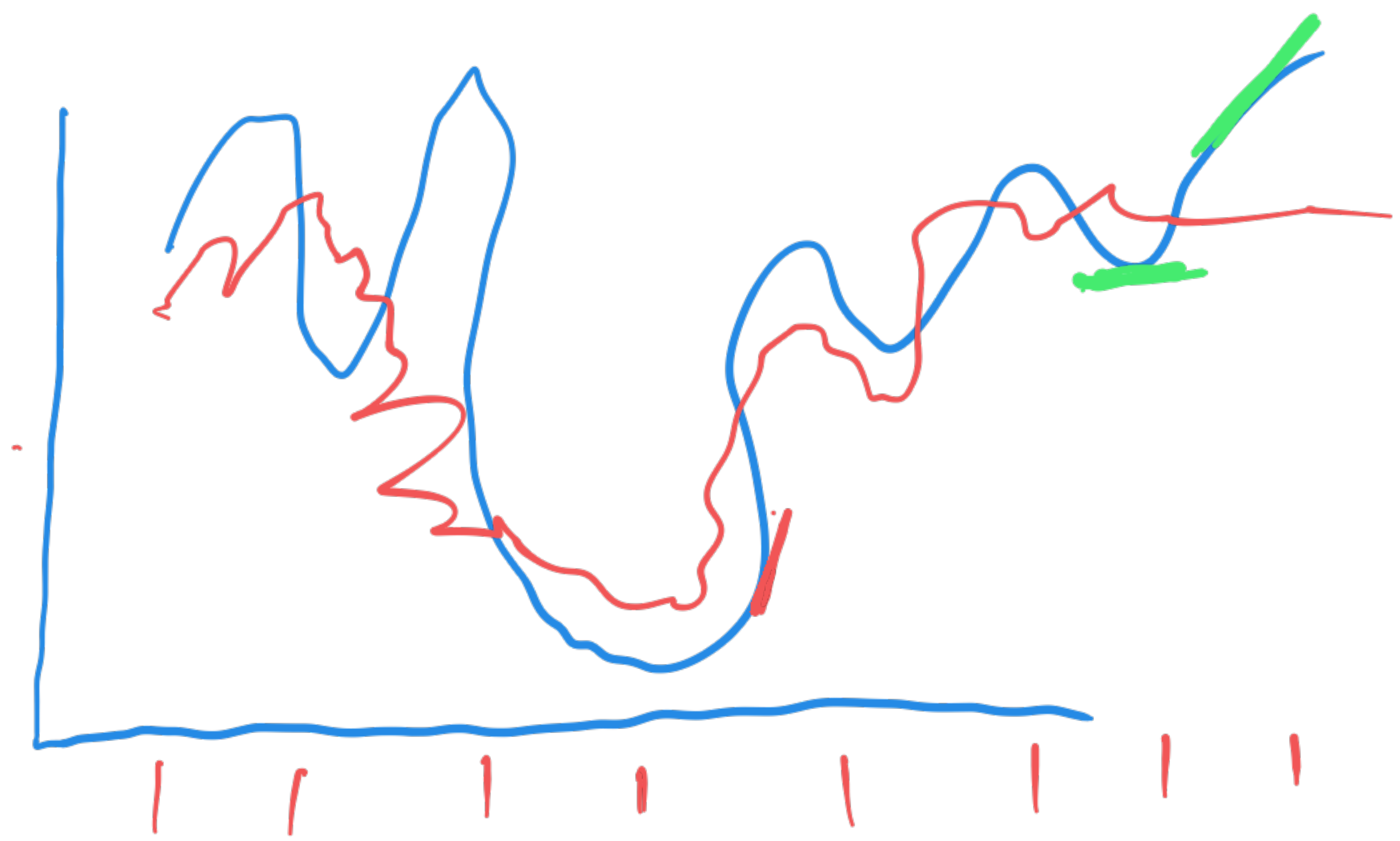
Overview

- Stochastic Gradient Descent

SGD



input
S
actual value
Vπ (s)
(y)
Feature extraction
x(s)
x(s)
initialized randomly
w
v̂(w, x(s)) = wᵀx
v̂
optimizer
w = w − ½α ∇loss
c
loss
loss function
MSE
VE = [v̂ − vπ]²
(L)



gradient descent

in between
mini-batches

one input
quicker to calculate
updating w sooner

all inputs (training examples)
1 problem
lots of computation
advantage:
gradient is exactly right

TD loop:

$$\underbrace{q(\overset{\text{input}}{\overbrace{s,a}}) = \overbrace{q(s,a) + \alpha[R + \max_{a'} \underbrace{\boxed{Q(s',a')}}_{\text{approximate } Q \text{ with our } \hat{q}}]}^{y}}_{\text{new value of } q(s,a)}$$

our new training example is

(s,a) pair

new calculated $q(s,a)$

target value (or y)

HW 10

$\hat{v}(s) = \boxed{\quad}$

input

state 2

$v_\pi(s)$

7

output

$\cancel{x(s) = s}$

$s = 2$

$\hat{v}(w,s) = w_2 s^2 + w_1 s + w_0$

$= 3\cdot 4 + 3\cdot 2 + 1$

$= 19$

$w = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$

19

7

State loss

$= (19-7)^2$

$= 12^2 = 144$

loss $= (\hat{v} - v_\pi)^2$

loss $= \frac{1}{2}(\hat{v} - v_\pi)^2$

loss $=$ 144

Optimizer

$\frac{\partial \text{loss}}{\partial w_0} = \frac{\partial \text{loss}}{\partial \hat{v}} \cdot \frac{\partial \hat{v}}{\partial w_0}$

$\frac{\partial \text{loss}}{\partial \hat{v}} = 2(\hat{v} - v_\pi)$

$\frac{\partial \hat{v}}{\partial w_0} = 1$

$\frac{\partial \hat{v}}{\partial w_1} = s$

$\frac{\partial \hat{v}}{\partial w_2} = s^2$

$w_0 = w_0 - \boxed{\cancel{\tfrac{1}{2}}\alpha} \frac{\partial \text{loss}}{\partial w_0} = 1 - \cancel{\tfrac{1}{2}}\alpha \cancel{2}(\hat{v} - v_\pi) = 1 - \alpha 2(19-7) = 1 - .12 = .88$

$.24$

$= .76$

$$\nabla loss = \begin{bmatrix} \frac{\partial loss}{\partial w_0} & \frac{\partial loss}{\partial w_1} & \frac{\partial loss}{\partial w_2} \end{bmatrix}$$

$$= \begin{bmatrix} (\hat{V} - V_\pi)1, & (\hat{V} - V_\pi)s, & (\hat{V} - V_\pi)s^2 \end{bmatrix}$$

$$= \begin{bmatrix} 12, & 24, & 48 \end{bmatrix}$$

$$w = w - \alpha(\nabla loss)^T$$

$$= \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix} - .01 \begin{bmatrix} 12 \\ 24 \\ 48 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix} - \begin{bmatrix} .12 \\ .24 \\ .48 \end{bmatrix} = \begin{bmatrix} .88 \\ 2.76 \\ 2.52 \end{bmatrix}$$

Features for Linear Methods

Let's say we have a state (s_1, s_2)

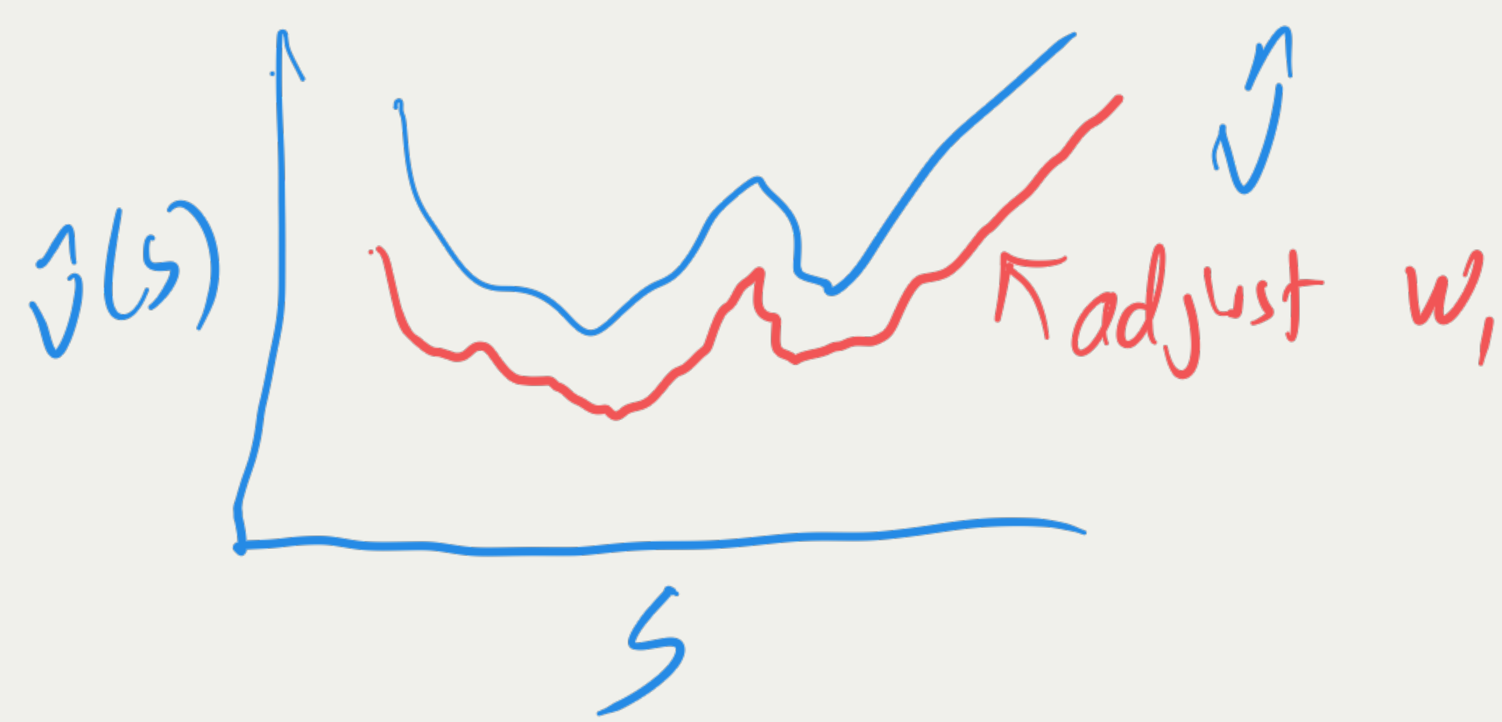
Polynomial

linear linear

Example: $x(s) = [1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2]$

$x(s) = \begin{bmatrix} 1 \\ \vdots \end{bmatrix}$

(dlr upcard, total so far)



non-NNs:

$s \to$ feature extraction $\to x(s) \to$ model

function approx

w

end-to-end learning

NNs: $s \to$ NN

beg part learns to do feature extraction

chess board

8x8 matrix w/ number @ each entry representing piece

8x8 for white pawns

8

8

white rooks

Cartpole problem

Actual Cartpole video Computer Cartpole Video

State: 2.39 2.38999995

- Cart Position $[-2.4, 2.4]$ $\mathbb{R}$ from left to right
- Cart velocity: $\mathbb{R}$
- Pole Angle: $[-41.8^\circ, 41.8^\circ]$ $\mathbb{R}$
- Pole tip velocity: $\mathbb{R}$ speed of pole (cartpos, cart v, poleangle, pole velocity)

$\mathbb{R}^4$

Action:

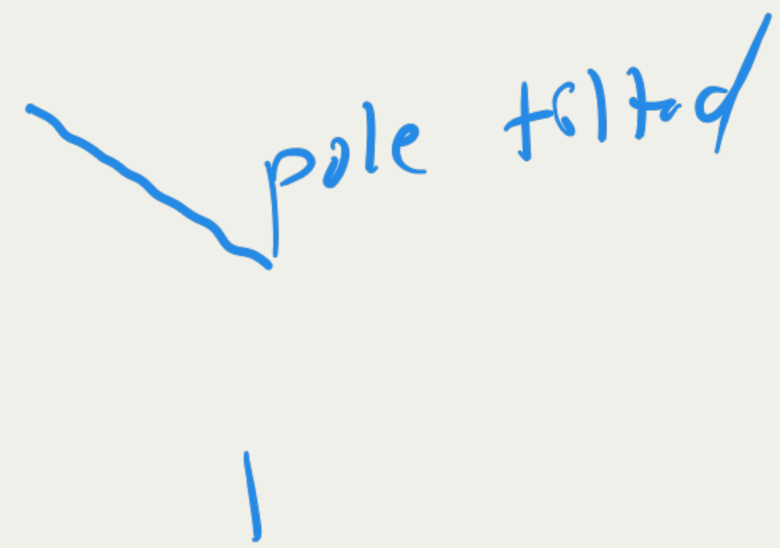
- Left
- Right

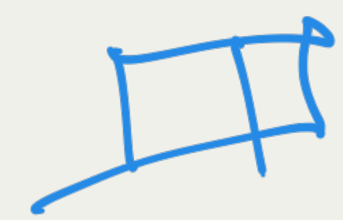
Cartpole problem

Reward: 1 for every step taken, including termination step

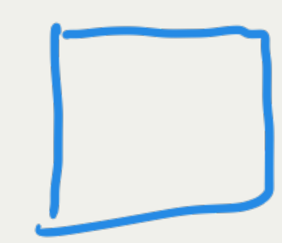
Episode Termination:

- Pole angle more than $\pm 12°$
- Cart position more than ± 24
- Episode length ≥ 500





4-dimensional bucket



assumption is states in bucket can be treated the same way

why approximation
infinite # of states
SGD deals w/ approximation
Linear model:
how many features?
$[1, s_1, s_2, s_3, s_4]$ $s_3 s_4$

OpenAI Gym

Atari games

walking 4-legged creatures

Gym is a toolkit for developing and comparing reinforcement learning algorithms

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  action = env.action_space.sample() # your agent here (this takes random actions)
  observation, reward, done, info = env.step(action)

  if done:
    observation = env.reset()
env.close()
```

to make a frame of movie

equiprobable

imm reward

over?

use SGD here