Wednesday Guest Lecture
Prof. Erin Talvitie
Model-based RL

Grading up-to-date
except:
A few late submitted PA 6
All have an option for P/NCredit

Quiz today
On grade scope
Overview

- Stochastic Gradient Descent
SGD

input $S$

$\text{Feature extraction}$

$X(S)$

$\mathcal{V}(w, X(S)) = w^T X$

optimizer

$w = w - \frac{1}{2} \alpha \nabla \text{loss}$

loss function

$\text{MSE}$

$\mathcal{V}_E = \frac{1}{2} (y - y')^2$

actual value $v_{xy}(S)$
gradient descent
in between mini-batches

all inputs (training example)

1 problem
lots of computation
advantages:
gradient is exactly right

one input
queue to calculate
updating w.sense
TD loop:

\[ g(s,a) = g(s,a) + \alpha \left[ R + \max_{a'} Q(s',a') - g(s,a) \right] \]

Approximate \( Q \) with our \( \hat{Q} \)

New value of \( g(s,a) \)

Our new training example is calculated as:

\[ \text{new } g(s,a) = \text{target value for } (s,a) \]
$v(s) = \frac{1}{2}$

$w = \begin{bmatrix} \frac{1}{3} \\ \frac{2}{3} \end{bmatrix}$

Input State $\mathbf{z}$

$\mathbf{z} \cdot \mathbf{w} = 5$

$S = 2$

$\frac{\partial \text{loss}}{\partial \mathbf{w}} = \frac{\partial \text{loss}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{w}}$

$\frac{\partial \text{loss}}{\partial \mathbf{z}} = 2 (v - v_f)$

$\frac{\partial v}{\partial v_f} = 1$

$\frac{\partial v}{\partial \mathbf{w}} = 5$

$\frac{\partial \text{loss}}{\partial \mathbf{v}} = \frac{1}{2} (v - v_f)^2$

$\text{loss} = \frac{1}{2} (v - v_f)^2$

$\text{loss} = (19 - 7)^2 = 12^2 = 144$

$w_0 = w_0 - \frac{\partial \text{loss}}{\partial w_0} = 1 - \frac{1}{2} \cdot 12 (v - v_f) = 1 - 6 (19 - 7) = 1 - 12 = 88 \div 24 = 3.66$
\[ \nabla \text{loss} = \left[ \frac{\partial \text{loss}}{\partial w_0}, \frac{\partial \text{loss}}{\partial w_1}, \frac{\partial \text{loss}}{\partial w_2} \right] \]

\[ = \left[ (v - v_i)_1, (v - v_i)_5, (v - v_i)_5^2 \right] \]

\[ = \sum 12, 24, 48 \]

\[ w = w - \alpha (\nabla \text{loss})^T \]

\[ = \left[ \begin{array}{c} 1 \\ 3 \\ 3 \end{array} \right] - 0.01 \left[ \begin{array}{c} 12 \\ 24 \\ 48 \end{array} \right] \]

\[ = \left[ \begin{array}{c} 1 \\ 3 \\ 3 \end{array} \right] - \left[ \begin{array}{c} 12 \\ -24 \\ -48 \end{array} \right] = \left[ \begin{array}{c} 0.88 \\ -2.76 \\ 2.52 \end{array} \right] \]
Features for Linear Methods

Polynomial

Example: \[ x(s) = [1, s_1, s_2, s_1 s_2, s_1^2, s_2^2, s_1 s_2^2, s_1^2 s_2, s_1^2 s_2^2] \]
8x8 matrix \( w \) of each entry of \( \text{chess board} \).

Each entry \( w \) encodes features of a chess piece feature extraction.

NNs: \( s \xrightarrow{\text{conv}} \nabla \text{NNs} \)

Non-NNs: \( s \xrightarrow{\eta} \text{model} \)

8x8 for which mean

8x8 with row
Cartpole problem

Actual Cartpole video  Computer Cartpole Video

State:
• Cart Position $[-2.4, 2.4]$ 2.39 2.3899999995
• Cart velocity: $\mathbb{R}$
• Pole Angle: $[-41.8^\circ, 41.8^\circ]$ TR from left to right
• Pole tip velocity: $\mathbb{R}$ speed at pole

Action:
• Left
• Right

(cartpos, cartv, poleang, polevelinity)
Cartpole problem

Reward: 1 for every step taken, including termination step

Episode Termination:

• Pole angle more than \( \pm 12^\circ \)
• Cart position more than \( \pm 24 \)
• Episode length \( \geq 500 \)

Why approximation?

Infinite # of states

SBD deals w/ approximation

Linear model: how many features?

\( \{ 1, 5, 52, 53, 54 \} \)
OpenAI Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms

```python
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()
```

Atari games
walking 4-legged creatus