ECE-GY 6143
Intro to ML

- Recap: Unsupervised learning
- Reinforcement learning (RL)
  - Model-free RL
  - Policy gradients.

Supervised learning
- Regression
- Classification

Unsupervised learning
- Data visualization
- Dimensionality reduction.

- Dataset is static
- Learning is performed offline
- Predictions do not affect data samples

Reinforcement learning (RL)

- Dataset is dynamically changing
- Learning is online
- Explicit modeling of feedback loops.

Applications

- Stockmarket predictions.
- Gaming
  - AlphaGo
  - AlphaZero
  - OpenAI Dota
- Robotics

[RL $\leftrightarrow$ Control theory
Dynamical systems, optimal
nonlinear control].

Setting

- Environment
- Agent / state
- Actions at time step
- Actions $\rightarrow$ rewards
- Goal: Maximize rewards.

$$s_{t+1} = f(s_t, a_t)$$

$\hookrightarrow$ state transition function / Dynamical system.

$$\text{Reward} = r(s_t, a_t)$$

Goal:

Agent observes

$$\tau_t = (s_0, a_0, s_1, a_1, \ldots, s_t)$$

↳ Trajectory / rollout ↳ Horizon

$r(s_t, a_t)$ for $t = 0 \ldots (T-1)$

Predict a strategy / policy for deciding the next action.

Policy: function π → $\boxed{a_t = \pi(\tau_t)}$

Apply ML to solve this prediction problem!

→ Choose representation for π

→ Choose a loss function

$$R = \sum_{t=0}^{T-1} -r(s_t, a_t)$$

subject to $s_{t+1} = f(s_t, a_t)$

$a_t = \pi(\tau_t)$, s_0 is given.

→ Optimize this function.

π → Linear models, logistic, neural nets

Deep RL.

Need to figure out $\frac{\partial R}{\partial \pi}$!?

- Idea: Assume that π is not deterministic, instead it predicts a prob. distribution of actions.

↳ a_t random

s_t random

∴ R is a random variable.

Modified problem

expected value.

$$\min_{\pi} \; \mathop{E}_{\tau \sim \pi(\tau)} R(\tau)$$

$$= E\left[-\sum_{t=0}^{T-1} r(a_t, s_t) \right]$$

$$\text{s.t.} \quad s_{t+1} = f(s_t, a_t)$$

$$a_t = \pi(s_t)$$

Training samples → Multiple trajectories / rollouts.

Supervised → RL ← Unsupervised.

$x \to y$ only 1 y per rollout (x) x

Policy gradients.

- Q-learning
- Value iterations
- Actor-critic.

Assume π is a linear policy parameterized by θ.

$$\frac{\partial}{\partial \theta} E_{\pi(\tau)} R(\tau) = ?$$

Log-derivative trick.

$$\frac{\partial}{\partial \theta} \log \pi(\tau) = \frac{1}{\pi(\tau)} \cdot \frac{\partial}{\partial \theta} \pi(\tau)$$

$$\therefore \quad \frac{\partial}{\partial \theta} \pi(\tau) = \pi(\tau) \frac{\partial}{\partial \theta} \log \pi(\tau)$$

$$\frac{\partial}{\partial \theta} E_{\pi(\tau)} R(\tau) = \frac{\partial}{\partial \theta} \sum_{\tau} \pi(\tau) R(\tau)$$

$$= \sum_{\tau} R(\tau) \frac{\partial}{\partial \theta} \pi(\tau)$$

$$= \sum_{\tau} \pi(\tau) R(\tau) \frac{\partial}{\partial \theta} \log \pi(\tau)$$

$$\nabla E_{\pi(\tau)} R(\tau) = E_{\pi(\tau)} \left[R(\tau) \frac{\partial}{\partial \theta} \log \pi(\tau) \right]$$

Gradient of expected rewards

= Expected value of Reward x log derivative.

∴ Just like in SGD,

- sample different roll outs
- Compute approximation to the gradient of expected rewards.
- Update θ in that direction.

Repeat

1) Sample rollout $\tau = (s_0, a_0, s_1, \dots, s_T)$
2) Compute $R(\tau) = \sum_{t=0}^{T-1} r(s_t, a_t)$
3) $\theta \leftarrow \theta - \eta R(\tau) \frac{\partial}{\partial \theta} \log \pi_\theta(\tau)$

→ REINFORCE

↳ R appears but never its gradient.

↳ Useful when rewards are discrete.

↳ Environment f does not appear,

only the rollout τ reveals $R(\tau)$.

"Model-free reinforcement learning".

↳ vs. "Model-based RL"

Derivative-free optimization

Gradient descent: $\theta \leftarrow \theta - \eta \nabla R(\theta)$

Random search:

1) Pick a random vector v
2) Compute step size (either negative or positive) that minimizes $R(\theta + \eta v)$.
3) $\theta \leftarrow \theta + \eta v$.

Derivative-free optimization.