APS360 Fundamentals of AI

Lisa Zhang

Lecture 13; March 14, 2019

Agenda

- Reinforcement Learning
- Unfortunately no code today

Reinforcement Learning

Example

- Game playing
  - Backgammon
  - Go
- Robot control
  - Make a humanoid robot walk
  - Fly a helicopter
- Control a power station
- Manage an investment portfolio

Game Playing (Pong)

Let's say we want to build a neural network agent to play Pong.

- **Input**: pixel intensities of the screen over the last few frames
- **Output**: how to move the paddle
- Use a convolutional network, maybe with a few fully-connected layers at the end

Why can't we do this?

Game Playing: Difficulty

What would the loss function be?

- We don’t have the “correct” answer (ground truth move) at each step
- We have a signal to tell us whether our agent won the game.
- . . . but the signal is delayed (we don’t see the signal until the end of the game)

Reinforcement Learning

Reinforcement learning is different from supervised learning:

- There is no supervision (no "correct" answer), only a **reward** signal
- There is a notion of "time"
- Feedback is delayed, not instantaneous
- Agent's action affects the subsequent data it receives

Reward

- A **reward** R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward (with possible discounting)

Examples of Rewards

- Game playing:
  - positive reward for winning
  - negative reward for losing
- Making a humanoid walk:
  - positive reward for forward motion
  - negative reward for falling over
- Control a power station:
  - positive reward for producing power
  - negative reward for exceeding safety thresholds
- Manage an investment portfolio:
  - positive reward for each $ in bank

Reward Hypothesis

All goals can be described by the maximization of expected cumulative reward.

Terminology

- **Environment**
  - Provides the agent with the current **observation**
  - Provides a **reward** at each time step
- **Agent**
  - Computes the current **state** given the current observation, and potentially other saved information
  - Chooses an **action** at each time step, given the state

The goal of the agent is to selection actions to maximize total future reward.

Agent and the Environment

- The envrionment emits observation O_t
- The agent computes the next state S_t
- The agent executes action A_t given the state S_t
- The environment emits scalar reward R_t

Example: Go

- **State**: position on the board
- **Reward**:
  - 0 if the game hasn't ended
  - 1 if agent wins
  - -1 if opponent wins
- **Action**: make a legal Go move

Goal: learn a model that, given the *state*, finds the optimal *action*

Example: Walking

https://www.youtube.com/watch?v=EQRsvCwME0g

https://github.com/openai/gym/wiki/BipedalWalker-v2

- **State**: information about the joints (speed, angle, etc)
- **Reward**:
  - positive reward for moving forward
  - negative reward for falling over
- **Action**:
  - forces to apply to each joint

Goal: learn a model that, given the *state*, finds the optimal *action*

Example: Pong

https://www.youtube.com/watch?v=YOW8m2YGtRg

- **Observation**: current screen pixel intensities
- **State**: screen intensities in the last few time steps
  - to encode ball velocity

Major Components of an RL Agent

- **Policy**: an agent's behaviour function
- **Value function**: how "good" is each state and/or action (its expected future reward)
- **Model**: the agent's representation of the environment

Policy

- A **policy** is the agent's behaviour
- A function from state to action, which can be
  - **deterministic**: $a = \pi(s)$
  - **stochastic**: $\pi(a|s) = P[A_t = a|S_t = s]$

Value Function

- A **value function** is a prediction of future reward, assuming we follow a particular policy
- Used to evaluate the goodness or badness of states

Model

- A **model** predicts what the environment will do next
  - predict the next state
  - predict the next reward

Types of RL Agents

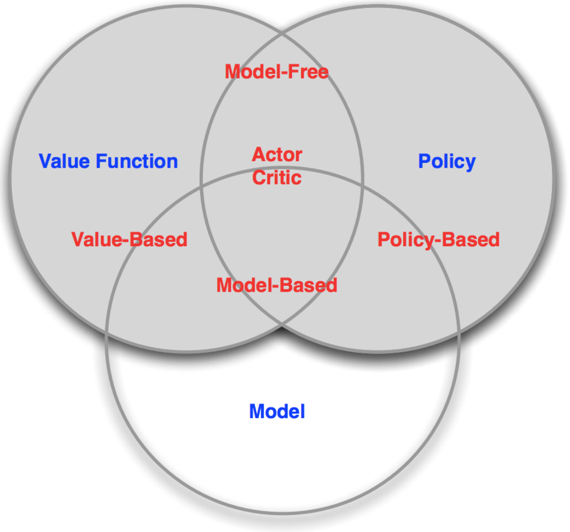
An RL agent can contain machine learning models that learns one or more of the policy, value function, or model.

- **Policy Based** agent:
  - learn policy function only (no estimate of value function, no prediction of the environment)
- **Value Based** agent:
  - learn value function only
  - implicit policy: choose the action that maximize the value function
- **Actor Critic**:
  - has an explicit policy (actor)
  - also learns a value function (critic)

Model Free vs Model Based RL

- **Model Free** agent:
  - Only policy and/or value functions
  - **No model**
- **Model Based** agent:
  - Policy and/or value functions
  - Model (to predict what the environment will do next)

RL Agent Taxonomy



Reinforcement Learning

- The environment is initially unknown
- The agent interacts with the environment, and receives rewards
- The agent improves its policy based on those rewards

Example: Pong

https://www.youtube.com/watch?v=YOW8m2YGtRg

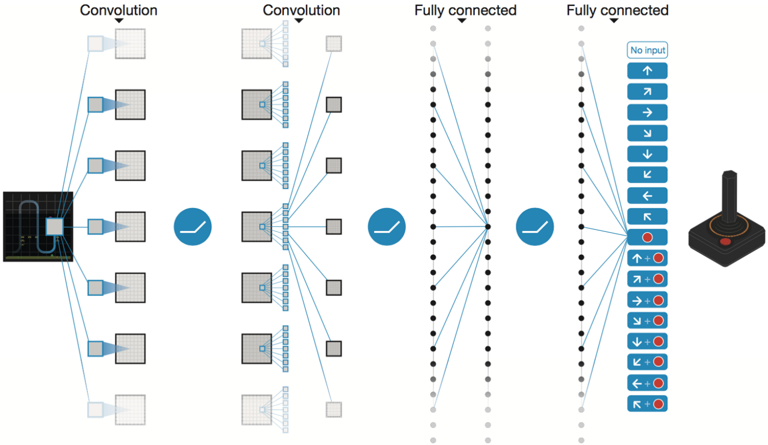
- **Policy Based** agent
  - Input to the policy function: pixel intensities (over last few time steps?)
  - Output of the policy function: velocity of paddle

Example: Breakout

https://www.youtube.com/watch?v=V1eYniJ0Rnk

- **Value Based** agent
  - Input to the value function:
    - pixel intensities (over last 4 time steps)
  - Output of the value function:
    - expected future rewar for each possible action

Model Architecture



From https://towardsdatascience.com/atari-reinforcement-learning-in-depth-part-1-ddqn-ceaa762a546f

Policy Learning

- A policy function takes the current state, and outputs the move the agent should take:
  - **deterministic**: $a = \pi(s)$
  - **stochastic**: $\pi(a|s) = P[A_t = a|S_t = s]$
- We can parameterize π using a neural network!

Back to our Pong example

We wanted to build a neural network agent to play Pong:

- **Input**: pixel intensities of the screen over the last few frames
- **Output**: how to move the paddle

What type of RL agent are we describing?

Back to our Pong example

We wanted to build a neural network agent to play Pong:

- **Input**: pixel intensities of the screen over the last few frames
- **Output**: how to move the paddle

What type of RL agent are we describing?

- policy-based, model-free
- we would be learning a **policy function**

Training the Policy Function: Idea

Why can't we just update the parameters of the neural network to maximize the immediate reward?

Training the Policy Function: Idea

Why can't we just update the parameters of the neural network to maximize the immediate reward?

- Some "good" moves may not produce an immediate reward
- We want to to maximize **all future reward**

Episode

An **episode** is a sequence:

$S_0, A_0, R_0, S_1, A_1, R_1, \ldots\ S_T, A_T, R_T$

that concludes with a terminal state.

In our game-playing example, an **episode** is one "match" or one "game".

Return

The **discounted return** at a time step R_t is defined to be:

$G_t = \sum_{s=t}^{T} \gamma^{s-t} R_s$

Where $0 < \gamma \leq 1$ is a constant **discount factor**.

With discounting, getting a reward now is better than getting the same reward later on

Training the Policy Function: Idea

Update the parameters of the neural network to maximize the **return**.

Rough idea:

- play several games (several episodes) using the current model weights
- for each episode i and time step k, compute the return at that step $G_k^{(i)}$
- modify the weights of the neural networks so that **actions** $A_k^{(i)}$ that produce **large returns** $G_k^{(i)}$ are **more likely**

Exploration vs Exploitation

- **Exploitation**: Make moves the function already thinks will lead to a good outcome vs
- **Exploration**: Try making novel moves and see if you discover a way to adjust the function to get even better outcomes

Need a good balance of **exploration** vs **exploitation** to learn a good RL agent

Where to go from here?

- Textbook: Reinforcement Learning by Sutton & Barto
- David Silver's video lectures:
  http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Half of this lecture is based on David Silver's first lecture
- Tic-tac-toe project:
  http://www.cs.toronto.edu/~guerzhoy/411_2018/projects/proj4/