Guided Tour of Machine Learning in Finance

Week 4: Reinforcement Learning

4-2-5-RL-and-IRL

Igor Halperin

NYU Tandon School of Engineering, 2017

Value Iteration: a recap

- The Bellman equation for optimal value function

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^*(s')$$

- Value Iteration algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^{(k)}(s')$$

- Optimal policy: $\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s' | s, a) V^*(s')$

Problems with DP and Value Iteration

- The Bellman equation for optimal value function

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^*(s')$$

- Value Iteration algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^{(k)}(s')$$

- Optimal policy: $\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s' | s, a) V^*(s')$

- **Problems** with the Value Iterations with Dynamic Programming:
  - Dynamics is assumed known - needs the model of the world!
  - Cannot be done exactly for large discrete or continuous state-action spaces

Enter Reinforcement Learning

- The Bellman equation for optimal value function

$$V^*(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^*(s')$$

- Value Iteration algorithm (for discrete state-action space)**:**
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^{(k)}(s')$$

- Optimal policy: $\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} p(s' | s, a) V^*(s')$

- **Problems** with the Value Iterations with Dynamic Programming:
  - Dynamics is assumed known - needs the model of the world!
  - Cannot be done exactly for large discrete or continuous state-action spaces

- Enter **Reinforcement Learning:**
  - Relies on samples from data, instead of a "model of the world"
  - Relies on function approximations to handle computational issues
  - Can use neural networks to parametrize the value function or policy function.

RL vs DP: model-free vs model-based

- Dynamic Programming approach assumes the **known dynamics**
  - If dynamics is unknown, it should be estimated from data
  - This can be difficult in real-world, multi-dimensional cases
  - Hard to control the impact of possible model misspecification on the resulting optimal policy
  - May be especially relevant for Finance

RL vs DP: model-free vs model-based

- Dynamic Programming approach assumes the **known dynamics**
  - If dynamics is unknown, it should be estimated from data
  - This can be difficult in real-world, multi-dimensional cases
  - Hard to control the impact of possible model misspecification on the resulting optimal policy
  - May be especially relevant for Finance

- Maybe we can do **without** learning dynamics?
  - "Do not use more concepts than you need to explain the observed facts" (V. Vapnik, "EMPIRICAL INFERENCE SCIENCE")

RL vs DP: model-free vs model-based

- Dynamic Programming approach assumes the **known dynamics**
  - If dynamics is unknown, it should be estimated from data
  - This can be difficult in real-world, multi-dimensional cases
  - Hard to control the impact of possible model misspecification on the resulting optimal policy
  - May be especially relevant for Finance

- Maybe we can do **without** learning dynamics?
  - "Do not use more concepts than you need to explain the observed facts" (V. Vapnik, "EMPIRICAL INFERENCE SCIENCE")

- **Reinforcement Learning**: a **model-free** approach
  - RL uses the data in the form of a set of samples
  - RL can be viewed as a sample-based, or trajectory-based, model-free DP!
  - RL may use its own internal "model of the world", this produces "model-learning" RL algorithms
- DP can be viewed as a model-based RL

RL vs DP: approximate vs exact

- Dynamic Programming approach works with small discrete state-action spaces
  - The value function or policy function can be represented in a tabulated form
  - Value Iteration or Policy Iteration amount to updating a set of discrete values for the value function or the policy function.
  - Can't be done this way for large discrete or continuous state-action spaces, need to rely on function approximation!
  - Function approximations are typically needed for DP/RL for Finance!

RL vs DP: approximate vs exact

- Dynamic Programming approach works with small discrete state-action spaces
  - The value function or policy function can be represented in a tabulated form
  - Value Iteration or Policy Iteration amount to updating a set of discrete values for the value function or the policy function.
  - Can't be done this way for large discrete or continuous state-action spaces, need to rely on function approximation!
  - Function approximations are typically needed for DP/RL for Finance!

- **Reinforcement Learning**: a **function approximation approach**
  - Parametric family for the value function, or policy function, or both
  - Substitute into the Bellman equation (for the Q-function) to find optimal parameters
  - Parametrized value function or policy function can be either linear or non-linear in tunable parameters
  - Neural networks can be used to provide flexible parametrized representations for a value function or a policy function

RL vs DP: approximate vs exact

- Dynamic Programming approach works with small discrete state-action spaces
  - The value function or policy function can be represented in a tabulated form
  - Value Iteration or Policy Iteration amount to updating a set of discrete values for the value function or the policy function.
  - Can't be done this way for large discrete or continuous state-action spaces, need to rely on function approximation!
  - Function approximations are typically needed for DP/RL for Finance!

- **Reinforcement Learning**: a **function approximation approach**
  - Parametric family for the value function, or policy function, or both
  - Substitute into the Bellman equation (for the Q-function) to find optimal parameters
  - Parametrized value function or policy function can be either linear or non-linear in tunable parameters
  - Neural networks can be used to provide flexible parametrized representations for a value function or a policy function
  - **Deep Reinforcement Learning:** Deep Neural Networks for function approximation + the Bellman equation.

Inverse Reinforcement Learning

- Value Iteration algorithm (for discrete state-action space):
  - Initialize the value function for each state $V(s) = V^{(0)}(s)$
  - Repeat the update of the value function until convergence:

$$V^{(k+1)}(s) = R(s) + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^{(k)}(s')$$

- Inverse Reinforcement Learning: we don't observe the rewards!

$$V^{(k+1)}(s) = \cancel{R(s)} + \max_{a \in A} \gamma \sum_{s' \in S} p(s' | s, a) V^{(k)}(s')$$

- **Inverse Reinforcement Learning:** objectives:
  - Learn the reward function $R(s)$
  - Learn the optimal policy $\pi^*(s)$
  - This is an ill-posed problem, many reward function and/or policy functions can be consistent with the same set of sample data
  - Some additional considerations/constraints are required to make the problem well-posed.
  - Computational methods based on iteration of actual rewards in Value Iteration can quickly become computationally very expensive.
  - IRL for finance?

Control question

Select all correct answers

1. Reinforcement Learning (RL) solves the same problem of optimal control as Dynamic Programming (DP), but relies on data samples instead of using a model of the world
2. RL can be viewed as a model-based DP.
3. DP can be viewed as a model-based RL.
4. Inverse Reinforcement Learning (IRL) can be viewed as a model-based DP.
5. The task of IRL is to learn the reward function and optimal policy.
6. The last of IRL is to build a model for the system dynamics.

Correct answers: 1, 3, 5.