Module 5: Lesson 4

Dynamic Programming: The windy gridworld, asynchronous DP

Outline

- Dynamic Programming in the windy gridworld
- Asynchronous Dynamic Programming
- Application: Optimal selling timing with two assets

Dynamic Programming in the windy gridworld

In the gridworld of Lesson 3, the agent had full control of the transition across states.

We can easily add some randomness to the gridworld environment by including "windy conditions": With some probability, the agent is moved in a given direction, say downwards, regardless of their actual choice.

In the Jupyter Notebook for Lesson 4, we implement the solution to the problem where there is a 10% chance that the agent goes downwards.

- Because there is a possibility that the agent is forced downwards, states in the upper part of the grid receive lower values, relative to the case without wind.
- Down movements are relatively more valuable now; it is easier to "go with the flow," reducing the scope for multiplicity.

The curse of dimensionality and asynchronous DP

A major drawback to the DP methods analyzed in previous lessons is that they involve operations over the entire state-space: The *curse of dimensionality*.

Asynchronous DP algorithms are iterative algorithms that update the values of states in any order, using whatever values of other states happen to be available.

For convergence, an asynchronous algorithm must still update the values of all the states at some point in the computation.

- A version of asynchronous DP, named "in-place" value iteration, exploits at each iteration whatever new updates of $v^{(1)}(s')$ exist at the time of updating to estimate $v^{(1)}(s)$.
- Another alternative is to update a selection of states that generated the highest updating error in previous steps of the iteration. This is called "Prioritized sweeping".
- Another method, which we will cover in Module 7, is to use "Real-Time DP", which updates the optimization objects according to the experience obtained by the agent in the environment.

Optimal selling time with two assets

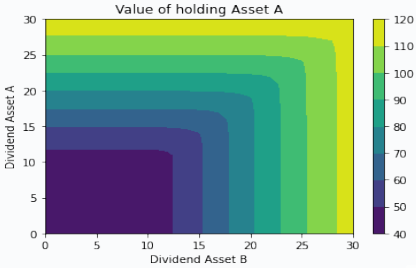
In the final implementation of Module 5, we extend the investor problem we developed in Lesson 1.

The example is meant to illustrate a financial implementation of DP and how asynchronous DP with "in-place" value iteration may help.

The model features an investor holding either of two assets and also having the option of selling the asset they hold and buying the other asset at fixed prices.

DP allows us to determine how much a position is worth for the agent across each state.

The qualitative implications of the model may be trivial—sell when low and buy when high—but the model generates valuations with cross-sectional dependence, even when dividends are independent!



Summary of Lesson 4

In Lesson 4, we have looked at:

- The implementation of DP in the windy gridworld
- Computing optimal policies in the context of the two-asset investor problem with asynchronous DP

⇒ **References for this Lesson:**

Sutton, Richard S., Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 2018. (see Chapters 3 & 4)

TO DO NEXT: Now, please go to the associated Jupyter Notebook for this lesson to analyze the DP implementation of the windy gridworld and the use of asynchronous DP to find the optimal policies in the two-asset investor problem.

In the next module, we will build on the remaining crucial element of RL algorithms: learning from experience in the context of Multi-Armed Bandits.