Lesson 17.

Solving stochastic dynamic programs with Python

Overview

- Let's solve the stochastic dynamic program we formulated for the investment problem in Lesson 16.
- In this class, we will use a package called `stochasticdp` to set up and solve stochastic dynamic programs.
    - *Warning.* This is a package that I wrote. There may be some bugs.
    - *Note.* This package is publicly available. Please feel free to use it in the future for other things. The source code is on GitHub.

Installing `stochasticdp`

- To install `stochasticdp`, open a WinPython Command Prompt and type:

```
pip install stochasticdp
```

- To use `stochasticdp`, we must first import it. In `stochasticdp`, we only need the object `StochasticDP`, so we can perform our import like this:

```
In [2]: from stochasticdp import StochasticDP
```

Setting up a stochastic dynamic program

- Recall the investment problem from Lesson 16:

Problem. Suppose you have $5,000 to invest, and at the beginning of each of the next 3 years, you have an opportunity to invest in either of two investments: A or B. Both investments have uncertain profits. For an investment of $5,000, the profits are as follows:

Investment	Profit ($)	Probability
A	-5,000	0.3
	5,000	0.7
B	0	0.9
	5,000	0.1

You are allowed to make at most one investment each year, and can invest only $5,000 each time. Any additional money accumulated is left idle.

Formulate a stochastic dynamic program to find an investment policy that maximizes the probability you will have $10,000 after 3 years.

- Let's walk through setting up the stochastic DP we formulated in the last lesson.

Initialization

- We had defined 4 stages - to make things easier, let's renumber the stages so they start at $t = 0$:

$$\begin{aligned} \text{stage } t = 0, 1, 2 \quad &\leftrightarrow \quad \text{beginning of year } t \\ t = 3 \quad &\leftrightarrow \quad \text{end of process} \end{aligned}$$

- In each stage, we defined 3 states:

$$\text{state } n \in \{0, 5000, 10000\} \quad \leftrightarrow \quad n \text{ dollars in account}$$

- At each stage and state, we defined 3 possible decisions:

$$\text{decision } x_t \in \{\text{A}, \text{B}, \text{no investment}\}$$

- The set of *allowable* decisions changed, depending on the stage and state. We'll address this later.
- For now, we can initialize a stochastic dynamic program with these stages, states, and decisions like this:

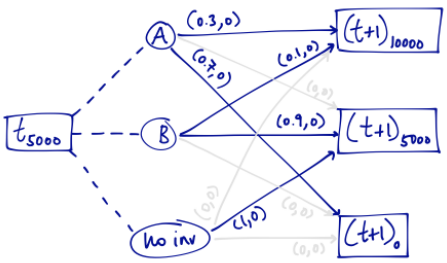
```
In [3]: # Number of stages
        number_of_stages = 4

        # List of states
        states = [0, 5000, 10000]

        # List of decisions
        decisions = ['A', 'B', 'no investment']

        # Initialize stochastic dynamic program - we want to maximize, so minimize = False
        dp = StochasticDP(number_of_stages, states, decisions, minimize=False)
```

- The code above initializes a stochastic dynamic program called `dp`.
- The transition probabilities, contributions, and boundary conditions in `dp` are all initialized to 0.
- We need to change these appropriately.

Transition probabilities and contributions

- First, let's tackle transitions from the state $n = 5000$:

- In the sketch we drew in Lesson 16, we left out the gray edges above, which represent transitions with probability 0.
    - This seems unnecessary, but will become important to consider later.
- Since the transition probabilities are already initialized to 0, we just need to focus on defining the blue edges.



n = 5000

- The transition probability $p(m \mid n, t, x)$ of moving from state n to state m in stage t under decision x is represented by

```
dp.transition[m, n, t, x]
```

- The contribution $c(m \mid n, t, x)$ of moving from state n to state m in stage t under decision x is represented by

```
dp.contribution[m, n, t, x]
```

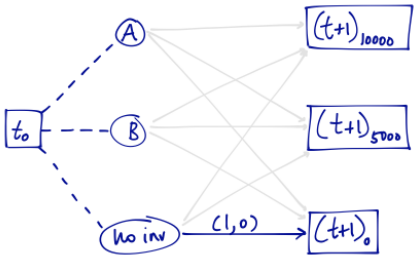
- So, we can input the transition probabilities and contributions from state $n = 5000$ in stages $t = 0, 1, 2$ as follows:

```
In [4]: # Transition probabilities and contributions from state n = 5000
        for t in range(number_of_stages - 1):
            # Investment A
            dp.transition[10000, 5000, t, 'A'] = 0.7
            dp.contribution[10000, 5000, t, 'A'] = 0

            dp.transition[0, 5000, t, 'A'] = 0.3
            dp.contribution[0, 5000, t, 'A'] = 0

            # Investment B
            dp.transition[10000, 5000, t, 'B'] = 0.1
            dp.contribution[10000, 5000, t, 'B'] = 0

            dp.transition[5000, 5000, t, 'B'] = 0.9
            dp.contribution[5000, 5000, t, 'B'] = 0

            # No investment
            dp.transition[5000, 5000, t, 'no investment'] = 1
            dp.contribution[5000, 5000, t, 'no investment'] = 0
```

- Remember that the contributions for all transitions are 0 in this stochastic DP.
- Since the contributions are all set to 0 in the initialization, we actually don't need to define the contributions, like we did above.
- However, we'll continue to do so, for illustration purposes.

- Next, let's tackle the transitions from state $n = 0$. Last time, we sketched these transitions like this:

n = 0



n = 0

- We can revise this sketch to explicitly include all the decisions, even the ones that are not allowable at state $n = 0$:

- The gray edges above represent transitions with probability 0 and contribution 0.
- Note that *all of the edges* coming out of decisions A and B are grey. This represents the fact that A and B are *not allowable* at this stage and state.

- *Quick check.* What can the sum of the transition probabilities from any decision node equal?

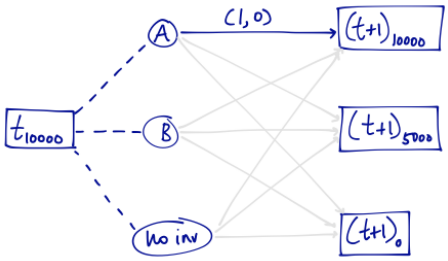
The transition probabilities from any decision node must add up to either 0 or 1. They will add up to 1 if the decision is allowable at that stage/state; otherwise they will add up to 0.

- So, we can input the transition probabilities and contributions from state $n = 0$ in stages $t = 0, 1, 2$ like this:

```
In [5]: # Transition probabilities and contributions from state n = 0
        for t in range(number_of_stages - 1):
            # Investment A - all transitions have probability 0, already done in initialization

            # Investment B - all transitions have probability 0, already done in initialization

            # No investment
            dp.transition[0, 0, t, 'no investment'] = 1
            dp.contribution[0, 0, t, 'no investment'] = 0
```

- We can tackle the transitions from state $n = 10000$ in an almost identical way:

n = 10000



n = 10000

```
In [6]: # Transition probabilities and contributions from state n = 10000
        for t in range(number_of_stages - 1):
            # Investment A - all transitions have probability 0, already done in initialization

            # Investment B - all transitions have probability 0, already done in initialization

            # No investment
            dp.transition[10000, 10000, t, 'no investment'] = 1
            dp.contribution[10000, 10000, t, 'no investment'] = 0
```

Boundary conditions

- Finally, we need to define the boundary conditions.
- In particular, we need to specify the value-to-go function at the last stage (in our case, $t = 3$) for each state.
- The boundary value for state n is represented by:

```
dp.boundary[n]
```

- So, we can input the boundary conditions like this:

```
In [7]: # Boundary conditions
        dp.boundary[0] = 0
        dp.boundary[5000] = 0
        dp.boundary[10000] = 1
```

Solving the stochastic dynamic program

- Once the stochastic DP is setup, we can solve it like this:

```
In [8]: # Solve the stochastic dynamic program
        value, policy = dp.solve()
```

- Note that the method `.solve()` outputs two objects: `value` and `policy`.
- `value[t, n]` is the value-to-go function $f_t(n)$ at stage t and state n.
- `policy[t, n]` is the optimal decision x_t^* that attains the value-to-go function $f_t(n)$ at stage t and state n.
- First, let's see what the value-to-go function looks like:

```
In [9]: # Examine the value-to-go function
        value
```

```
Out[9]: {(0, 0): 0,
         (0, 5000): 0.757,
         (0, 10000): 1,
         (1, 0): 0,
         (1, 5000): 0.73,
         (1, 10000): 1,
         (2, 0): 0,
         (2, 5000): 0.7,
         (2, 10000): 1,
         (3, 0): 0,
         (3, 5000): 0,
         (3, 10000): 1}
```

- Next, let's look at the corresponding policy:

```
In [10]: # Examine the policy
         policy
```

```
Out[10]: {(0, 0): 'no investment',
          (0, 5000): 'B',
          (0, 10000): 'no investment',
          (1, 0): 'no investment',
          (1, 5000): 'B',
          (1, 10000): 'no investment',
          (2, 0): 'no investment',
          (2, 5000): 'A',
          (2, 10000): 'no investment'}
```

On your own

- Solve the stochastic DP we formulated in Lesson 15 for this problem:

Problem. The Hit-and-Miss Manufacturing Company has received an order to supply one item of a particular type. However, manufacturing this item is difficult, and the customer has specified such stringent quality requirements that the company may have to produce more than one item to obtain an item that is acceptable.

The company estimates that each item of this type will be acceptable with probability 1/2 and defective with probability 1/2. Each item costs \$100 to produce, and excess items are worthless. In addition, a setup cost of \$300 must be incurred whenever the production process is setup for this item. The company has time to make no more than 3 production runs, and at most 5 items can be produced in each run. If an acceptable item has not been obtained by the end of the third production run, the manufacturer is in breach of contract and must pay a penalty of \$1600.

The objective is to determine how many items to produce in each production run in order to minimize the total expected cost.

```
In [11]: # Number of stages
         number_of_stages = 4

         # List of states
         states = [0, 1]

         # List of decisions
         decisions = [0, 1, 2, 3, 4, 5]
```

```
# Initialize stochastic dynamic program
dp = StochasticDP(number_of_stages, states, decisions, minimize=True)

# Transition probabilities and contributions from state n = 0
for t in range(number_of_stages - 1):
    for x in decisions:
        dp.transition[1, 0, t, x] = 0
        dp.contribution[1, 0, t, x] = 0

        dp.transition[0, 0, t, x] = 1
        dp.contribution[0, 0, t, x] = 0

# Transition probabilities and contributions from state n = 1
for t in range(number_of_stages - 1):
    for x in decisions:
        if x > 0:
            K = 3
        else:
            K = 0

        dp.transition[0, 1, t, x] = 1 - (1/2)**x
        dp.contribution[0, 1, t, x] = K + x

        dp.transition[1, 1, t, x] = (1/2)**x
        dp.contribution[1, 1, t, x] = K + x

# Boundary conditions
dp.boundary[0] = 0
dp.boundary[1] = 16

# Solve the stochastic dynamic program
value, policy = dp.solve()
```

```
In [12]: # Examine value-to-go
         value
```

```
Out[12]: {(0, 0): 0,
          (0, 1): 6.75,
          (1, 0): 0,
          (1, 1): 7.0,
          (2, 0): 0,
          (2, 1): 8.0,
          (3, 0): 0,
          (3, 1): 16}
```

```
In [13]: # Examine policy
         policy
```

```
Out[13]: {(0, 0): 0, (0, 1): 2, (1, 0): 0, (1, 1): 2, (2, 0): 0, (2, 1): 3}
```