Lab 1

Introduction to Python Programming for Economics & Finance

Richard Foltyn
University of Glasgow

May 23, 2023

Contents

1 Two-period consumption-savings problem

Consider the following standard consumption-savings problem over two periods with lifetime utility $U(c_1, c_2)$ given by

$$\begin{aligned} \max_{c_1, c_2} \quad & \frac{c_1^{1-\gamma}}{1-\gamma} + \beta \frac{c_2^{1-\gamma}}{1-\gamma} \\ \text{s.t.} \quad & c_1 + \frac{c_2}{1+r} = w \\ & c_1 \geq 0,\ c_2 \geq 0 \end{aligned}$$

where γ is the RRA coefficient, β is the discount factor, r is the interest rate, w is initial wealth, and (c_1, c_2) is the optimal consumption allocation to be determined.

2 Per-period utility function

Write a function `util(c, gamma)` which takes as arguments the consumption c and the risk-aversion γ and returns the per-period utility given by $u(c) = \frac{c^{1-\gamma}}{1-\gamma}$. Make sure that the function works with log preferences ($\gamma = 1$) as well as general CRRA preferences with $\gamma \neq 1$.

Hint: Use `np.log()` from the NumPy package to evaluate logs.

3 Solving the problem using grid search

In a first step, you are going to solve the household problem using grid search, a basic algorithm that evaluates the objective function (lifetime utility) for every possible value of (c_1, c_2) on a grid of candidate consumption levels.

3.1 Objective function (lifetime utility)

Write the objective function `objective(c1, c2, beta, gamma)` which takes the candidate consumption choices and parameters as arguments and returns the associated lifetime utility. This function should call the per-period utility function `util(c, gamma)` you wrote above.

3.2 Candidate consumption grid

Assume that the problem is parametrised using the following values:

[3]:
```
# Parameters
r = 0.04
beta = 0.96
gamma = 1.0

# Initial wealth
wealth = 1.0
```

Create a uniformly spaced grid for candidate period-1 consumption levels c_1 called `c1_grid` with 20 points. What is the interval of feasible values for c_1?

Hint: Use `np.linspace()` to create a linearly (uniformly) spaced array.

3.3 Grid search algorithm

Write a function `find_optimum(c1_grid, beta, gamma, r, wealth)` which takes as arguments the candidate first-period consumption levels as well as parameters and returns the following objects:

1. An array containing the candidate lifetime utility for each candidate c_1.
2. An integer index identifying the maximizer on that array.

Use this function to find the optimal consumption levels.

Hint: Loop over the points in `c1_grid`. For each c_1 on the grid, use the budget constraint to obtain the implied c_2. Then use the `objective(c1, c2, ...)` function to evaluate lifetime utility and keep track of the maximum.

3.4 Reporting the results

Write a loop that prints the allocation (c_1, c_2) and the implied utility level $U(c_1, c_2)$ for each point on the candidate grid.

4 Analytical solution

Compute the analytical solution for this problem and contrast it with what you found above. Why are the values not identical?

Hint: The analytical solution can be trivially derived from the first-order conditions given by the Euler equation and the budget constraint:

$$c_1^{-\gamma} = \beta(1+r)c_2^{-\gamma}$$
$$c_1 + \frac{c_2}{1+r} = w$$

The optimal c_1 is then given by

$$c_1 = \alpha \cdot w \qquad \text{with} \qquad \alpha = \left[1 + \beta^{\frac{1}{\gamma}}(1+r)^{\frac{1}{\gamma}-1}\right]^{-1}$$

where α is the fraction of initial wealth consumption in period 1.

```
Utility at optimal c1=0.5102, c2=0.5094: -1.32051e+00
```

The values are unlikely to be identical to the result we found using grid search since it is unlikely that the optimal c_1 was included in the grid. We can of course get closer to the analytical value by increasing the grid size which is trivial in this setting but might not be computationally feasible in a more realistic problem.

5 Solving the problem using a minimizer

We usually don't use grid search to find an optimum since the method is often slow and imprecise. In this part you will therefore explore how the optimal allocation can be found using SciPy's optimization routines.

5.1 Objective function

Since this is a scalar maximization problem (in either c_1 or c_2 as the other is implied by the budget constraint), you first need to write a modified objective function that is compatible with SciPy's optimisation routines. To this end, define a function `objective_scipy(c1, beta, gamma, r, wealth)` which takes c_1 and parameters as arguments and evaluates lifetime utility. Because we are going to run a **minimizer** implemented in `minimize_scalar()` to find the optimum, your objective function needs to return the **negative** lifetime utility.

Hint: Use the function `util(c, gamma)` you implemented previously to evaluate the objective function.

5.2 Running the minimizer

To run the scalar minimizer, you need to import the function `minimize_scalar` as follows:

```
from scipy.optimize import minimize_scalar
```

In this particular case, `minimize_scalar()` takes the following arguments:

1. The objective function
2. The `method` parameter specifying the minimization algorithm (use `method='bounded'`)
3. The `bounds` parameter specifying the range of admissible values.

Note that because your objective function takes 4 arguments but SciPy expects a function with a single argument, you'll need to use a `lambda` function to pass any additional arguments.

Write the call to `minimize_scalar()` as follows:

```
res = minimize_scalar(..., method='bounded', bounds=... )
```

Alternatively, you can skip the `lambda` function and pass any additional parameters as a `tuple` using the `args` argument as follows:

```
res = minimize_scalar(..., method='bounded', bounds=..., args=...)
```

Run the minimizer and inspect the attributes of the resulting object `res`. You'll see that the maximizer is returned as `res.x` and the objective is stored in `res.fun`.

5.3 Reporting the results

Print the optimal allocation (c_1, c_2) and the associated lifetime utility obtained by the minimizer. Does it differ from the result you found with grid search?