647 Final Project: Solving Water Allocation Problem using Dynamic Programming with Bayesian Updating

Mahnoor Babar (261032538)

15th May 2022

Abstract

This project sets out to create a model for Water Allocation that enables a social planner to allocate water from a reservoir between a farmer and tourists for irrigation and recreational uses respectively. The result is a maximisation problem with no closed form solution which therefore needs to be solved numerically through computational methods. This project employs the dynamic programming technique and uses policy iteration to allow the planner to decide whether water from the reservoir is to be given to the farmer only - who is the priority - or whether tourists should also be given water. The decision is based on the amount of water already in the dam as well as the planner's assessment (or expectation) of rain based on rainfall observed in the previous period. This is included in the solution using a Bayesian updating routine.

Objective

The aim behind this project is to use dynamic programming with policy iteration to solve the planner's problem to allocate water from a dam between a farmer for agricultural use and tourists for recreational needs. The planner is constrained by the level of water already in the dam as well as the prospective amount of rainfall over the period. Water in the dam is replenished by rain which is uncertain for the most part however, the planner can analyse rainfall from the previous period and update his beliefs about rain in the upcoming period using Bayes Theorem. The Bayesian approach to probability is different from that of the more ubiquitous frequentist approach as it considers probability as the 'degree of belief in an event'. This enables the planner to make more appropriate allocation decisions by maximising the utility of both the farmer and tourists who use water from a reservoir for their individual needs.

Literature Review

Dynamic Programming (DP) has been widely used for water management problems. Keckler and Larson (1968) use iteration in policy space in their DP routine and apply in to a range of water management problems over short, medium and long run. This is probably the first application of the DP technique to a water allocation problem. Similarly, Tejada-Guibert, Johnson and Stedinger (2003) use two different approaches for implementing reservoir operation policies using stochastic dynamic programming. The water release decisions are made by interpolating policy tables or by re-optimizing policy within simulation. Moreover, Luo, Maqsood and Huang (2007) address a water systems planning problem in an uncertain environment using stochastic dynamic programming. The uncertainty allows incorporation of different allocations based on whether they are facing a dry or a regular rain season.

The literature quoted above yields legitimacy to my choice of technique applied to water allocation. However, I take this a step further by making the problem a bit more realistic by incorporating the uncertainty in rainfall by using Bayesian updating of the probability of a given distribution of rainfall. Although first introduced in 1763 by Thomas Bayes, Bayesian probability has resurfaced in the recent years and has been used in a wide array of applications including applications in economics. However, to the best of my knowledge this kind of analysis has not been conducted for water management topics.

Therefore, it is not only timely to apply this to the current topic but thinking about rainfall from a Bayesian perspective also makes a lot of sense.

Theory

Dynamic Programming

Dynamic Programming (DP) is an algorithmic technique whereby an optimisation problem is solved for by breaking it down into simpler sub-problems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its sub-problems.

The present case is a discrete time continuous state Markov Decision model such that in the current time period t, the planner observes the state of a process, takes an action and earns a reward dependant on both both the state and the action. In particular, the state follows a controlled Markov Probability Law, and the state the following period depends on the state in the current period and an exogenous random shock that is unknown and is assumed independantly and identically distributed (iid).

The planner is looking for a policy of state contingent actions that maximizes the present value of current and expected future rewards, discounted at a per-period factor δ.

These problems can be analysed using DP based on Bellman's Principle of Optimality which yields a recursive functional equation known as the value function.[1]

Value Function Iteration

The Value function iteration method also features in this problem. In general, this technique is good for using a fixed-point but can also be applied to a root-finding problem if it is re-stated as a fixed-point problem.

[1] Miranda and Fackler, "Applied Computational Economics and Finance". p.219

The iteration can be initiated by by supplying a guess $x^{(0)}$ for a fixed point g. Then, subsequent iterations are generated using the simple rule

$$x^{(k+1)} \leftarrow g(x^{(k)})$$

Because g is a continuous function, if the iterations converge, the convergence is on a fixed point of g if g is differentiable and if the initial value of x provided is close enough to the fixed point x^* of g at which $\|g'x^*\|$[2]

Bayes Theorem

According to Bayes Theorem, the probability of an event could be based on prior knowledge of conditions that might be related to the event. With Bayesian probability interpretation, the theorem expresses how a degree of belief, expressed as a probability, should rationally change to account for the availability of related evidence. Bayesian inference is fundamental to Bayesian statistics.

Bayes Theorem can be expressed as follows:

$$\text{P(A|B)} = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events such that P(B)≠0

Moreover, the theorem and its implications can be better understood by statements and definitions which can be summarised as follows:

* P(A|B) is a conditional probability: probability of event A occurring given event B is true. This is called posterior probability of A given B.

* P(B|A) is also a conditional probability: probability of event B occurring given event A is true. This is called the likelihood of A given a fixed B as P(B|A) = L(A|B).

* P(A) and P(B) are the probabilities of observing A and B respectively, without any given conditions. This is also known as marginal or prior probability.

* A and B must be different events.

Setting up the Problem

I took the idea for this project from the examples given in the book [3] and then added complexity to the problem based on what I think is closer to a possible real-life scenario as well as both theoretical ideas and computational methods that I have been finding interesting lately. This process resulted in a problem which is similar to the book only in terms of its basic idea but ends up being considerably different in terms of the final product. This also means I do not borrow from the book or toolbox at all in terms of the code provided and did my best to solve the problem. The version of the problem I solve for can be summarised as follows:

"Water from a dam can be used for either irrigation or recreation. Irrigation benefits farmers but damages recreational users. The benefits accrued by farmers and tourists are $F(s_t)$ and $G(s_t, a_t)$ respectively where s_t is the total water in the dam (or the state) and a_t is the amount of water used by the farmer (or the action space). The difference between the two is what remains for recreational purpose. The water level is replenished by rainfall that falls in one of two distributions: low or high. With probability π rainfall in the current period falls in the low distribution. This is important as it gives the planner a signal to how much rain to expect in the upcoming period and plan accordingly using Bayesian updating.

[2] Miranda and Fackler, "Applied Computational Economics and Finance". p.33

[3] *Ibid.*, p.186

This program derives an irrigation flow policy to maximise the benefit to both farmer and tourists over an infinite time horizon."

To lay out the differences from the model sketched in the book clearly: I work with continuous state and action space instead of simply using a discrete space; and I make the model a lot more realistic by considering two distributions of rainfall and updating the planner's beliefs of rain in the upcoming period based on rainfall in the on-going period, where the probability is also variable and not fixed. Whereas the book considers both a fixed probability and a fixed (one unit of) rain or none at all.

The model is as follows:

The State Space is

s = units of water in the dam at the beginning of the year

$$\text{s} \in \text{S} = [0,1000]$$

The Action is

a = units of water released for irrigation during the year

$$\text{a} \in \text{A} = [0,1000]$$

Because I chose to complicate the problem, another state would be the probability of observing rain from the 'Low' distribution, so

$$\pi = [0,1]$$

The state transition probability rule is then given by a range of probabilities from 0 to 1, such that r' is the expected level of rain that varies according to the expected probability π. The state transition is given by the integral of the minimum amount between the total water in the dam minus what the farmer uses plus the expected rain, and the maximum capacity of the dam (given by M)

$$\text{P(s'| s,a)} = \int_0^1 min(s - x + r'), M$$

The Reward Function maximises the utility of both the farmer and the tourists:

$$\text{f(s,a)} = \text{F(a)} + \text{G(s-a)}$$

The Value Function is:

V(s) = Value of s units of water in dam at beginning of year t

$$v(s,\pi) = maxf(s,x) + \delta \int_0^1 \pi vhat(min(s - x - r'), M)ds'$$

Or more generally as:

$$v(s,\pi) = maxf(s,a) + \delta \int_0^1 \{s', \omega, \pi q_\pi(s')ds'$$

The updating rule for the probability 'Low' rain distribution is:

$$\pi_{t+1} = \frac{\pi_t l(s_{t+1})}{\pi_t l(s_{t+1}) + (1-\pi_t)h(s_{t+1}}$$

Hence, we use the fact that the equation above is recursive allows us to use a recursive solution method such that:

$$q_\pi(s) = \pi l(s) + (1 - \pi)h(s)$$

The equation above allows us to summarise the planner's ignorance regarding which of the two rainfall distributions was chosen (by nature) in the previous time period. Whereas 'Learning' by the planner regarding the probability of observing rain from the 'Low' distribution in the next period is given by:

$$\kappa(s,\pi) = \frac{\pi l(s)}{\pi l(s) + (1+\pi)h(s)}$$

Coding Explanation

The code file first creates the two distributions of rain that 'nature' could have chosen from at time period t=-1 by creating a grid from 0 to 1 with 100 intervals in between. The 'Low' distribution labeled l is created with a mean of 10 and variance of 4, whereas the

'High' distribution labeled h is created with a mean of 100 and variance of 25. Figure 1 illustrates the two different distributions of rain.

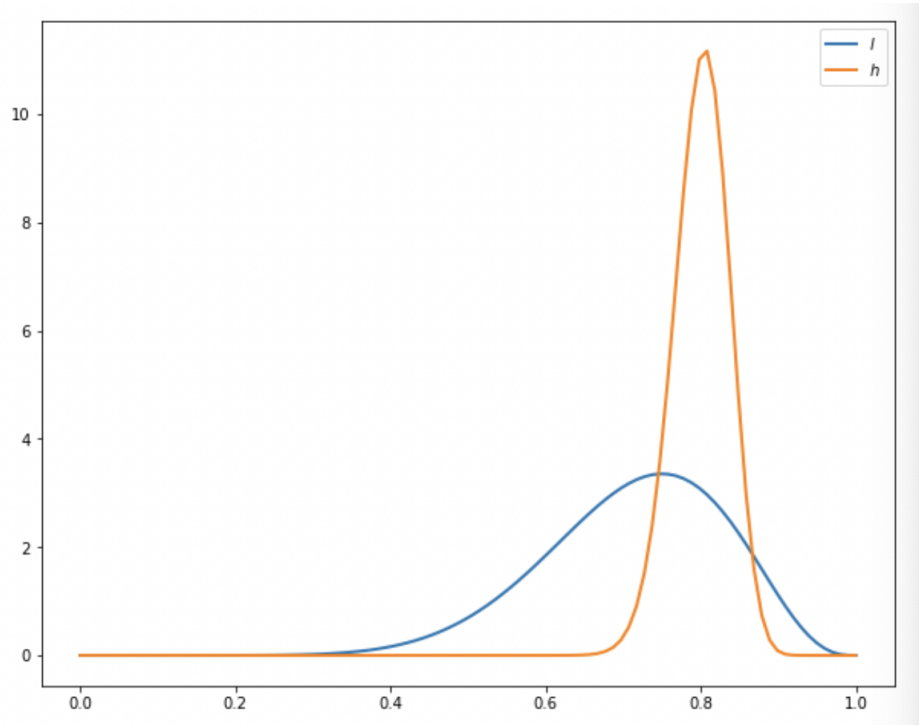

Figure 1: Figure 1: Two distributions of Rainfall

Next, parameters are created for the two distributions and stored in a class called Water Allocation. The in-built function init then stores the parameters δ or the discount factor, the mean and variance for the low and high rainfall distributions. S indicates the state space with is proportion of water in the reservoir, π is the probability that the 'Low' distribution is chosen. This is another state that the planner may find himself in. Both of these are continuous and so are discretized using a grid. Similarly, x indicates the water used by the farmer and so is the continuous action space. The program then randomly draws from the two distributions.

The updating rule for the $\pi_{t+1} = \frac{\pi_t l(s_{t+1})}{\pi_t l(s_{t+1})+(1-\pi_t)h(s_{t+1}}$

Hence, we use the fact that the equation above is recursive allows us to use a recursive solution method such that the function ω solves for the minimum water required in the dam for the planner to be indifferent between providing water to the farmer only or both the farmer and the tourists.

The ignorance of the planner regarding the distribution of rain is given as follows: $q_\pi(s) = \pi l(s) + (1 - \pi)h(s)$ Our planner has to decide on the optimal use of water given that he observes rainfall in the previous period, however he does not observe which distribution has been chosen. Therefore, he summarises his ignorance in the form of a joint probability distribution that gives him the expected amount of rainfall as well as its variance at a given time period, which is computed by the program.

Where updating rule for the probability above is given as follows: $\kappa(s,\pi) = \frac{\pi l(s)}{\pi l(s)+(1+\pi)h(s)}$

κ updates π using Bayes' rule and the current level of water in the dam s.

The function ω uses interpolation to update the minimum probability equation using the operator Q. Using the probability and omega function

The function Q updates the miniumum water required (or state s) equation.

Then, we solve for sbar which is the level of water in the dam when the planner is indifferent between giving water only to the farmer versus giving water to both the farmer and the tourists. This is done by setting up a loop and then initialising s.

Finally, Figure 3 plots the resulting policy function that will enable the planner to decide upon the allocation of water between the farmer and the tourist based on the initial level of water as well as the probability of the 'Low' rain distribution. As seem from the resulting graph, increasing probability of low rain means the initial level of water in the dam needs to be higher for the planner to allow tourists to use the water. Intuitively this makes a lot of sense.

```
Error at iteration 5 is 0.018876382309432893.
Error at iteration 10 is 7.187965262389628e-05.

Converged in 10 iterations.
```

Figure 2: Iterations

The coding of this program was done in Python using the packages quantecon (version 0.5.3), glyphsLib (version 2.4.0) and interpolation (version 2.2.1). The libraries used are cm from matplotlib, njit from numba, mlinterp from interpolation, gamma from math, numpy, cumfreq and beta from scipy.stats, and scipy.optimize. The code can also be viewed here https://github.com/MahnoorBabar/647-Final-Project

Conclusion

The program created allows the social planner to allocate water from the reservoir between a farmer and tourists using dynamic programming with policy iteration to solve a maximisation problem that has no closed form solution. The techniques used turn out to be an efficient way to solve the problem and achieve our objective. Going forward, I would like to be able to add to this problem and complicate it further to mimic a possible real-life scenario. For instance, I would like to be able to say how much water should be given to both the farmer and the tourist instead of just stating if it should be given to the tourist.

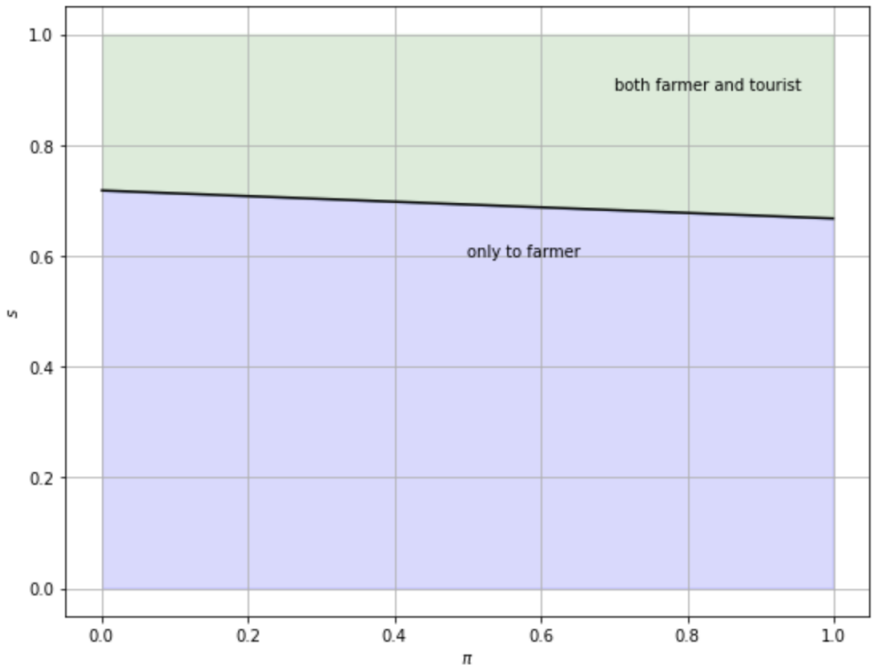

Figure 3: Allocation of Water

References

Keckler, William G. and Larson, Robert E. "Dynamic Programming Applications to the Water Resources System Operation and Planning". *Journal of Mathematical Analysis and Applications*. Volume 24 (1968): 80-109.

Tejada-Guibert, J. Alberto. Johnson, Sharon A. and Stedinger, Jery R. "Comparison of Two Approaches for Implementing Multireservoir Operating Policies Derived Using Stochastic Dynamic Programming". *Water Resources Research*. Volume 29, No. 12 (1993): 3969-3980.

Luo, B. Maqsood, I. and Huang, G. H. "Planning Water Resources systems with Interval Stochastic Dynamic Programming". *Water Resources Research*. Volume 21 (2007): 997-1014.

Miranda, Mario J. and Fackler, Paul l. "Applied Computational Economics and Finance". *The MIT Press*.