Inventory Control

Michel De Lara

October 10, 2017

Contents

Let time t be measured in discrete units (such as days, weeks or months). Consider the *inventory problem*

$$x(t+1) = x(t) + u(t) - w(t)\,, \quad u(t) \geq 0 \tag{1}$$

where

- time $t \in \{t_0, \ldots, T\}$ is discrete,
- $x(t)$ is the *stock* at the beginning of period t, belonging to $\mathbb{X} = \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$,
- $u(t)$ is the *stock ordered* at the beginning of period t, belonging to $\mathbb{U} = \mathbb{N} = \{0, 1, 2, \ldots\}$,
- $w(t)$ is the uncertain *demand* during the period t, belonging to $\mathbb{W} = \mathbb{N}$.

When $x(t) < 0$, this corresponds to a *backlogged demand*, supposed to be filled immediately once inventory is again available.

1 Expected costs minimization

(Bertsekas, 2000)

The manager problem is one of *cost minimization* where

- ordering stock has unitary *purchasing cost* c,
- unitary *shortage cost* for unfilled demand is b,
- unitary *holding cost* for excess inventory is h.

The costs incurred in period t are

$$\underbrace{cu(t)}_{purchasing} + \underbrace{b\max\{0,-\big(x(t)+u(t)-w(t)\big)\}}_{shortage} + \underbrace{h\max\{0,x(t)+u(t)-w(t)\}}_{holding} . \tag{2}$$

For simplicity, we shall denote

$$\mathcal{R}(x) := b\max\{0,-x\} + h\max\{0,x\} . \tag{3}$$

On the period from t_0 to T, the costs sum up to

$$\sum_{t=t_0}^{T-1}[cu(t) + \mathcal{R}\big(x(t)+u(t)-w(t)\big)] . \tag{4}$$

We shall suppose that unitary costs are ranked as follows:

$$b > c > 0 .$$

We suppose that $w(t)$, the uncertain demand, is a random variable with distribution $p_0, \ldots, p_N$ on the set $\{0, \ldots, N\}$:

$$\mathbb{P}\{w(t)=0\} = p_0, \ldots, \mathbb{P}\{w(t)=N\} = p_N . \tag{5}$$

We also suppose that the random variables

$$w(\cdot) = \big(w(t_0), \ldots, w(T-1)\big) \tag{6}$$

are independent.

A decision rule $\mathfrak{u} : \mathbb{N} \times \mathbb{X} \to \mathbb{U}$ assigns a (stock) order $u = \mathfrak{u}(t,x)$ to any state x of inventory stock and to any period t. Once given, we obtain random trajectories

$$\begin{array}{rcl} x(t_0) & = & x_0 \\ x(t+1) & = & x(t) + u(t) - w(t) \\ u(t) & = & \mathfrak{u}\big(t, x(t)\big) \end{array}$$

and thus random costs

$$\mathsf{Crit}^{\mathfrak{u}}\big(t_0, x_0, w(\cdot)\big) := \sum_{t=t_0}^{T-1} cu(t) + \mathcal{R}\big(x(t)+u(t)-w(t)\big) . \tag{7}$$

The expected costs are

$$\mathbb{E}\Big[\mathsf{Crit}^{\mathfrak{u}}\big(t_0, x_0, w(\cdot)\big)\Big] = \mathbb{E}\big[\sum_{t=t_0}^{T-1} cu(t) + \mathcal{R}\big(x(t)+u(t)-w(t)\big)\big] . \tag{8}$$

The dynamic programming equation associated to the problem of *minimizing the expected costs* is

$$\left\{\begin{array}{rcl} V(T,x) & = & \underbrace{0}_{final\ cost}\ , \\ V(t,x) & = & \inf\limits_{u\geq 0}\mathbb{E}\big[\Big(\underbrace{cu+\mathcal{R}\big(x+u-w\big)}_{instantaneous\ cost}+V\big(t+1,\underbrace{x+u-w}_{future\ stock}\big)\Big)\big]\ , \end{array}\right. \tag{9}$$

where w is a random variable with the distribution $p_0,\ldots,p_N$ on the set $\{0,\ldots,N\}$.

We have that

$$V(T-1,x)=\inf_{u\geq 0}\mathbb{E}\big[cu+\mathcal{R}\big(x+u-w\big)\big]=\inf_{u\geq 0}[cu+\sum_{i=0}^{N}p_i\mathcal{R}\big(x+u-i\big)]\ . \tag{10}$$

Question 1 *Recalling that $b>c>0$, show that $g(z)=\mathbb{E}\big[cz+\mathcal{R}\big(z-w\big)\big]$ is a convex function with a minimum achieved at some $S_{T-1}\in\{0,\ldots,N\}$. You can draw a graph of the above function for $N=1$ and $p_0=1/2$.*

Observe that $\mathbb{E}\big[cu+\mathcal{R}\big(x+u-w\big)\big]=-cx+g(x+u)$, and deduce that the optimal rule is

$$\mathfrak{u}(T-1,x)=\left\{\begin{array}{rll} S_{T-1}-x & if & x<S_{T-1} \\ 0 & if & x\geq S_{T-1}\ . \end{array}\right. \tag{11}$$

Interpret this rule, sometimes called base-stock policy*.*

Question 2 *Show by induction that there exist thresholds $S_{t_0},\ldots,S_{T-1}$ in $\{0,\ldots,N\}$ such that the optimal rule at period t is*

$$\mathfrak{u}(t,x)=\left\{\begin{array}{rll} S_t-x & if & x<S_t \\ 0 & if & x\geq S_t\ . \end{array}\right. \tag{12}$$

We know will make numerical simulations, and try different rules.

Question 3 *Sample one random sequence of $T-t_0$ demands with the macro* **grand***. Picture the trajectories of the stocks corresponding to the constant decision rule $\mathfrak{u}(t,x)=2$. Evaluate the costs as in (4).*

```
// exec inventory_control.sce

// Costs functions parameters
purchasing=100;
shortage=150;
holding=20;

// Shortage/Holding costs
function c=SHcosts(zz)
```

```
  c=shortage*maxi(-zz,0)+holding*maxi(zz,0);
endfunction

// Instantaneous costs function
function c=instant_costs(xx,uu,ww)
  c=purchasing*uu+SHcosts(xx+uu-ww);
endfunction

// Decision rule
function u=constant_rule(t,x)
  u=2;
endfunction

// Trajectories simulations

function [XX,UU,CC]=trajectories(simulations,rule)
  XX=[];
  UU=[];
  CC=[];
  WW=grand(horizon,'markov',transition,ones(1,simulations));

  for s=1:simulations do
    xx=0;
    uu=[];
    cc=0;
    ww=WW(s,:);
    for t=0:(horizon-1) do
      uu=[uu,rule(t,xx($))];
      xx=[xx,xx($)+uu($)-ww(t+1)];
      cc=cc+instant_costs(xx($),uu($),ww(t+1));
    end
    //  plot2d([0:horizon],xx)
    XX=[XX;xx];
    UU=[UU;uu];
    CC=[CC;cc];
  end
  //
  disp('expected costs are '+string(mean(CC)));
endfunction

horizon=12;
proba=0.01*[40,20,30,10];
```

```
// simulation of iid sequences of length horizon
transition=ones(proba')*proba;

// Number of Monte Carlo simulations
simulations=1;
simulations=100;
simulations=10;

// Trajectories simulations and visualization
xset("window",21);xbasc();
[XX,UU,CC]=trajectories(simulations,constant_rule);
plot2d(0:horizon,XX')

constant_rule_costs=mean(CC)
```

Question 4 *Draw 100 random sequences of $T - t_0$ demands. Picture the trajectories of the stocks corresponding to the constant decision rules $\mathfrak{u}_k(t, x) = k$, for $k = 0, 1, 2, 3$. What do you observe? Evaluate the expected costs by Monte Carlo. What is the optimal rule among the constant decision rules?*

```
// number of Monte Carlo simulations
simulations=1;
simulations=100;

[XX,UU,CC]=trajectories(simulations,constant_rule);
plot2d(0:horizon,XX')

constant_rule_costs=mean(CC)
```

Question 5 *Picture the trajectories of the stocks corresponding to the following optimal decision rule. Compare these trajectories with those obtained with the constant decision rules above. What do you observe? Evaluate the expected costs by Monte Carlo. Compare with the expected costs given by the best constant decision rule.*

```
////////////////////////////////////////////////////////////////////
//    STOCHASTIC DYNAMIC PROGRAMMING EQUATION
////////////////////////////////////////////////////////////////////

function [FEEDBACK,state_min]=SDP()
  states=[(mini(controls)-maxi(demands))*(horizon): ...
          (maxi(controls)-mini(demands))*(horizon)];
  // Due to bounds on controls and demands, the state is bounded a priori.
```

```
cardinal_states=prod(size(states));
cardinal_controls=prod(size(controls));
cardinal_demands=prod(size(demands));

state_min=mini(states);
state_max=maxi(states);

function i=index(x)
  i=x-state_min+1;
endfunction

function xdot=dynamics(x,u,w)
  xdot=maxi(state_min,mini(state_max,x+u-w));
endfunction

VALUE=zeros([0:horizon]'*states);
FEEDBACK=zeros([0:horizon]'*states);

VALUE(horizon+1,:)=zeros(states);

shift=1+[(horizon-1):(-1):1];
for tt=shift do
  loc=zeros(cardinal_controls,cardinal_states);
  // local variable containg the values of the function to be minimized
  for jj=1:cardinal_controls do
    uu=controls(jj);
    loc(jj,:)=0;
    // the following loop computes an expectation
    for dd=1:cardinal_demands do
      ww=demands(dd);
      loc(jj,:)=loc(jj,:)+ ...
                proba(dd)* ...
                (instant_costs(states,uu,ww)+ ...
                 VALUE(tt+1,index(dynamics(states,uu,ww))));
    end;
  end
  //
  [mmn,jjn]=mini(loc,'r');
  [mmx,jjx]=maxi(loc,'r');
  // mm is the extremum achieved
  // jj is the index of the extremum argument
```

```
    //
    VALUE(tt,:)=mmn;
    // minimal cost
    FEEDBACK(tt,:)=controls(jjn);
    // optimal feedback
  end
endfunction


horizon=12;
proba=0.01*[40,20,30,10];
transition=ones(proba')*proba;
demands=[0:(prod(size(proba))-1)];
controls=[0:2*(prod(size(proba)))];

[FEEDBACK,state_min]=SDP();

// Decision rule
function u=optimal_rule(t,x)
  tt=t+1;
  u=FEEDBACK(tt,x-state_min+1);
endfunction
//



// Trajectories simulations and visualization
xset("window",22);xbasc();
[XX,UU,CC]=trajectories(simulations,optimal_rule);
plot2d([0:horizon],XX')
xtitle("Stock trajectories (minimal cost)");

optimal_costs=mean(CC)
```

Question 6 *Increase the unitary holding cost h and decrease the unitary shortage cost b. What do you observe?*

Question 7 *Decrease the horizon T to $T = 4$ and the maximal demand to 2. Examine the matrix* ***FEEDBACK****. Show that the optimal rule is the form* (12)*. What are the values of the thresholds $S_{t_0}, \ldots, S_{T-1}$?*

```
horizon=4;
proba=0.01*[40,20,40];
```

```
demands=[0:(prod(size(proba))-1)];
controls=[0:10];

[FEEDBACK,state_min]=SDP();
```

Figure 1: Optimal stock trajectories

Question 8 *Suppose now that there is a fixed cost $K > 0$ for ordering stock. Simulate optimal trajectories of the stocks and of the controls. What do you observe for the controls trajectories?*

```
K=30;

// Instantaneous costs function
function c=instant_costs(xx,uu,ww)
  c=K*sign(uu)+purchasing*uu+SHcosts(xx+uu-ww);
endfunction
```

2 Stochastic viability

(De Lara and Doyen, 2008)

Now, the manager aims at *maximizing the probability*

$$\mathbb{P}\{x(t) \geq x^\flat \ , \ cu(t) + h \max\{0, x(t) + u(t) - w(t)\} \leq C^\sharp \ , \quad \forall t = t_0, \ldots, T-1\} \tag{13}$$

that stocks are above a critical level $x^\flat$ (to limit unsatisfied clients) and that purchasing and holding costs are bounded above by $C^\sharp$.

The dynamic programming equation associated is

$$\left\{\begin{array}{rcl} V(T,x) & = & 1 \\ V(t,x) & = & \sup_{u\geq 0} \mathbb{E}\left[\mathbf{1}_{\{x\geq x^\flat\}}\mathbf{1}_{\{cu+h\max\{0,x+u-w\}\leq C^\sharp\}} \times V\big(t+1,x+u-w\big)\right], \end{array}\right. \tag{14}$$

where w is a random variable with the distribution $p_0,\ldots,p_N$ on the set $\{0,\ldots,N\}$.

```
////////////////////////////////////////////////////////////////////////
//    STOCHASTIC VIABILITY DYNAMIC PROGRAMMING EQUATION
////////////////////////////////////////////////////////////////////////

// The following code contains a bug:
// it produces probability value functions larger than 1.
//

function [FEEDBACK,state_min]=SVSDP()
  states=[(mini(controls)-maxi(demands))*(horizon): ...
          (maxi(controls)-mini(demands))*(horizon)];

  cardinal_states=prod(size(states));
  cardinal_controls=prod(size(controls));
  cardinal_demands=prod(size(demands));

  state_min=mini(states);
  state_max=maxi(states);

  function i=index(x)
    i=x-state_min+1;
  endfunction

  function xdot=dynamics(x,u,w)
    xdot=maxi(state_min,mini(state_max,x+u-w));
  endfunction

  VALUE=zeros([0:horizon]'*states);
  FEEDBACK=zeros([0:horizon]'*states);

  VALUE(horizon+1,:)=ones(states);

  shift=1+[(horizon-1):(-1):1];
```

```
  for tt=shift do
    loc=zeros(cardinal_controls,cardinal_states);
    // local variable containg the values of the function to be maximized
    for jj=1:cardinal_controls do
      uu=controls(jj);
      loc(jj,:)=0;
      // the following loop computes an expectation
      for dd=1:cardinal_demands do
        ww=demands(dd);
        loc(jj,:)=loc(jj,:)+ ...
                  proba(dd)* ...
                  (bool2s(states >= stockflat) .* ...
                   bool2s((purchasing*uu+holding*maxi(states+uu-ww,0)) <= costsharp) .*
                   VALUE(tt+1,index(dynamics(states,uu,ww))));
      end;
    end
    //
    [mmn,jjn]=mini(loc,'r');
    [mmx,jjx]=maxi(loc,'r');
    // mm is the extremum achieved
    // jj is the index of the extremum argument
    //
    VALUE(tt,:)=mmx;
    // maximal probability
    FEEDBACK(tt,:)=controls(jjx);
    // optimal feedback
  end
  //
endfunction


horizon=12;
proba=0.01*[40,20,30,10];
transition=ones(proba')*proba;
demands=[0:(prod(size(proba))-1)];
controls=[0:2*(prod(size(proba)))];

stockflat=-2;
costsharp=1000;

[FEEDBACK,state_min]=SVSDP();
```

```
// Decision rule
function u=optimal_viable_rule(t,x)
  tt=t+1;
  u=FEEDBACK(tt,x-state_min+1);
endfunction
//


// Trajectories simulations and visualization
xset("window",23);xbasc();
[XX,UU,CC]=trajectories(simulations,optimal_viable_rule);
plot2d([0:horizon],XX')
xtitle("Stock trajectories (maximal viability probability)");

// Take care: CC is not the right criterion
// The additive criterion should be replaced by a multiplicative one
```

References

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusets, second edition, 2000. Volumes 1 and 2.

M. De Lara and L. Doyen. *Sustainable Management of Natural Resources. Mathematical Models and Methods*. Springer-Verlag, Berlin, 2008.