‘MATH+ECON+CODE’ MASTERCLASS ON MATCHING MODELS, OPTIMAL TRANSPORT AND APPLICATIONS

Alfred Galichon (New York University)

Spring 2018
Day 1, January 15 2018: Linear programming
Block 3. Dynamic programming

Learning objectives: block 3

- Basics of (finite-horizon, discrete) dynamic programming: Bellman's equation; forward induction, backward induction
- Dynamic programming as linear programming: interpretation of duality
- Vectorization, Kronecker products, multidimensional arrays

References for block 3

- Bertsekas, D. (2011), *Dynamic Programming and Optimal Control* Vols. I and II. 3rd ed. Athena.
- Ljungqvist, Sargent (2012), *Recursive Macroeconomic Theory* 3rd ed. MIT.
- Rust (1987), "Optimal replacement of GMC bus engines: an empirical model of Harold Zurcher. *Econometrica*.

Section 1

THEORY

Motivation: dynamic programming

- John Rust describes the problem of Harold Zurcher, an engineer who runs a bus fleet as follows:
  - each month, buses operate a stochastic number of miles
  - operations costs increase with mileage (maintenance, fuel, insurance and costs of unexpected breakdowns)
  - there is a fixed cost associated with overhaul (independent on mileage)
  - each month, Zurcher needs to decide to send the bus to overhaul, which resets their mileage to zero, or to let them operate.
- This problem is a *dynamic programming problem*. When taking the decision whether to perform the overhaul or not, Zurcher needs to compare the operation cost not only with the cost of overhaul, but also take into account the reduction in operation costs in the future periods.
- While in this instance of the problem there is no externality across buses, so the buses could decide in isolation whether to go on maintenance or not, it is not hard to envision a variant of this problem where there are externalities. For instance, one may assume that there is a maximum number of buses that can go on overhaul at the same time.
- Later in this lecture, we shall derive the optimal policy for Harold Zurcher, (somewhat freely) based on Rust's data.

A look at the data

- Our data (synthetized on the basis of Rust's paper) consist of:
  - The number of buses in each state at the initial time; there are 30 states, each standing for a bracket of 12,500 mile .
  - The Markov transitions from a state-decision pair (xy) to the next state x'. There are 40 periods (quarter years over 10 years); when no maintenance is performed, one stays in the same state with probability .75 and transits to the next state with probability .25; when maintenance is performed, one transits to states 1 for sure.
  - The cost associated with each decision (overhaul or not) given the state x. The cost of replacing an engine is \$8,000 (in 1985 dollars), and the operations cost is $c(x) = x \times 5.10^2$.
  - The discount factor is $\beta = 0.9$.
- Later in this course, we shall provide methods for estimating these parameters based on the observation of Harold Zurcher's decisions. For now, let's take them as given.

DYNAMIC PROGRAMMING AS LINEAR PROGRAMMING

- Consider a finite set of individual states $x \in \mathcal{X}$; and a set of possible actions $y \in \mathcal{Y}$; assume that at initial time, n_x individuals in state x. The total number of individuals is $N = \sum_{x \in \mathcal{X}} n_x$. (Note that n_x is not necessarily an integer, so it would be more correct to talk about "mass" than "number").
- The immediate payoff associated with choice $y \in \mathcal{Y}$ at time $t \in \mathcal{T} = \{1, ..., T\}$ in state $x \in \mathcal{X}$ is u_{xy}^t, discounted at time zero (typically: $u_{xy}^t = \beta^t u_{xy}$ where β is a constant discount factor).
- The individual states undergo a Markov transition. The transition depends on the y chosen; hence, let $P_{x'|xy}$ be the probability of a transition to state x' conditional on the current state being $X_t = x$ and the current choice being $Y_t = y$. For $U \in \mathbb{R}^{\mathcal{X}}$, $P^\intercal U = \sum_{x'} P_{x'|xy} U_{x'}$ denotes the expectation of $U_{X_{t+1}}$ given $X_t = x$ and $Y_t = y$.

Policy variable

- Let π^t_{xy} be the number of individuals who are in state x and choose y ("policy variable").
- Define n^t_x be the number of individuals in state x at time t. We have the counting equation
$$\sum_{y \in \mathcal{Y}} \pi^t_{xy} = n^t_x.$$
- We have $n^1_x = n_x$ and because of the Markov transitions,
$$\sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} P_{x'|xy} \pi^{t-1}_{xy} = n^t_x,\ 1 \leq t \leq T,$$
which express that among the individual in state x who choose y at time $t-1$, a fraction $P_{x'|xy}$ transit to state x' at time t.

Primal problem: central planner's problem

- The central planner's problem is:

$$\max_{\pi^t_{xy}\geq 0} \sum_{x\in\mathcal{X},\ y\in\mathcal{Y},\ t\in\mathcal{T}} \pi^t_{xy} u^t_{xy} \tag{1}$$

$$s.t. \sum_{y'\in\mathcal{Y}} \pi^t_{x'y'} = \sum_{x\in\mathcal{X},\ y\in\mathcal{Y}} P_{x'|xy} \pi^{t-1}_{xy} \ \forall t \in \mathcal{T}\setminus\{1\} \quad \left[U^t_x\right]$$

$$\sum_{y'\in\mathcal{Y}} \pi^1_{x'y'} = n_x \ \left[U^1_x\right]$$

Dual problem

We have introduced U_x^t the Lagrange multiplier associated with the constraints at time t. It will be convenient to also introduce $U_x^{T+1} = 0$. The dual problem is

$$
\begin{aligned}
\min_{U_x^t,\ t\in\mathcal{T},\ x\in\mathcal{X}} & \sum_{x\in\mathcal{X}} n_x U_x^1 \qquad (2)\\
s.t.\ & U_x^t \geq u_{xy}^t + \sum_{x'} U_{x'}^{t+1} P_{x'|xy}\ \forall x \in \mathcal{X},\ y \in \mathcal{Y},\ t \in \mathcal{T} \setminus \{T\}\\
& U_x^T \geq u_{xy}^T\ \forall x \in \mathcal{X}, y \in \mathcal{Y}
\end{aligned}
$$

Complementary slackness and Bellman's equation

By complementary slackness, we have

$$\pi_{xy}^{t} > 1 \Longrightarrow U_{x}^{t} = u_{xy}^{t} + \sum_{x'} U_{x'}^{t+1} P_{x'|xy} \tag{3}$$

whose interpretation is immediate: if y is the optimal choice in state x at time t, then the intertemporal payoff of x at t is the sum of her myopic payoff u_{xy}^{t} and her expected payoff at the next step.
As a result, the dual variable is called *intertemporal payoff* in the vocable of dynamic programming. The relation (3) yields *Bellman's equation*

$$U_{x}^{t} = \max_{y \in \mathcal{Y}} \left\{ u_{xy}^{t} + \sum_{x'} U_{x'}^{t+1} P_{x'|xy} \right\}. \tag{4}$$

DP AS LP: VECTORIZATION

- We will need to represent matrices (such as U_x^t) and 3-dimensional arrays (such as u_{xy}^t). Consistent with the use in R, we will represent a matrix M_{ij} by varying the first index first, i.e. a 2×2 matrix will be represented as $vec\left(M\right)=M_{11},M_{21},M_{12},M_{22}$. Likewise, a $2\times 2\times 2$ 3-dimensional array A will be represented by varying the first index first, then the second, i.e. $vec\left(A\right)=A_{111},A_{211},A_{121},A_{221},A_{112},A_{212},A_{122},A_{222}$.
  - In R, this is implemented by `c(A)`; in Matlab, by `reshape(A,[n*m,1])`.
- A very important identity is

$$vec\left(BXA'\right)=\left(A\otimes B\right)vec\left(X\right),$$

where $\otimes$ is the Kronecker product: for 2x2 matrices,

$$A\otimes B=\begin{pmatrix}a_{11}B & a_{12}B\\ a_{21}B & a_{22}B\end{pmatrix}.$$

DP as LP: vectorizing the data and unknowns

- Recall, indices $xy \in \mathbb{R}^{|\mathcal{X}||\mathcal{Y}|}$ are represented by varying the first index first.
- Let:
  - P be the $(|\mathcal{X}|\,|\mathcal{Y}|) \times |\mathcal{X}|$ matrix of term $P_{x'|xy}$.
  - J be the $(|\mathcal{X}|\,|\mathcal{Y}|) \times |\mathcal{X}|$ matrix of term $1\{x = x'\}$. One has

$$J = 1_{\mathcal{Y}} \otimes I_{\mathcal{X}}.$$

  - U be the column vector of size $|\mathcal{X}|\,|T|$ obtained by stacking the vectors U^1,...,U^T.
  - b be the column vector of size $|\mathcal{X}|\,|T|$ whose $|\mathcal{X}|$ first terms are the terms of n, and whose other terms are zero.
  - u be the column vector of size $|\mathcal{X}|\,|\mathcal{Y}|\,|T|$ obtained by stacking the vectors u^1,..., u^T.
  - π be the vector obtained by stacking the vectors π^1,...,π^T.

DP as LP: vectorizing the constraint matrix

- A is the $|T|\,|\mathcal{X}|\,|\mathcal{Y}| \times |T|\,|\mathcal{X}|$ matrix

$$A = \begin{pmatrix} J & -P & 0 & \cdots & 0 \\ 0 & J & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -P & 0 \\ \vdots & & \ddots & J & -P \\ 0 & \cdots & \cdots & 0 & J \end{pmatrix}$$

- Letting $N_{\mathcal{T}}$ be the $T \times T$ matrix given by

$$N_{\mathcal{T}} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & 0 \\ \vdots & & & \ddots & 1 \\ 0 & \cdots & & \cdots & 0 \end{pmatrix}$$

the constraint matrix can be reexpressed as

$$A = I_{\mathcal{T}} \otimes J - N_{\mathcal{T}} \otimes P = I_{\mathcal{T}} \otimes 1_{\mathcal{Y}} \otimes I_{\mathcal{X}} - N_{\mathcal{T}} \otimes P.$$

DP AS LP: DUALITY

- Although we'll see much faster direct methods, the primal and dual problems could be solved by a black-box linear programming solver.
- Then the primal problem expresses as

$$\begin{aligned} &\max_{\pi \geq 0} u^\intercal \pi \\ &s.t.\ A^\intercal \pi = b\ \ [U] \end{aligned}$$

while the dual problem is given by

$$\begin{aligned} &\min_{U} b^\intercal U \\ &s.t.\ AU \geq u\ \ [\pi]. \end{aligned}$$

Backward induction

But there is in fact a much faster way to compute the primal and dual solutions without having to use the full power of a linear programming solver. Along with the fact that $U^{T+1} = 0$, Bellman's equation (4) implies that there is a particularly simple method to obtain the dual variables U^t, by solving recursively backward in time, from $t = T$ to $t = 1$. This method is called *backward induction*:

Algorithm

```
Set $U^{T+1} = 0$
For $t = T$ down to $1$, set $U_x^t := \max_{y \in \mathcal{Y}} \left\{ u_{xy}^t + \sum_{x'} U_{x'}^{t+1} P_{x'|xy} \right\}$.
```

Forward induction

The primal variables π^t are then deduced also by recursion, but this time forward in time from $t = 1$ to $t = T - 1$, by the so-called *forward induction* method:

Algorithm

```
Set $n^1 = n$ and compute $(U^t)$ by backward induction.
For $t = 1$ to $T$,
    Pick $\pi^t$ such that $\pi^t_{xy} / n^t_x$ is a probability measure supported in the set
        $$\left\{ y : U^t_x = u^t_{xy} + \sum_{x'} U^{t+1}_{x'} P_{x'|xy} \right\}.$$
    Set $n^{t+1}_{x'} := \sum_{x \in \mathcal{X},\ y \in \mathcal{Y}} P_{x'|xy} \pi^{t-1}_{xy}$
End
```

Remarks

1. The dual variable is U always unique (this follows from the backward induction computation); the primal variable is not, as there may be ties between several states.
2. The computation by the combination of the backward and forward algorithms is much faster than the computation by a black-box linear programming solver.

Introducing capacity constraints

- Now assume that the total number of alternatives y chosen at time t cannot be more than m_y^t (either because the workshop has a maximal capacity, or because operations require a minimum number of buses in service).
- The primal problem becomes

$$\begin{aligned}
&\max_{\pi_{xy}^t \geq 0} \sum_{x\in\mathcal{X},\ y\in\mathcal{Y},\ t\in\mathcal{T}} \pi_{xy}^t u_{xy}^t \\
&s.t. \sum_{y'\in\mathcal{Y}} \pi_{x'y'}^t = \sum_{x\in\mathcal{X},\ y\in\mathcal{Y}} P_{x'|xy}\pi_{xy}^{t-1} \ \left[U_x^t\right] \\
&\sum_{y'\in\mathcal{Y}} \pi_{x'y'}^1 = n_x \ \left[U_x^1\right] \\
&\sum_{x\in\mathcal{X}} \pi_{xy}^t \leq m_y^t \ [\lambda_y^t]
\end{aligned} \tag{5}$$

Introducing capacity constraints (ctd)

- Let us describe this problem in matrix form. Let $\tilde{\pi}^t$ be the matrix of term π^t_{xy} for fixed t. The last constraint rewrites $1^\intercal_{\mathcal{X}} \tilde{\pi}^t \leq (m^t)^\intercal$. Vectorizing yields $vec\left(1^\intercal_{\mathcal{X}} \tilde{\pi}^t I_{\mathcal{Y}}\right) \leq vec\left(m^t\right)$, thus

$$\left(I_{\mathcal{Y}} \otimes 1^\intercal_{\mathcal{X}}\right) vec\left(\tilde{\pi}^t\right) \leq vec\left(m^t\right),$$

hence the constraint rewrites $B^\intercal \pi \leq m$, with

$$B = \begin{pmatrix} I_{\mathcal{Y}} \otimes 1_{\mathcal{X}} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & I_{\mathcal{Y}} \otimes 1_{\mathcal{X}} \end{pmatrix} = I_{\mathcal{T}} \otimes I_{\mathcal{Y}} \otimes 1_{\mathcal{X}}.$$

Introducing capacity constraints (ctd)

- The primal problem then writes

$$\max_{\pi\geq 0} u^{\intercal}\pi$$
$$s.t.\ A^{\intercal}\pi = b\ \ [U]$$
$$B^{\intercal}\pi \leq m\ \ [\Lambda]$$

whose dual is

$$\min_{U,\Lambda\geq 0} b^{\intercal}U + m^{\intercal}\Lambda$$
$$s.t.\ AU + B\Lambda \geq u\ \ [\pi]$$

- The dual becomes

$$\begin{aligned} &\min_{U_x^t,\lambda_y^t\geq 0} \sum_{x\in\mathcal{X}} n_x U_x^1 + \sum_{x\in\mathcal{X}}\sum_{t\in\mathcal{T}} m_y \lambda_y^t \\ &s.t.\ U_x^t \geq u_{xy}^t - \lambda_y^t + \sum_{x'} U_{x'}^{t+1} P_{x'|xy}\ \forall x\in\mathcal{X},\ y\in\mathcal{Y},\ t\in\mathcal{T}\setminus\{T\} \\ &U_x^T \geq u_{xy}^T\ \forall x\in\mathcal{X}, y\in\mathcal{Y} \end{aligned} \tag{6}$$

and λ_y^t interprets as the shadow price of alternative y at time t.

Section 2

CODING

Preventive maintance

- Harold Zurcher's problem: The state $x \in \mathcal{X} = \{x_0, ..., \bar{x}\}$ of each bus at each period t is the mileage since last overhaul.
- The transition between states is as follows:
  - When no overhaul is performed, these states undergo random transitions (depending on how much the bus is used): $x_i \to x_{i'}$ with some probability $P_{i'|i}$, where $i' \geq i$.
  - When overhaul is performed on a bus, the state is restored to the zero state x_0.
- There is a fixed cost C associated with overhaul (independent on mileage), while operations costs $c\left(x\right)$ increase with mileage (maintenance, fuel, insurance and costs of unexpected breakdowns).

Translating the problem

- Assume the states are discretized into 12,500 mile brackets. There are 30 states, so $\mathcal{X} = \{1, ..., 30\}$. The choice set is $\mathcal{Y} = \{y_0 = 1, y_1 = 2\}$ (operate or overhaul).
- There are 40 periods (quarter years over 10 years)
- Transitions:
  - If no overhaul is performed, each state but the last one has a probability 25% of transiting to the next one, and probability 75% of remaining the same. The last state transits to 1 with probability 25% (overhaul is performed when beyond last state).
  - If overhaul is performed, the state transits to 1 for sure.
- Costs:
  - The cost of replacing an engine is C=\$8,000 (in 1985 dollars).
  - The operations cost is $c(x) = x \times 5.10^2$.
- The discount factor is $\beta = 0.9$.

Building the transitions

- First, we build the transition matrix $P_{x'|xy}$ matrix of dim=(nbX*nbY,nbX), so that P[xy,xprime] stands for $P_{x'|xy}$.
- Let

$$L_{\mathcal{X}} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \text{ and } R_{\mathcal{X}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & \vdots & & \vdots \\ 1 & \vdots & & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix}$$

so that P is given by

$$P = 1_{y_0} \otimes (0.75 I_{\mathcal{X}} + 0.25 L_{\mathcal{X}}) + 1_{y_1} \otimes R_{\mathcal{X}}$$

- This is implemented by:

```
IdX=sparseMatrix(1:nbX,1:nbX)
LX=sparseMatrix(c(nbX,1:(nbX-1)),1:nbX)
RX=sparseMatrix(1:nbX,rep(1,nbX),dims=c(nbX,nbX))
P=kronecker(c(1,0),0.75*IdX+0.25*LX)+kronecker(c(0,1),RX)
```

Building the constraint matrix

- Recall that
$$A = I_{\mathcal{T}} \otimes 1_{\mathcal{Y}} \otimes I_{\mathcal{X}} - N_{\mathcal{T}} \otimes P.$$
- This is implemented by:

```
IdT = sparseMatrix(1:nbT,1:nbT)
NT = sparseMatrix(1:(nbT-1),2:nbT,dims = c(nbT,nbT))
A = kronecker(kronecker(IdT,matrix(1,nbY,1)),IdX) -
kronecker(NT,P)
```

Building the payoffs and the distributions

- Next, we build u_{xyt}
  - First the u_{xy}'s so that $u_{x1} = -x \times 5.10^2$ for $x < \bar{x}$, and $u_{\bar{x}1} = -C$, while $u_{x2} = -C$ for all x. Thus u_{xy} is given by:
    ```
    u_xy = c(-5E2*(1:(nbX-1)),-5E2*nbX*.75-C*.25 ,rep(-C,nbX))
    ```
  - Next the u_{xyt} so that $u_{xyt} = u_{xy}\beta^t = vec\left(\left(\beta^1, ..., \beta^T\right) \otimes u_{xy}\right)$, hence u_{xyt} is given by:
    ```
    u = c(kronecker(beta^(1:nbT),u_xy))
    ```
- We build b_{xt}:
  ```
  b_xt = c(n1_x,rep(0,nbX*(nbT-1)))
  ```

Calling the optimizer

- Finally, we call the optimizer:
```
result = gurobi (
list(A=A,obj=c(b_xt),modelsense="min",rhs=u_xyt,sense=">",
lb=-Inf), params=list(OutputFlag=0) )
```
- We recover the residual value at time t by:
```
U_x_t_gurobi = array(result$x,dim=c(nbX,nbT))
```