Exercise set 5: NPL

Dynamic Programming Spring 2020,

by Patrick Kofod Mogensen and Maria Juul Hansen

Introduction

This week we will revisit the Rust model. We will try to estimate the parameters using an alternative method, namely Nested Pseudo Likelihood (NPL) as proposed in Aguirregabiria and Mira (2002).

NPL

In NFXP we had an outer maximum likelihood loop searching over θ-values, and an inner solution loop making sure that EV was updated for each θ-trial. Now, we swap the nesting. This means that we are going to solve the model in the outer loop using policy iterations, but every time we take a step towards the solution, we fully estimate a conditional logit. This is why it is called nested pseudo likelihood (NPL), since the pseudo likelihood step is nested in the solution method. Let us first consider the policy iterations. Policy iterations essentially take a policy, use it to update the value function, and then use the updated value function to calculate a new policy. This is a very efficient method, obtaining quadratic convergence rates. In NPL our policies are the choice probabilities.

In the inner loop of NFXP, we solve for the solution of the model using value function iterations and Newton-Kantorovich iterations. Aguirregabiria and Mira (2002) [AM] show how we can solve this model using policy iterations on the space of conditional choice probabilities instead. This means that instead of searching for a fixed point in $EV = \Gamma(EV)$, we look for a fixed point in $P = \Psi(P)$, where $\Psi : [0,1]^N \to [0,1]^N$is the policy iteration operator. Let us first see how this operator is derived. Details are in AM, but it is really surprisingly simple. Notation is as follow: $u(s,a)$ is the instantaneous utility function, where $s = (x, \epsilon)$, and $a \in A$ is the decision. In addition, $u(\cdot,\cdot)$ is parameterized by a vector θ_c. We dichotomize such that x contains observed states, and ϵ contains unobserved states. The joint state s follows a transition process characterized by the transition probabilities

$$p(x', \epsilon'|x, a, \epsilon) = g(\epsilon'|x')f(x'|x, a),$$

which is simply Rust's (CI) assumption. The transition process $g(\epsilon|x)$ is parameterized by θ_g, and $f(x'|x,a)$ is parameterized by θ_f. Assume a discount factor $\beta \in (0,1)$ and an infinite horizon. Then we are in "Blackwell territory" (since we also have discrete states) with a value function $V(s)$ as the solution to

$$V(s) = \max_{a \in A} \left\{ u(s,a) + \beta \int V(s')p(ds'|s,a) \right\}.$$

To solve this, first assume that $u(s,a) = \tilde{u}(x,a) + \epsilon(a)$. We can then use VFI and numerical integration and find $V(s)$. This is computationally expensive, so we might do as Rust, and solve for EV defined as

$$EV(x,\epsilon,a) \equiv \int_y \int_\eta V(y,\eta) p(\mathrm{d}y, \mathrm{d}\eta | \mathrm{x}, \epsilon, \mathrm{a}, \theta_\mathrm{g}, \theta_\mathrm{f}).$$

From this we get the following Bellman equation

$$\begin{aligned} EV(s,a) &= \int_y \int_\eta \max_{a \in A} \left\{ u(s,a) + \beta EV(s',a)) \right\} p(\mathrm{d}y, \mathrm{d}\eta | \mathrm{x}, \epsilon, \mathrm{a}, \theta_\mathrm{g}, \theta_\mathrm{f}) \\ &= \int_y \int_\eta \max_{a \in A} \left\{ u(s,a) + \beta EV(s',a)) \right\} g(\mathrm{d}\eta | \mathrm{x}, \theta_\mathrm{g}) \mathrm{f}(\mathrm{d}y | x, a, \theta_f) \end{aligned}$$

The inner integral over the max is the social surplus function from McFadden (1973), which has a closed form solution given our usual iid extreme value type 1 shocks. This means we simply have $EV(x,a)$, since it doesn't depend on ϵ. AM do something similar, but define a different value function. This is called the integrated value function or ex-ante value function, and is defined as

$$V_\sigma(x) = \int V(x,\eta) g(\mathrm{d}\eta | x, \theta_g).$$

Notice how this is only a function of x, not a. Just as in Rust, this leads to a contraction mapping, this time in V_σ instead of EV. Before we had that $EV(s,a) = EV(x,a)$ was the social surplus function integrated with respect to the transition process of the observed state. In AM's ex-ante value function approach, we simply get that V_σ is the social surplus function. The Hotz-Miller insight was that the partial derivative of the social surplus function with respect to a choice-specific value function is the associated conditional choice probability (CCP), and that this mapping is invertible such that we can find value functions from CCPs. We proceed as follows:

1. Re-write the ex-ante value function by "replacing" the integral over ϵ with CCPs
2. Isolate $V_\sigma$to obtain our inverse mapping $\varphi : [0,1]^N \to \mathbf{V}$, where $V_\sigma \in \mathbf{V}$.
3. Form the composite function $\Psi(P) : [0,1]^N \to [0,1]^N$ to create a mapping giving us current period best CCPs $P' = \Psi(P) = \Lambda(\varphi(P))$ given some belief of future behaviour P.

We want to find a P^* such that $\Lambda(\varphi(P^*)) = P^*$. The interpretation of such a fixed point would be: if I think I will follow $P^*(x)$ in the future, $P^*(x)$ is also my optimal behaviour today. This should also explain the name ex-ante above. P^* tells us what the best randomization policy is ex-ante the realization of the shock. Notice we are talking about choice probabilities as behaviour. This is because we want to characterize behaviour without using ϵ explicitly. This is fine, since our final goal is estimation, and there we do not know ϵ.

Finding (Ψ, φ)

First consider a given $x \in X$. What is the Bellman equation at this x? This is the integrated Bellman equation we saw earlier. An important step in doing NPL is to realize that this is equivalent to (see Aguirregabiria (1999))

$$V_\sigma(x) = \sum_{a \in A} P(a|x) \left\{ \tilde{u}(x,a) + E[\epsilon(a)|x,a] + \beta \sum_{x'} f(x'|x,a) V_\sigma(x') \right\}. \tag{1}$$

Here, we have assumed that X is finite (discrete). The conditional choice probabilities are defined as

$$P(a|x) = \int I \left\{ a = \arg\max_{j \in A} [v(x,j) + \epsilon(j)] \right\} g(\mathrm{d}\epsilon|\mathrm{x}).$$

With our extreme value type I distributional assumption on ϵ, this is simply the conditional multinomial logit formula, which has a closed form. Given future behaviour P and the induced choice-specific continuation values, this is our Ψ-mapping. Now we just need to find φ. This is done by stacking the equations in 1 for all x to get

$$V_\sigma = \sum_{a \in A} P(a) * [\tilde{u}(a) + e(a,P) + \beta F(a) V_\sigma].$$

Recognizing that his is possible puts us in a very strong position, as this can easily be rewritten as

$$\begin{aligned} V_\sigma &= (I - \beta F^U(P))^{-1} \left\{ \sum_{a \in A} P(a) * [\tilde{u}(a) + e(a,P)] \right\} \\ V_\sigma &= \varphi(P). \end{aligned}$$

To calculate this, we need to know what $e(a,P)$ is, but other than that, it is simple linear algebra. The function (or vector) is a stack of conditional expected values of choice j-specific shock $\epsilon(j)$ conditional on x, and conditional on a actually being optimal (remember it is multiplied by the CCP of a). This means that

$$e(a,P) \equiv E[\epsilon(a)|x,a] = (P(a|x))^{-1} \int \epsilon(a) \cdot \{\tilde{v}(x,a) + \epsilon(a) \geq \tilde{v}(x,j) + \epsilon(j), \forall j \in A\} \, g(\mathrm{d}\epsilon|x).$$

This is the usual conditional expectation, where we condition on two things: x, and a being optimal (having the highest choice-specific value). For a specific x, this expectation is not 0, even though $E[\epsilon(a)] = 0$. In some states it might be more probable that $a = 1$ is more likely to be optimal, even though the converse is true in other states.

Implementation - spelling it out

- Step 1) Set K=0.

- Step 2) Estimate (non-parametrically) θ_p . Here it is $\pi_0, \pi_1, \ldots, \pi_{max}$, where $\pi_{max}$is the probability associated with the largest increase in mileage from one period to the next.
- Step 3) Initialize with a guess for CCPs (can be anything, but closer to true will speed things up) $\hat{P}_K$, and parameters $\hat{\theta}_{c,K}$.
- Step 4.1) Maximize the likelihood of observing the data for fixed $\hat{P}_K$, update $\hat{\theta}_{c,K+1} = \arg\max_\theta \ell^1(\theta|data, \theta_p, \hat{P}_K)$
- Step 4.2) Update $\hat{P}_{K+1} = \Psi(P) = \Lambda\left(\varphi\left(\hat{P}_K, \hat{\theta}_{c,K+1}\right)\right)$.
- Step 4.3) If $||\hat{P}_{K+1} - \hat{P}_K||_\infty < \text{tol}_P$ and $||\hat{\theta}_{K+1} - \hat{\theta}_K||_\infty < \text{tol}_\theta$ stop, and accept the current iterates, else start from Step 4).

Questions

Note the following about notation: P is used as the variable name for the Markovian transition matrix in the Rust-problem and code. The correspondence is: P1 (code) = F (notes), pk (code) = P (notes), Fu (code) = F^U(notes). In the following we use code notation

1. Look at the ReadMe.txt to get an overview of the code
2. What is the difference between $EV(x, a)$ and $V_\sigma(x)$?
3. Write the formula for $P = \Lambda(V_\sigma)$, $V_\sigma = \varphi(P)$ and $P = \Psi(P)$, exploiting the extreme value type I distribution on ϵ. (Hint: what is $e(a, P) \equiv E[\epsilon(a)|x, a]$ under this distributional assumption?). Insert the found formulars under zurcher.phi ($V_\sigma = \varphi(P)$) and zurcher.lambdaa ($P = \Lambda(V_\sigma)$)
4. Solve the model using NPL, and plot the convergence of psi to check that your results are correct
5. Now you have to compare NFXP and NPL. Remeber that NFXP solve the fixed-point problem in value function space (EV = G(EV)) and NPL solves it in CCP space (pk = Y(pk)). Calculate the CCPs from either method and compare the results
6. $F^U(pk)$ is the unconditional transition probabilities induced by pk (vector) - what does that mean?
   (a) Try changing the number of gridpoints to 5, place a debugger in the zucher.phi.py-function and compare Fu and P1. What is the difference between the two?
7. What determines if NFXP is computationally cheaper to use than NPL? Think about what is in the inner loop of either algorithm.

8. Now we have to Estimate the model using NPL. In order to estimate the model you should understand npl.estimate, npl.ll (skip the part of computing the gradient and the Hessian).

9. Fill in the missing stuff in NPL.ll and run the code below to check that your results are correct