Estimación de Demanda BLP

Clase de Reforzamiento — `pyblp`

Organización Industrial Empírica · Magíster en Análisis Económico

Basado en Berry, Levinsohn & Pakes (1995) y `pyblp` v1.x (Conlon & Gortmaker, 2020)

Objetivo: Este documento sirve de apoyo para la clase de reforzamiento sobre estimación de demanda con el modelo de coeficientes aleatorios (BLP). Se presenta el marco teórico de forma concisa, la lógica de identificación y estimación, y tips prácticos para implementar el modelo con `pyblp` en Python.

Índice

1. El Modelo BLP: Fundamentos Teóricos

1.1. Configuración del mercado

Consideramos un mercado con J productos diferenciados y un *outside good* (bien exterior, $j = 0$). Hay T mercados (e.g., ciudades $\times$ períodos). Cada consumidor i elige el producto que maximiza su utilidad indirecta:

Utilidad indirecta del consumidor i por el producto j en el mercado t

$$u_{ijt} = \underbrace{\alpha_i p_{jt}}_{\text{precio}} + \underbrace{x_{jt}\beta_i}_{\text{características}} + \underbrace{\xi_{jt}}_{\text{calidad no obs.}} + \varepsilon_{ijt} \tag{1}$$

donde:

- p_{jt}: precio del producto j en el mercado t.
- x_{jt}: vector de características observables (K_2 de ellas entran con coeficientes aleatorios, K_1 entran de forma lineal en δ).
- ξ_{jt}: atributo no observable que también valoran los consumidores (absorbe calidad, marca, etc.).
- ε_{ijt}: choque de gusto i.i.d., distribuido Valor Extremo Tipo I.

1.2. Coeficientes aleatorios

La clave del modelo BLP es que los coeficientes varían entre consumidores, capturando *heterogeneidad en preferencias*:

$$\begin{pmatrix} \alpha_i \\ \beta_i \end{pmatrix} = \begin{pmatrix} \bar{\alpha} \\ \bar{\beta} \end{pmatrix} + \Pi d_i + \Sigma \nu_i \tag{2}$$

- $\bar{\alpha}, \bar{\beta}$: coeficientes medios de la población (*parámetros lineales* θ_1).
- d_i: vector de características demográficas del consumidor i (ingreso, edad, etc.).
- $\nu_i \sim \mathcal{N}(0, I)$: shocks de gusto inobservables (distribución normal estándar).
- Π: matriz de interacciones demográficas (*parámetros no lineales*).
- Σ: matriz triangular inferior de escala de la heterogeneidad no observada (*parámetros no lineales*).

Intuición clave

Si $\Sigma = 0$ y $\Pi = 0$ recuperamos el Logit Multinomial simple (MNL). El BLP generaliza el MNL al permitir que la *sustitución* sea más realista: productos con características similares compiten más entre sí.

1.3. Utilidad media y descomposición

Es conveniente descomponer la utilidad en una parte **media** (igual para todos) y una parte **individual**:

$$u_{ijt} = \underbrace{\delta_{jt}(\theta)}_{\text{utilidad media}} + \underbrace{\mu_{ijt}(\theta, d_i, \nu_i)}_{\text{desviación individual}} + \varepsilon_{ijt} \tag{3}$$

donde $\delta_{jt} \equiv x_{jt}\bar{\beta} + \bar{\alpha}p_{jt} + \xi_{jt}$ es la **utilidad media** del producto j (o *mean utility*).

2. Participaciones de Mercado y Contracción

2.1. Shares predichos

Dado que ε_{ijt} es i.i.d. Valor Extremo Tipo I, la probabilidad de elección del individuo i es:

$$s_{ijt}(\delta, \theta) = \frac{\exp(\delta_{jt} + \mu_{ijt})}{\sum_{k=0}^{J_t} \exp(\delta_{kt} + \mu_{ikt})} \tag{4}$$

Los shares de mercado predichos se obtienen integrando sobre la distribución de consumidores:

$$s_{jt}(\delta, \theta) = \int s_{ijt}(\delta, \theta)\, dP(d, \nu) \approx \frac{1}{I} \sum_{i=1}^{I} s_{ijt}(\delta, \theta) \tag{5}$$

Esta integral no tiene forma cerrada $\Rightarrow$ se aproxima mediante **integración numérica** (Monte Carlo, cuadraturas de producto, etc.).

2.2. El operador de contracción (Berry 1994)

Para estimar el modelo, necesitamos δ_{jt} que iguale los shares predichos a los observados S_{jt}. Berry (1994) muestra que existe un **único** vector δ que satisface $s(\delta, \theta) = S$ y propone el siguiente operador de punto fijo:

Contracción de Berry

$$\delta^{(r+1)} = \delta^{(r)} + \ln S - \ln s(\delta^{(r)}, \theta) \tag{6}$$

Se itera hasta convergencia: $\|\delta^{(r+1)} - \delta^{(r)}\| < \epsilon_{\text{inner}}$.

`pyblp` implementa también el **método de Newton** para la solución del sistema interno, que converge más rápido que la contracción simple (opción por defecto en versiones recientes).

3. Identificación y Estimación GMM

3.1. La ecuación estructural y las restricciones de momentos

Una vez obtenido $\delta(\theta)$, recuperamos la perturbación estructural:

$$\xi_{jt}(\theta) = \delta_{jt}(\theta) - x_{jt}\bar{\beta} - \bar{\alpha}p_{jt} \tag{7}$$

El supuesto de identificación central es: $\mathbb{E}[\xi_{jt} \cdot z_{jt}] = 0$, donde z_{jt} son **instrumentos** válidos.

3.2. Estimador GMM de dos etapas

La función objetivo GMM es:

$$\hat{\theta} = \arg\min_{\theta} \; g(\theta)' W g(\theta), \quad g(\theta) = \frac{1}{JT}\sum_{j,t} z_{jt} \cdot \xi_{jt}(\theta) \tag{8}$$

donde W es una matriz de pesos. El estimador eficiente usa $W = (Z'\hat{\Omega}Z)^{-1}$ con $\hat{\Omega} = \mathbb{E}[\xi\xi'|Z]$ (dos etapas o iterado). En `pyblp`, el parámetro `W_type` controla esto.

3.3. Separación de parámetros lineales y no lineales

El algoritmo BLP opera en dos niveles (*nested optimization*):

1. **Loop externo** (sobre $\theta_2 = \{\Pi, \Sigma\}$, parámetros no lineales): el optimizador minimiza la función objetivo GMM.
2. **Loop interno** (sobre δ dado θ_2): se resuelve la contracción de Berry.
3. **Parámetros lineales** ($\theta_1 = \{\bar{\alpha}, \bar{\beta}\}$): se obtienen analíticamente por mínimos cuadrados instrumentales (eliminación concentrada).

Tip: ¿Qué parámetros son "no lineales"?

Los no lineales son los que aparecen en μ_{ijt}, es decir, los que **interactúan con la heterogeneidad** (Σ, Π). Los coeficientes medios $\bar{\beta}$ y $\bar{\alpha}$ son lineales y no requieren optimización numérica directa.

4. Instrumentos para Identificación

4.1. ¿Por qué necesitamos instrumentos?

El precio p_{jt} es endógeno: las empresas fijan precios conociendo ξ_{jt}, por lo que $\text{Cov}(p_{jt}, \xi_{jt}) \neq 0$. Los instrumentos deben ser:

1. **Relevantes**: correlacionados con el precio (o las variables endógenas).
2. **Exógenos**: incorrelacionados con ξ_{jt}.

Tipo	Descripción	Justificación
BLP (1995)	Suma de características de rivales en el mercado	Costo marginal varía con competi
Hausman	Precio del mismo producto en otros mercados	Shocks de costo comunes, no de d
Costo	Variables de costo observadas (salarios, insumos)	Directamente vinculadas al precio
"Differentiation IVs"	Distancia en el espacio de características	Relevante para los coef. aleatorios

Cuadro 1: Instrumentos habituales en modelos BLP

4.2. Tipos comunes de instrumentos

Tip: Instrumentos para los parámetros no lineales

Para identificar Σ (la dispersión de los coeficientes aleatorios) se necesitan instrumentos que varíen la *distribución* de las alternativas, no solo el precio medio. Los "differentiation IVs" de Gandhi & Houde (2023) son una buena opción en `pyblp`: se construyen automáticamente con `pyblp.build_differentiation_instruments()`.

5. Elasticidades e Implicaciones

Con el modelo estimado, las elasticidades precio se calculan como:

$$\varepsilon_{jkt} = \frac{\partial s_{jt}}{\partial p_{kt}} \cdot \frac{p_{kt}}{s_{jt}} \tag{9}$$

$$\frac{\partial s_{jt}}{\partial p_{kt}} = \begin{cases} -\frac{1}{M_t} \int \alpha_i \, s_{ijt}(1 - s_{ijt}) \, dP & \text{si } j = k \\ +\frac{1}{M_t} \int \alpha_i \, s_{ijt} s_{ikt} \, dP & \text{si } j \neq k \end{cases} \tag{10}$$

A diferencia del Logit simple, aquí la elasticidad cruzada entre j y k **depende de las características** de ambos productos, reflejando patrones de sustitución realistas.

6. Implementación en `pyblp`: Guía Práctica

6.1. Instalación y versión

```
pip install pyblp

import pyblp
import numpy as np
import pandas as pd

print(pyblp.__version__)   # Verificar version (>= 1.0 recomendada)
pyblp.options.verbose = True  # Ver progreso de la estimacion
pyblp.options.digits = 2      # Decimales en la salida
```

Listing 1: Instalación y verificación

6.2. Estructura de datos requerida

`pyblp` requiere que los datos estén en un DataFrame con columnas específicas:

Columna	Tipo	Descripción
`market_ids`	str/int	Identificador de mercado
`product_ids`	str/int	Identificador de producto
`shares`	float	Participación de mercado
`prices`	float	Precio del producto
`demand_instruments*`	float	Instrumentos de demanda (0, 1, 2...)
otras características	float	Variables para las fórmulas

Cuadro 2: Columnas requeridas en `product_data`

Tip: Convención de instrumentos

Los instrumentos de demanda deben llamarse `demand_instruments0`, `demand_instruments1`, etc. Del mismo modo, los de oferta se llaman `supply_instruments0`, etc. `pyblp` los recoge automáticamente si siguen esta convención.

6.3. Formulaciones (`Formulation`)

Las formulaciones especifican qué variables entran en cada componente del modelo usando sintaxis de tipo Patsy/R:

```
# X1: variables con coeficientes LINEALES (usualmente precio)
# "0 +" elimina el intercepto; absorb='C(product_ids)' para EF de producto
X1 = pyblp.Formulation('0 + prices', absorb='C(product_ids)')

# X2: variables con coeficientes ALEATORIOS
# Incluye intercepto (para la escala del outside good)
X2 = pyblp.Formulation('1 + prices + sugar + mushy')

product_formulations = (X1, X2)
```

Listing 2: Definición de formulaciones

¡Ojo con el intercepto!

El intercepto en X_2 da un coeficiente aleatorio sobre la constante, capturando heterogeneidad general en el nivel de utilidad. Si se omite, se pierde un parámetro de heterogeneidad importante.

6.4. Integración numérica (`Integration`)

```
# Monte Carlo: rapido pero ruidoso, util para explorar
mc = pyblp.Integration('monte_carlo', size=50,
                       specification_options={'seed': 42})

```

```python
# Cuadratura de producto (Gauss-Hermite): mas precisa, recomendada
gh = pyblp.Integration('product', size=7)

# Cuadratura sparse (Smolyak): eficiente en alta dimension
sparse = pyblp.Integration('grid', size=5)
```

Listing 3: Métodos de integración

Tip: ¿Cuántos nodos usar?

Para resultados finales use integración por producto (`'product'`) con `size=7` o más nodos por dimensión. Para exploración inicial, Monte Carlo con `size=50–200` y semilla fija es más rápido. El error de Monte Carlo decrece como $O(1/\sqrt{I})$.

6.5. Configuración del Problema (`Problem`)

```python
problem = pyblp.Problem(
    product_formulations = product_formulations,
    product_data         = product_data,
    agent_formulation    = agent_formulation,  # Si hay demograficos
    agent_data           = agent_data,          # DataFrame de consumidores
    integration          = integration          # Si NO hay agentes
        externos
)
print(problem)  # Resumen del problema: dimensiones, IVs, etc.
```

Listing 4: Creación del objeto Problem

6.6. Estimación (`problem.solve()`)

```python
# Parametros iniciales para los no-lineales (Sigma, Pi)
# Sigma: desviaciones estandar de los coef. aleatorios
# Pi: interacciones con demograficos

sigma_0 = np.diag([0.3, 0.0, 0.0, 0.0])  # Solo intercepto con RC
pi_0 = None  # Sin demograficos

results = problem.solve(
    sigma         = sigma_0,
    pi            = pi_0,
    method        = '2s',       # GMM dos etapas ('1s', '2s', 'iterated')
    optimization  = pyblp.Optimization('l-bfgs-b'),  # Optimizador
    iteration     = pyblp.Iteration('squarem'),       # Contraccion
        interna
)
print(results)
```

Listing 5: Estimación con valores iniciales

Tip: Valores iniciales

Los valores iniciales de Σ y Π son cruciales. Empezar con ceros puede llevar a singularidades. Una estrategia práctica:

1. Estimar primero el Logit simple (todos en cero) para obtener ξ_{jt} iniciales.
2. Luego explorar una grilla de valores iniciales y quedarse con el mínimo global.

Signo de sigma_prices

El coeficiente de precio $\alpha_i = \bar{\alpha} + \sigma_\alpha \nu_i$ debe ser **negativo** en promedio. Si el valor inicial de σ_α es libre, el optimizador puede converger a óptimos locales con signo incorrecto. Fijar el signo con bounds o reparametrizar.

6.7. Post-estimación: elasticidades y markups

```python
# Elasticidades precio propias y cruzadas (una por mercado)
elasticities = results.compute_elasticities()

# Diagonales = elasticidades propias
own_elast = results.extract_diagonal(elasticities)
print(f"Elasticidad propia media: {own_elast.mean():.3f}")

# Markups bajo competencia de Nash-Bertrand
costs = results.compute_costs()
markups = results.compute_markups(costs=costs)

# Indices de diversidad y poder de mercado
hhi = results.compute_hhi()
```

Listing 6: Post-estimación

7. Replicación Nevo (2000): Datos de Cereales

7.1. Descripción de los datos

El ejercicio canónico de `pyblp` replica a Nevo (2000) con:

- **24 marcas** de cereales listos para comer.
- **94 ciudades** (mercados) y 4 trimestres $\rightarrow$ 94 mercados en la versión simplificada.
- Variables de producto: precio, contenido de azúcar (`sugar`), si es esponjoso (`mushy`).
- Datos demográficos: ingresos, edad, presencia de niños.
- Instrumentos BLP y costos de entrada como IVs.

7.2. Reproducción paso a paso

Ver el archivo `blp_nevo_ejemplo.py` adjunto para el código completo comentado. La estructura general es:

1. **Cargar datos**: `pyblp.data.NEVO_PRODUCTS_LOCATION` y `NEVO_AGENTS_LOCATION`.
2. **Definir formulaciones**: X_1 con FE de producto, X_2 con intercepto, precio, azúcar y mushy.
3. **Configurar demograficos**: formulación de agentes con ingresos e indicadoras.
4. **Construir el Problema** con datos de agentes externos.
5. **Estimar** con valores iniciales de Nevo y GMM de dos etapas.
6. **Post-estimar**: elasticidades, costos, markups.

8. Errores Comunes y Tips de Depuración

Tips de depuración rápida

1. **Verificar shares**: $\sum_j s_{jt} < 1$ para todo t, y $s_{0t} = 1 - \sum_j s_{jt} > 0$. Si no, revisar la construcción de los datos.
2. **Activar verbose**: `pyblp.options.verbose = True` para ver el progreso de la optimización. Si la función objetivo no disminuye, los valores iniciales son malos.
3. **Revisar el `print(problem)`**: muestra el número de IVs, parámetros y dimensiones. Un número insuficiente de IVs (menos que parámetros) causará problemas de identificación.
4. **Matriz de pesos singular**: ocurre si los instrumentos son colineales. Revisar si hay variables redundantes en `demand_instruments`.
5. **Convergencia del loop interno**: si la contracción no converge, usar `Iteration('squarem')` o Newton. Verificar con `results.contraction_evaluations`.
6. **Gradiente numérico vs. analítico**: `pyblp` calcula gradientes analíticos por defecto. Si se usan formulaciones complejas con absorción, revisar que el gradiente sea correcto con `pyblp.options.finite_differences`.

Mínimos locales

El problema de optimización BLP es no convexo. Se recomienda:

- Probar múltiples puntos de partida (grilla).
- Usar `pyblp.ProblemResults.to_pickle()` para guardar resultados intermedios.
- Comparar con estimaciones logit simples: si los parámetros RC son muy distintos del caso sin heterogeneidad, verificar la identificación.

9. Referencias

Referencias

Berry, S., Levinsohn, J., & Pakes, A. (1995). *Automobile Prices in Market Equilibrium. Econometrica*, 63(4), 841–890.

Berry, S. (1994). *Estimating Discrete-Choice Models of Product Differentiation. RAND Journal of Economics*, 25(2), 242–262.

Nevo, A. (2000). *A Practitioner's Guide to Estimation of Random-Coefficients Logit Models of Demand. Journal of Economics & Management Strategy*, 9(4), 513–548.

Conlon, C., & Gortmaker, J. (2020). *Best Practices for Differentiated Products Demand Estimation with pyblp. RAND Journal of Economics*, 51(4), 1108–1161.

Gandhi, A., & Houde, J.-F. (2023). *Measuring Substitution Patterns in Differentiated-Products Industries. Econometrica*, forthcoming.

Documento preparado como material de apoyo para la clase de reforzamiento de estimación de demanda BLP. Para dudas sobre el código o la teoría, revisar la documentación oficial de `pyblp` en `https://pyblp.readthedocs.io`.