Estimation Principles for Structural Models

Janoś Gabler

July 3, 2019

Outline

Introduction

Introduction

- Structural models are used for ex-ante policy evaluation
- They are tailored to one policy question
- Their parameters have to be estimated
- We fully abstracted from the estimation problem
- This time we fully abstract from complex models
  - Only estimate means and linear models

Goals for this lecture

- Explain the core estimation principles on toy models
  - Maximum Likelihood
  - Method of Simulated Moments
- Make you aware of typical numerical problems
- Introduce you to numerical optimization
- Explain why it is important to separate models from estimators

Basic Ideas

Preparation

Parametric Models

- A model is called parametric if all functions and distributions are specified up to a finite set of parameters
- Given a parameter vector and a parametric model:
  - A dataset can be simulated
    - Will be used for Method of Simulated Moments
  - The probability of observing a certain data point can be calculated
    - Will be used for Maximum Likelihood Estimation

Examples for Parametric and Non-Parametric Models

Example	Assumptions
$y_i = m(x_i, u_i)$	Smoothness of m, iid sampling
$y_i = m(x_i) + u_i$	+ additivity of errors
$y_i = m(x_i'\beta) + u_i$	+ single linear index
$y_i = x_i\beta + u_i$	+ m is the identity function
$u_i \sim \mathcal{N}(\mu, \sigma^2)$	+ distributional assumption

Example 1: Mean of a Normal Distribution

- Dataset: iid sample of variable y_i
- Model: $y_i \sim \mathcal{N}(\mu, \sigma^2)$
- Parameters to estimate: μ, σ

Maximum Likelihood

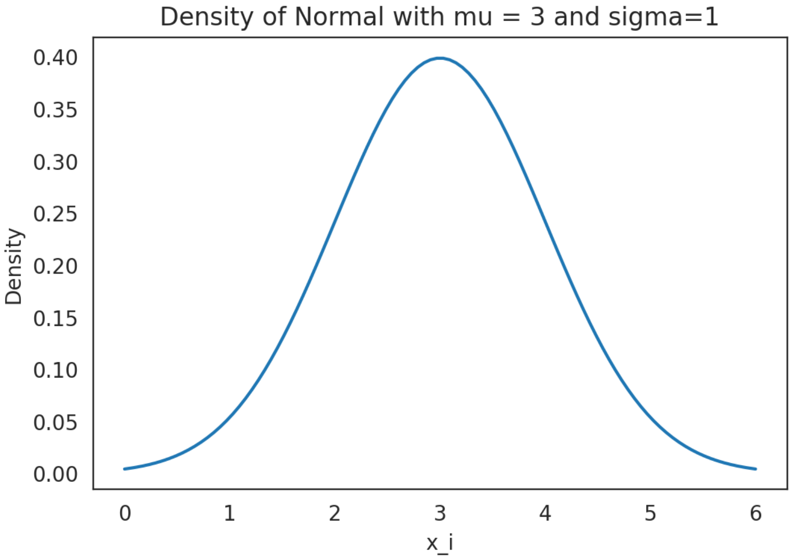
Example 1

- The model implies:

$$f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2\sigma^2}(y_i - \mu)^2\right)} \tag{1}$$

- f is the probability density function (pdf) of y_i
- Interpretation: y_i varies, μ and σ are fixed
- Density of whole sample is just product of individual densities

Graphical Intuition



Basic Idea of Maximum Likelihood

$$f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2\sigma^2}(y_i - \mu)^2\right)} = \phi(y_i, \mu, \sigma) = l_i(\mu, sigma) \quad (2)$$

- l is the likelihood contribution of individual i
- Interpretation: y_i is fixed, μ and σ vary
- Likelihood of whole sample is just product of individual likelihoods
- Use the parameters that maximize the likelihood of the sample as estimates

Python Implementation

```python
import numpy as np
import scipy

def likelihood(mu, sigma, sample):
    likelihoods = scipy.stats.norm.pdf(sample, loc=mu, scale=sigma)
    return np.prod(likelihoods)
```

Notebook

Look at likelihood_example_1.ipynb

Some Insights from the Example

- Use log-likelihood to avoid numerical problems
- Likelihood estimates can be counter-intuitive
- Larger sample -> more curved likelihood -> more precision
  - Standard errors will depend on curvature of likelihood
- Mean is estimated quite precisely, even in small samples

A Note on Terminology

- Parameters are constants, not random variables
- Maximum likelihood estimates are not “the most likely parameters”
- They a the parameter values that make the observed sample most likely!

Method of Simulated Moments

Basic Idea

- Remember: If we have a fully parametric model and a parameter vector, we can simulate a dataset generated by the model with that parameter vector
- For the true parameter vector, simulated and observed sample should be similar
- MSM: Take the parameter vector that produces the dataset which is most similar to the empirical data as estimate

What Does Similar Mean?

- Depends on the model
- Typically: Key moments are similar
- Selection of key moments depends on model
- Requires weighting matrix

Example 1: Which moments

- Key moments: unconditional mean and variance
- Looks like chicken-or-egg problem
- Only an artifact of very simple example

Example 1: Objective function

- empirical sample: $Y = y_1, y_2, \ldots, y_n$
- parameter vector: $\theta = (\mu, \sigma)^T$
- simulated sample, given θ: $\hat{Y}^\theta = \hat{y}_1^\theta, \hat{y}_2^\theta, \ldots, \hat{y}_n^\theta$
- empirical moments: $m^{emp} = (mean(Y), var(Y))^T$
- simulated moments: $m(\theta) = (mean(\hat{Y}^\theta), var(\hat{Y}^\theta))^T$
- Objective: $C(\theta) = (m(\theta) - m^{emp})^T W (m(\theta) - m^{emp})$
- θ^{MSM} minimizes this function

Python Implementation

```python
import numpy as np
import scipy

def msm_criterion(mu, sigma, sample):
    simulated_sample = np.random.normal(loc=mu, scale=sigma, size=len(sample))
    m_emp = np.array([[sample.mean()], [sample.std()]])
    m_sim = np.array([[simulated_sample.mean()], [simulated_sample.std()]])
    diff = m_emp - m_sim
    w = np.eye(2)
    return diff.T.dot(w).dot(diff)[0][0]
```

Notebook

Look at msm_example_1.ipynb

Some Insights from the Example

- Precision is now limited by two factors:
  - Sample variation
  - Simulation noise
- Likelihood needs the same assumption and will always be more efficient!

Why Would We Ever Use MSM?

- Likelihood often involves high dimensional integrals
- Likelihood is harder to derive
- Some say, MSM is more transparent
- Data might be on a slow server
- Different datasets

Numerical Optimization

Why Numerical Optimization

- Typically optimum cannot be calculated in closed form
  - Derivatives are too complicated to evaluate
  - Non differentiable functions
  - First order conditions are too complicated
- Numerical optimization works in these cases but:
  - Typically no guarantee of global optimum
  - Can be slow
  - Less precise than closed form
- WLOG, from now on only talk about minimization

We Already did it

- For surface plots, we evaluated objective over whole reasonable parameter space
- Know coordinates of optimum
- This methods is called grid search or brute force
- Infeasible for high dimensional parameter vectors:
  - Grid with 100 points in each dimension
  - 50 parameters

  $\longrightarrow$ Criterion has to be evaluated at 10^{100} gridpoints
- Need smarter algorithms to save evaluations!

Gradient Based vs. Gradient Free

- Gradient based algorithms
  - Require differentiable objective functions
  - Use first derivative to determine direction of step
  - Use second derivative to determine length of step
  - Need much fewer function evaluations
- Gradient free algorithms
  - Work with non-smooth functions
  - Basically a trial-and-error approach

Termination Criteria

- ftol
- gtol
- xtol

Local vs. Global Optimizers

- Local Optimizers:
  - Start from start parameters and go down-hill
  - Stop in first local minimum
- Genetic Algorithms (typically global):
  - Need bounds on parameter space
  - Sample uniformly from parameters space to get start population
  - Evolve to next generation by:
    - Killing worst parameters
    - Produce offspring of good parameters
- Pseudo global optimizers:
  - Do local optimization from random start values

Notebook

look at 1d_minimization.ipynb

Insights from the Example

- Sampling scheme can be very important
- Robust optimizers can solve some problems, but maybe you don't need them!
- Always understand the problem you want to solve!

Smooth Python Implementation

```
import numpy as np
import scipy

def smooth_msm_criterion(mu, sigma, sample):
    np.random.seed(5471)
    simulated_sample = np.random.normal(size=len(sample)) * sigma + mu
    m_emp = np.array([[sample.mean()], [sample.std()]])
    m_sim = np.array([[simulated_sample.mean()], [simulated_sample.std()]])
    diff = m_emp - m_sim
    w = np.eye(2)
    return diff.T.dot(w).dot(diff)[0][0]
```

Monte Carlo Results

Notebook

look at monte_carlo.ipynb

Some Theory

Closed Form Solution for Likelihood Function in Example 1

$$L(\mu,\sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} e^{\left(-\frac{1}{2\sigma^2}(y_i-\mu)^2\right)} \qquad \text{Likelihood}$$

$$\ell(\mu,\sigma) = \sum_{i=1}^{n} -\frac{1}{2} ln(2\pi\sigma) - \frac{1}{2\sigma^2}(y_i-\mu)^2 \qquad \text{Log-Likelihood}$$

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i-\mu) \overset{!}{=} 0 \qquad \text{FOC}$$

$$\mu^* = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Example 2: Linear Model

- Dataset: iid sample of variables y_i, x_i
- Model: $y_i = x_i'\beta + u_i$, $u_i \sim \mathcal{N}(0, \sigma^2)$
- Parameters to estimate: β, σ

Example 2: Linear Model

- Dataset: iid sample of variables y_i, x_i
- Model: $y_i = x_i'\beta + u_i$, $u_i \sim \mathcal{N}(0, \sigma^2)$
- Parameters to estimate: β, σ
- You might be tempted to call this an OLS model
- Later you'll see why this is not a good idea

Likelihood Function of Linear Model

$$f(y_i \mid \beta, \sigma, x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{1}{2\sigma^2}(y_i - x_i'\beta)^2\right)}$$

$$L(\beta, \sigma) = \prod_{i=1}^{n} f(y_i \mid \beta, \sigma, x_i)$$

- Take logs
- Set FOCs to zero
- If you make no mistake, you will see that β^* equals the OLS estimator

The Deep Reason (simplified!)

- It can be shown that:
  - Maximum Likelihood is asymptotically efficient
  - No unbiased estimator has smaller variance than maximum likelihood (Cramér-Rao Bound)
- From Gauss Markov Theorem we know:
  - OLS estimator is Best Linear Unbiased Estimator
  - Remember: sample mean = OLS on constant
- Since OLS and Maximum Likelihood estimators are both optimal, they have to coincide

MSM of Linear Model

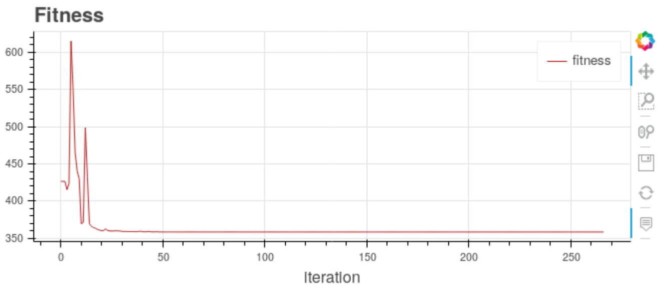
- Identifying assumption: u_i is uncorrelated with x_i
- $E(x_i u_i) = E(x_i(y_i - x_i'\beta)) = 0$
- Gives us k moment conditions for k parameters
- Those are the moments used when Linear model is estimated with GMM
- Could use the simulated version of those moments to estimate Linear model with MSM
- Reminder: This is just to explain the method

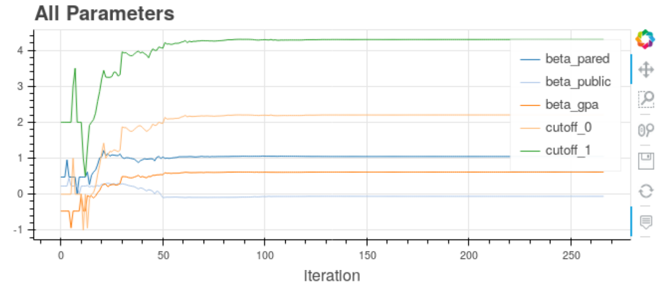
Estimagic

Estimagic

- Wrap local, global and pseudo global optimizers from Pygmo, Scipy and TAO
- Standard errors for likelihood and MSM models
- Elegant interface for typical constraints
  - e.g. estimate covariance matrices, probabilities, . . .
- Interactive dashboard with live convergence plots

Convergence Plots





Some Tipps

- Start by implementing toy examples
- When you hit problem, reproduce and solve them in minimal examples
- Use as much existing code as possible
- If you think you need a fancy optimizer, you probably don't
- If you ever do structural econometrics: use estimagic!
- But maybe wait a few months . . .