Practical Advice on Estimation

March 18, 2020

Estimation

- Estimates are often the solutions to optimization problems
  - Least squares
  - Maximum likelihood
  - Methods of moments
  - MPEC (Su-Judd and other papers by Che-Lin Su)
- These are not easy optimization problems
  - How can I find a good initial guess?
  - What solver to use?
  - What stopping rules to use?

Multidimensional Unconstrained Optimization: Comparison Methods

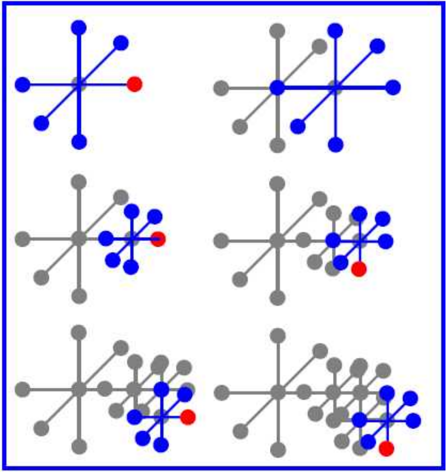
- Grid Search
  - Pick a finite set of points, X; for example, a Cartesian grid:

$$V = \{v_i | i = 1, ..., n\}$$
$$X = \{x \in \mathbb{R}^n | \forall i, x_i \in V\}$$

  - Compute $f(x)$, $x \in X$, and locate max
  - Grid search is often the first method to use.
    * Only involves function evaluations
    * It is embarassingly parallelizable
    * It should get you a good initial guess
  - A good initial guess is not critical for grid search, but is for all good algorithms
  - Grid search is slooooooow, so you should always switch to something better

- Random sampling
  - If sample is large enough then you will surely find a good initial guess
  - Of couse, if sample is large enough then you will have solved the problem by exhaustion
  - Remember Law of Large Numbers: error falls according to the negative square root of sample size.
  - Remember Central Limit Theorem: some regions will be poorly sampled
- Quasi-Random sampling
  - Designed to give you a uniform sample
  - Remember qMC theory: error is inversely proportional to sample size
- Parallelize, Parallelize, Parallelize

A *compass search* **example**



- Polytope Methods (a.k.a. Nelder-Mead, simplex, "amoeba")

Algorithm 4.3 Polytope Algorithm

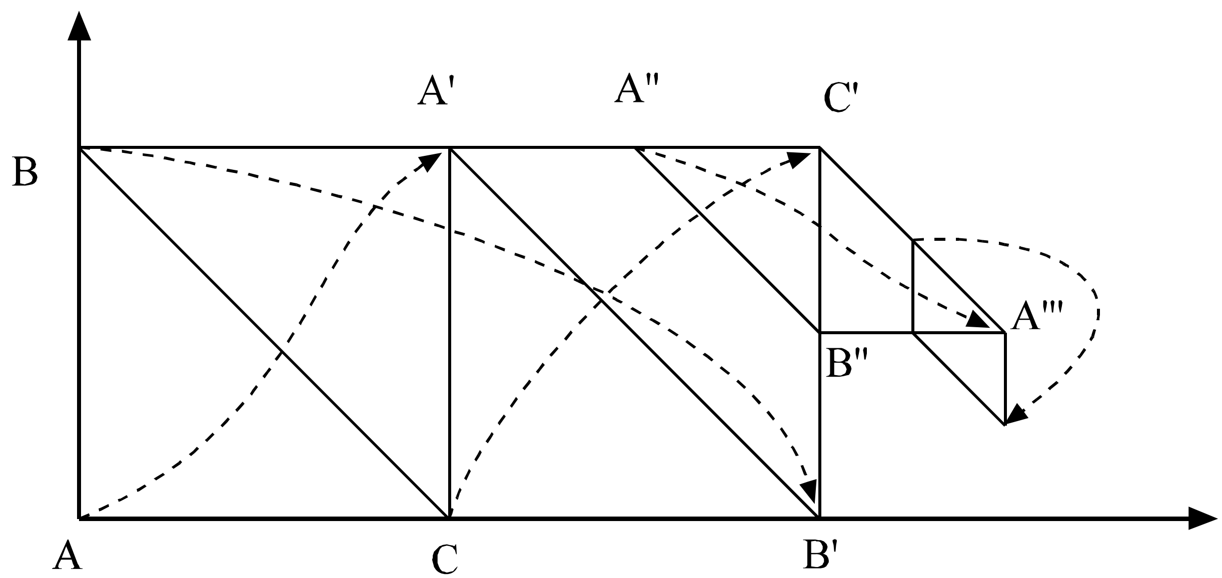
Initialize. Choose the stopping rule parameter ϵ. Choose an initial simplex $\{x^1, x^2, \cdots, x^{n+1}\}$.

Step 1. Reorder vertices so $f(x^i) \geq f(x^{i+1}),\ i = 1, \cdots, n$.

Step 2. Look for least i s.t. $f(x^i) > f(y^i)$ where y^i is reflection of x^i. If such an i exists, set $x^i = y^i$, and go to step 1. Otherwise, go to step 3.

Step 3. Stopping rule: If the width of the current simplex is less than ϵ, STOP. Otherwise, go to step 4.

Step 4. Shrink simplex: For $i = 1, 2, \cdots, n$ set $x^i = \frac{1}{2}(x^i + x^{n+1})$, and go to step 1.



Solving ill-conditioned problems via Proximal Point method

- Suppose you have an objective which has a singular Hessian at the minimum (or maximum).
  - Economic examples: Flat top of likelihood hill, flat bottom to a moments criterion minimum
- Newton's method may not properly converge for such problems
  - Round-off errors could cause convergence far from true solution
  - Any convergence will be slow.

Simple example

In[647]:=
```
a = 5; wgt =.; xold =.; yold =.
```

Suppose your objective is

In[648]:=
```
obj = (x + y - a)^4
```

Out[648]= $(-5 + x + y)^4$

There are multiple minima: any (x,y) such that x+y=5.

You can identify x+y but not (x,y)

In[649]:=
```
FindMinimum[obj, {x, 2}, {y, 2}]
```

Out[649]= $\{1. \times 10^{-16}, \{x \to 2.49995, y \to 2.49995\}\}$

This problem is so trivial and FindMinimum good enough that we get a solution. We stay with simple case to show basic idea.

So, suppose things did not go well.

Proximal Point method

Construct a penalty function

(xold, yold) is most recent guess

the penalty function is a quadratic penalty for choosing (x,y) different from (xold, yold)

In[650]:=
```
pen = (x - xold)^2 + (y - yold)^2
```
Out[650]= $(x - xold)^2 + (y - yold)^2$

Create a new objective function

In[651]:=
```
objProx = obj + wgt pen
```
Out[651]= $(-5 + x + y)^4 + wgt\left((x - xold)^2 + (y - yold)^2\right)$

objProx wants to minimize obj but imposes a cost for straying from (xold, yold)

We need to set the weight, and initial values for (xold, yold)

In[655]:=
```
wgt = 0.1;
xold = yold = 10;
```

In[657]:=
```
objProx
```
Out[657]= $0.1\left((-10 + x)^2 + (-10 + y)^2\right) + (-5 + x + y)^4$

Solve

In[658]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[658]= {x → 2.85478, y → 2.85478}

We get a solution. Let’s reset (xold, yold) and try again.

In[659]:=
```
{xold, yold} = {x, y} /. %
```
Out[659]= {2.85478, 2.85478}

In[660]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[660]= {x → 2.61451, y → 2.61451}

Repeat

In[661]:=
```
{xold, yold} = {x, y} /. %
```
Out[661]= {2.61451, 2.61451}

In[662]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[662]= {x → 2.56681, y → 2.56681}

In[663]:=
```
{xold, yold} = {x, y} /. %
```
Out[663]= {2.56681, 2.56681}

In[664]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[664]= {x → 2.54853, y → 2.54853}

In[665]:=
```
{xold, yold} = {x, y} /. %
```
Out[665]= {2.54853, 2.54853}

We now seemed to have become stuck. Remember that the weight is 0.1.
Let’s reduce the weight on the penalty

```
In[666]:= wgt = 0.001;

In[667]:= FindMinimum[objProx, {x, 2}, {y, 2}][[2]]

Out[667]= {x → 2.51304, y → 2.51304}
```

Progress! Let’s repeat this a few times

```
In[668]:= {xold, yold} = {x, y} /. %

Out[668]= {2.51304, 2.51304}

In[669]:= FindMinimum[objProx, {x, 2}, {y, 2}][[2]]

Out[669]= {x → 2.50716, y → 2.50716}

In[670]:= {xold, yold} = {x, y} /. %

Out[670]= {2.50716, 2.50716}

In[671]:= FindMinimum[objProx, {x, 2}, {y, 2}][[2]]

Out[671]= {x → 2.50507, y → 2.50507}

In[672]:= {xold, yold} = {x, y} /. %

Out[672]= {2.50507, 2.50507}
```

We could reduce the penalty weight further and get closer to some (x, y) such that x+y=5, but let's stop here.

What was the benefit of doing this?

Each step in the optimization problem was well-conditioned

Each step will converge quadratically to the solution of the penalized objective

You get arbitrarily close to some solution

You still cannot identify (x, y) but you can find a point that solves the problem

Identification

Economists are obsessed with identification

Why? No good reason.

My opinion: write down the model you think is valid and then let the computer tell you if you have identification.

MPEC estimation of demand curve

- Assume a conventional demand problem
  - utility function $u(c;\beta)$ with parameters β
  - price p is observed
  - demand c is observed with error $q = c + \varepsilon$.
  - the marginal utility of true consumption equals the price

$$u_c\left(q_i - \varepsilon_i; \beta\right) = p_i$$

- Therefore, the least squares estimation problem is

$$\begin{array}{cc} \min\limits_{c_i,\varepsilon_i,\beta} & \sum\limits_{i=1} \varepsilon_i^2 \\ \text{s.t.} & u_c\left(c_i;\beta\right) = p_i \\ & c_i = q_i - \varepsilon_i \end{array}$$

Challenges

- Need good initial guess for all variables
  - zero errors is natural, $\epsilon_i = 0$ with parameters β
  - initial guess for β is same here as with all other methods
  - choose β, and for each p_i compute $c_i = q_i$ from first-order condition; a large numb nonlinear equations even if you don't have a closed-form solution for demand
  - Conclusion: Initial guess for MPEC is no worse than for other approaches

- Solving constrained optimization problem may run into problems

$$\begin{aligned} \min_{c_i, \varepsilon_i, \beta} \quad & \sum_{i=1} \varepsilon_i^2 \\ \text{s.t.} \quad & u_c(c_i; \beta) = p_i \\ & c_i = q_i - \varepsilon_i \end{aligned}$$

- Relax. Give the equality constraints some breathing room

$$\begin{aligned} \min_{\lambda_i, c_i, \varepsilon_i, \beta} \quad & \sum_{i=1} \varepsilon_i^2 + P \sum_{i=1} \lambda_i \\ \text{s.t.} \quad & -\lambda_i \le p_i - u_c(c_i; \beta) \le \lambda_i \\ & c_i = q_i - \varepsilon_i \\ & \lambda_i \ge 0 \end{aligned}$$

where P is a penalty parameter and the λ_i variables are relaxation variables

- Advantages of relaxation
  - The relaxed problem always has feasible initial guesses
  - The true solution of the relaxed problem with high penalty is the same as the real pro
  - If things don't work, it probably is because you messed up on coding the constraints.