Computational Complexity

(Lectures on Solution Methods for Economists II: Appendix)

Jesús Fernández-Villaverde[1] and Pablo Guerrón[2]

February 18, 2018

[1]University of Pennsylvania

[2]Boston College

Computational complexity

- We now visit the main concepts of computational complexity.
- Discrete computational complexity deals with problems that:
  1. Can be described by a finite number of parameters.
  2. Can be solved in a finite number of step.
- It is based on the *Turing model of computation*.
- Assumes that a computer can compute a finite number of operations on a finite number of symbols (with unbounded memory and/or storage space).

A Turing machine

- A Turing Machine is a generic model of a computer that can compute functions defined over domains of finitely representative objects.
- A function is computable if exists a computer program that can compute $f(x)$ for any input x in a finite number of time using a finite amount of memory.
- Problem: it can take very long and a lot of memory.
- The theory of computational complexity classifies the problems in terms of time and memory needed.

A metric to measure complexity.

- Let us give a look to the traveling salesman problem.
    1. x is a finite list $\{c_1 \ldots c_n\}$ and a list of distance $\{d(c_i, c_j)\}$.
    2. f is an ordering $\{c_{\pi(1)} \ldots c_{\pi(n)}\}$ that minimizes the length of a trip from $\pi(1)$, visiting all the cities and ending at $\pi(1)$.
- The natural metric of "size" is n, the number of cities.
- $comp(n)$ returns the minimal time required by any algorithm to compute a problem of size n.

Polynomial-time versus exponential-time problems

- A polynomial-time problem has $comp\left(n\right) = O\left(P\left(n\right)\right)$ for some polynomial.
- For example, a multiplication of two $n \times n$ matrices has $comp\left(n\right) = O\left(n^{2376}\right)$.
- Polynomial-time problems are said to be tractable.
- If a problem is not bounded by any $P\left(n\right)$ is said to be an exponential-time problem.
- Exponential-time problems are said to be intractable.
- It will be shown that a $n-$dimensional DP problem is a polynomial-time problem.

Continuous computational complexity

- It deals with continuous mathematical problems.
- This type of problems cannot be solved exactly in a computer.
- We can only compute arbitrarily closed solutions.
- The theory of continuous computational complexity is based on the real number of model instead of the Turing model.

A real number of model

- A real number model is a machine that can compute infinite precision computations and store exact values of real numbers as $2^{1/2}$.
- It does not consider approximation error and/or round-off error.

Information-based complexity

- Since continuous time problems depend on an infinite number of parameters.
- Since a computer can only store a finite number of parameters, any algorithm trying to solve a has to deal with partial information.
- For example an integral.
- We have being able to characterize the complexity of some continuous problems as the DP with continuous state variables.

Theory of continuous computational complexity

- A continuous mathematical problem can be defined as:

$$\Lambda : F \to B$$

where F and B are infinite dimensional.

Example I: A multivariate integral

- For example:

$$\Lambda(f) = \int_{[0,1]^d} f(s)\,\lambda(ds)$$

where F and B are infinite dimensional.

- $B = R$ and

$$F = \left\{ f : [0,1]^d \rightarrow R | D^r f \text{ is continuos and } \|D^r f\| \leq 1 \right\}$$

where

$$\begin{aligned} \|D^r f\| &= \max_{k_1,\ldots,k_d} \sup_{s_1,\ldots,s_d} \left| \frac{\partial^r f(s_1,\ldots,s_d)}{\partial^{k_1} s_1 \ldots \partial^{k_d} s_d} \right| \\ r &= k_1 + \ldots + k_d \end{aligned}$$

Example II: A MPD problem

- F consists in all the pairs $f = (u, p)$.
- The operator $\Lambda : F \to B$ can be written as $V = \Lambda(u, p)$.
- Where in the finite case $V = (V_0, \ldots, V_T)$ as described in the recursive algorithm described the other day.
- An in the infinite order case V is the unique solution to the Bellman equation.

The approximation I

- Since $f \in F$ an infinite dimensional space, we can only compute an approximation using a computable mapping $U : F \to B$ can be computed only using a finite amount of information about f and can be computed using a finite number of algebraic operations.
- Given a norm in B, we can define $\|\Lambda(f) - U(f)\|$.
- $U(f)$ is an $\varepsilon-$approximation of f if $\|\Lambda(f) - U(f)\| \leq \varepsilon$.
- Let us analyze deterministic algorithms.
- $U : F \to B$ can be represented as the composition of:

$$U(f) = \phi_N(I_N(f))$$

  where $I_N(f) : F \to R^N$ maps information about f into R^N.
- In general $I_N(f) = (L_1(f), \ldots, L_N(f))$ where $L_i(f)$ is a functional of f.

The approximation II

- Consider $I_N(f) : F \rightarrow R$ and $L_i(f) = f(s_i)$ for some $s_i \in S$.
- In this case, $I_N(f)$ is called the standard information

$$I_N(f) = (f(s_1), \ldots, f(s_N))$$

  where $s_1, \ldots, s_N$ can be thought as the "grid points."
- $\phi_N(I_N(f))$ is a function that maps $I_N(f)$ into B.
- I_N, ϕ_N, and N are choice variables to get an accuracy ε.

The computational cost

- Call $c(U, f)$ the computational cost of computing and approximation solution $U(f)$.

$$c(U, f) = c_1(I_N, f) + c_2(\phi_N, I_N(f))$$

where $c_1(I_N, f)$ is the cost of computing f at $s_1, \ldots, s_N$ and $c_2(\phi_N, I_N(f))$ is the cost of using $f(s_1), \ldots, f(s_N)$ to compute $U(f) = \phi_N(I_N(f))$.

The computational cost of example

- The multivariate integration problem.
- Step 0: Chose $s_1, \ldots, s_N$ in $[0,1]^d$.
- Step 1: Calculate $f(s_1), \ldots, f(s_N)$.
- Step 2:

$$\phi_N(I_N(f)) = \frac{\sum_{i=1}^{N} f(s_i)}{N}$$

- It can be shown that in this case $c_1(I_N, f)$ and $c_2(\phi_N, I_N(f))$ are proportional to N.

ε-complexity

- $\varepsilon -$ *Complexity* is the minimal cost of computing an $\varepsilon-$approximation to $\Lambda(f)$.
- *The worst case deterministic complexity* of a problem Λ is:

$$comp^{wor-\det}(\varepsilon) = \inf_{U} \{c(U) \,|\, e(U) \leq \varepsilon\}$$

where

$$c(U) = \sup_{f \in F} c(U, f)$$

$$e(U) = \sup_{f \in F} \|\Lambda(f) - U(f)\| .$$

- For the multivariate integration problem, it can be shown that

$$comp^{wor-\det}(\varepsilon) = O\left(\frac{1}{\varepsilon^{d/r}}\right)$$

- Given Θ, ε, and r, exponential function on $d \to$ curse of dimensionality.
- Chow and Tsitsiklis (1989,1991) show that the MPD problem is also subject to the course of dimensionality.

Random algorithms

- Random algorithms break the course of dimensionality.
- $\widetilde{U} : F \to B$ can be represented as the composition of:

$$\widetilde{U}(f) = \widetilde{\phi}_N\left(\widetilde{I}_N(f)\right)$$

  where $\widetilde{I}_N(f)$ is a random information operator nd $\widetilde{\phi}_N\left(\widetilde{I}_N(f)\right)$ is a random algorithm.

- The multivariate integration problem:
  1. *IID* draws $\widetilde{s}_1, \ldots, \widetilde{s}_N$ from $[0,1]^d$.
  2. Calculate $f(\widetilde{s}_1), \ldots, f(\widetilde{s}_N)$.
  3. We compute:

$$\phi_N(I_N(f)) = \frac{\sum_{i=1}^{N} f(\widetilde{s}_i)}{N}$$

ε-complexity of random algorithms I

- $\widetilde{U}$ is a random variable.
- Let us define the underlying probability space $(\Omega, Borel\,(\Omega)\,, \mu)$.
- $\widetilde{I}_N : \Omega \to R^N$.
- $\widetilde{\phi}_N : \Omega \times R^N \to B$.
- So that $\widetilde{U}$ is a well-defined random variable for each $f \in F$.
- *The worst case randomized complexity* of a problem Λ is:

$$comp^{wor-ran}\left(\varepsilon\right) = \inf_{\widetilde{U}} \left\{ c\left(\widetilde{U}\right) | e\left(\widetilde{U}\right) \le \varepsilon \right\}$$

where

$$e\left(\widetilde{U}\right) = \sup_{f \in F} \int \left\| \Lambda\left(f\right) - \widetilde{U}\left(\omega, f\right) \right\| \mu\left(d\omega\right)$$

$$c\left(\widetilde{U}\right) = \sup_{f \in F} \int c\left(\widetilde{U}\left(\omega, f\right), f\right) \mu\left(d\omega\right).$$

ε-complexity of random algorithms II

- For the multivariate integration problem, it can be shown that

$$comp^{wor-ran}(\varepsilon) = O\left(\frac{1}{\varepsilon^2}\right).$$

- There is not course of dimensionality.
- However: are random variables really random?
- We know that this is not the case: we only have pseudo-random numbers.
- Therefore, random algorithms are deterministic algorithms.
- A problem?

Random algorithms not always a solution

- Sometimes even random algorithms cannot solve the course of dimensionality when we evaluate algorithms using the worse case.
- Examples nonlinear optimization and the solution to PDE.
- An option is to evaluate the algorithm on basis of the average rather than the worst case.
- *The average case deterministic complexity* of a problem Λ is:

$$comp^{ave-ran}\left(\varepsilon\right) = \inf_{U} \left\{ c\left(U\right) \middle| e\left(U\right) \leq \varepsilon \right\}$$

where

$$e\left(U\right) = \int \left\| \Lambda\left(f\right) - U\left(f\right) \right\| \mu\left(df\right)$$

$$c\left(U\right) = \int c\left(U\left(f\right), f\right) \mu\left(df\right).$$

- Why deterministic? They are equivalent.
- It is difficult to define priors: μ.