Integral equations, eigenvalue, function interpolation

Marcin Chrząszcz
mchrzasz@cern.ch

University of Zurich[UZH]

Monte Carlo methods,
26 May, 2016

Integral equations, introduction

⇛ Fredholm integral equation of the second order:

$$\phi(x) = f(x) + \int_a^b K(x,y)\phi(y)dy$$

⇛ The f and K are known functions. K is called kernel.

⇛ The CHALLENGE: find the ϕ that obeys the above equations.

⇛ There are NO numerical that can solve this type of equations!

⇛ Different methods have to be used depending on the f and K functions.

⇛ The MC algorithm: construct a probabilistic algorithm which has an expected value the solution of the above equations. There are many ways to build this!

⇛ We assume that the Neumann series converges!

Integral equations, approximations

⇛ The following steps approximate the Fredholm equation:

$$\phi_0(x) = 0, \qquad \phi_1(x) = f(x) + \int_a^b K(x,y)\phi_0(y)dy = f(x)$$

$$\phi_2(x) = f(x) + \int_a^b K(x,y)\phi_1(y)dy = f(x) \int_a^b K(x,y)f(y)dy$$

$$\phi_3(x) = f(x) \int_a^b K(x,y)dy = f(x) + \int_a^b K(x,y)f(y)dy + \int_a^b \int_a^b K(x,y)K(y,z)f(z)$$

⇛ Now we put the following notations:

$$K^{(1)} = K(x,y) \qquad K^{(2)}(x,y) = \int_a^b K(x,t)K(t,y)dt$$

⇛ One gets:

$$\phi_3(x) = f(x) + \int_a^b K^{(1)}(x,y)f(y)dy + \int_a^b K^{(2)}(x,y)f(y)dy$$

Integral equations, approximations

⇛ Continuing this process:

$$K^{(n)}(x,y) = \int_a^b K(x,t)K^{(n-1)}(t,y)dt$$

and the n-th approximation:

$$\phi_n(x) = f(x) + \int_a^b K^{(1)}(x,y)f(y)dy + \int_a^b K^{(2)}(x,y)f(y)dy + \\ ... + \int_a^b K^{(n)}(x,y)f(y)dy$$

⇛ Now going with the Neumann series: $n \to \infty$:

$$\phi(x) = lim_{n\to\infty}\phi_n(x) = f(x) + \sum_{i=1}^{n} \int_a^b K^{(n)}(x,y)f(y)dy$$

⇛ The above series converges only inside the square: $a \leqslant x,\ y \leqslant b$ for:

$$\int_a^b \int_a^b |K(x,y)|^2 dxdy < 1$$

Integral equations, algorithm

⇛ The random walk of particle happens on the interval (a, b):

- In the $t = 0$ the particle is in the position $x_0 = x$.
- If the particle at time $t = n - 1$ is in the x_{n-1} position then in time $t = n$ the position is: $x_n = x_{n-1} + \xi_n$. The numbers $\xi_1, \ \xi_2, ...$ are independent random numbers generated from ρ p.d.f..
- The particle stops the walk once it reaches the position a or b.
- The particle life time is n when $x_n \leqslant a$ and $x_n \geqslant b$.
- The expected life time is given by the equation:

$$\tau(x) = \rho_1(x) + \int_a^b [1 + \tau(y)] \, \rho(y - x) dy$$

where:

$$\rho_q(x) = \int_{-\infty}^{a-x} \rho(y) dy + \int_{b-x}^{\infty} \rho(y) dy$$

is the probability of particle annihilation in the time $t = 1$.

Integral equations, algorithm 2

⇛ The above can be transformed:

$$\tau(x) = 1 + \int_a^b \tau(y)\rho(x-y)dy \tag{1}$$

⇛ Now if $p(x)$ is the probability that the particle in time $t=0$ was in position x gets annihilated because it crosses the border a.
⇛ The probability obeys the analogous equation:

$$p(x) = \rho(x) + \int_a^b p(y)\rho(y-x)dy \tag{2}$$

where

$$\rho(x) = \int_{-\infty}^{a-x} \rho(y)dy$$

is the probability of annihilating the particle in the first walk.
⇛ For the functions τ and ρ we got the integral Fredholm equation.
⇛ So the above random walk can be be used to solve the Equations 1 and 2.

Integral equations, algorithm 3

⇛ The $\rho(x)$ is the p.d.f. of random variables ξ_n.
⇛ We observe the random walk of the particle. The trajectory: $\gamma = (x_0, x_1, x_2, ..., x_n)$. This means for $t = 0, 1, 2..., n-1$ and $x_n \leqslant a$ or $x_n \geqslant b$. Additionally we mark: $\gamma_r = (x_0, x_1, ..., x_r),\ r \leqslant n$.
⇛ We defined a random variable:

$$S(x) = \sum_{r=1}^{n} V(\gamma_r) f(x_{r-1})$$

where

$$V(\gamma_0) = 1,$$
$$V(\gamma_r) = \frac{K(x_{r-1}, x_r)}{\rho(x_r - x_{r-1})} V(\gamma_{r-1})$$

⇛ One can prove that $E\left[S(x)\right]$ treated as a function of x variable is the solution to the integral equation.

Integral equations, algorithm 4

⇛ We define a new random variable:

$$c_r(x) = \begin{cases} \frac{V(\gamma_{n-r} f(x_{n-r}}{\rho(x_{n-r}}, & r \leqslant n, \\ 0, & r > n \end{cases}$$

where $\rho_r(x)$ is defined as:

$$\rho_1(x) = \int_{-\infty}^{a-x} \rho(y)dy + \int_{b-x}^{+\infty} \rho(y)dy,$$

$$\rho_r(x) = \int_a^b ... \int_a^b \rho(x_1 - x)\rho(x_2 - x)...\rho(x_{r-1} - x_{r-2})\rho_1(x_{r-1})dx_1...dx_{r-1}$$

is the probability that the particle that is at given time in the x coordinate will survive r moments.

⇛ One can prove that $E\left[c_r(x)\right]$ treated as a function of x variable is the solution to the integral equation.

Integral equations, general remark

There is a general trick:

Any integral equation can be transformed to linear equation using quadratic form. If done so one can use the algorithms form lecture 8 to solve it. Bullet prove solution!

Eigenvalue problem

⇛ The Eigenvalue problem is to find λ that obeys the equation:

$$H\overrightarrow{x} = \lambda\overrightarrow{x}$$

⇛ For simplicity we assume there the biggest Eigenvalue is singular and it's real.

⇛ The numerical method is basically an iterative procedure to find the biggest Eigenvalue:

- We choose randomly a vector $\overrightarrow{x}_0$.
- The m vector we choose accordingly to formula:

$$\overrightarrow{x}_m = H\overrightarrow{x}_{m-1}/\lambda_m$$

where λ_m is choose such that

$$\sum_{j=1}^{n} |(\overrightarrow{x}_m)_j| = 1$$

the $(\overrightarrow{x})_j$ is the j coordinate of the $\overrightarrow{x}$ vector, $j = 1, 2, 3, ..., n$

⇛ The set λ_m is converging to the largest Eigenvalue of the H matrix.

Eigenvalue problem

⇛ From the above we get:

$$\lambda_1\lambda_2...\lambda_m(\overrightarrow{x}_j) = (H^m\overrightarrow{x}_0)_j; \quad \lambda_1\lambda_2...\lambda_m = \sum_{j=1}^{n}(H^m\overrightarrow{x}_0)_j$$

⇛ For big k and $m > k$ one gets:

$$\frac{\sum_{j=1}^{n}(H^m\overrightarrow{x}_0)_j}{\sum_{j=1}^{n}(H^k\overrightarrow{x}_0)_j} = \lambda_{k+1}\lambda_{k+2}...\lambda_m \approx \lambda^{m-k}$$

from which:

$$\lambda \approx \left[\frac{\sum_{j=1}^{n}(H^m\overrightarrow{x}_0)_j}{\sum_{j=1}^{n}(H^k\overrightarrow{x}_0)_j}\right]^{\frac{1}{m-k}}$$

⇛ This is the Eigenvalue estimation corresponding to $H^m\overrightarrow{x}_0$ for sufficient large m.

Eigenvalue problem, probabilistic model

⇛ Let $Q = (q_{ij})$, $i, j = 1, 2, ..., n$ is the probability matrix:

$$q_{ij} \geqslant 0, \quad \sum_{j=1}^{n} = 1$$

⇛ We construct a random walk on the set: $\{1, 2, ...n\}$ accordingly to the above rules:

- In the $t = 0$ the particle is in a randomly chosen state i_0 accordingly to binned p.d.f.: p_j.
- If in the moment $t = n - 1$ the particle is in i_{n-1} state then in the next moment it goes to the state i_n with the probability $q_{i_{n-1}j}$.
- For $\gamma = (i_0, i_1, ...)$ trajectory we define a random variable:

$$W_r(\gamma) = \frac{(\overrightarrow{x})_{i_0}}{p_{i_0}} \frac{h_{i_1 i_0} h_{i_2 i_1} h_{i_3 i_2} ... h_{i_r i_{r-1}}}{q_{i_1 i_0} q_{i_2 i_1} q_{i_3 i_2} ... q_{i_r i_{r-1}}}$$

⇛ Now we do:

$$\frac{E\left[W_m(\gamma)\right]}{E\left[W_k(\gamma)\right]} \approx \lambda^{m-k}$$

⇛ So to estimate the largest Eigenvalue:

Function interpolation

⇛ Lets put $f(x_1) = f_1,\ f(x_2) = f_2$, which we know the functions.
⇛ The problem: calculate the $f(p)$ for $x_1 < p < x_2$.
⇛ From the interpolation method we get:

$$f(p) = \frac{p - x_1}{x_2 - x_1} f_2 + \frac{x_2 - p}{x_2 - x_1} f_1$$

⇛ I am jet-lagged writing this so let me put: $x_1 = 0$ and $x_2 = 1$:

$$f(p) = (1 - p) f_1 + p f_2$$

⇛ For 2-dim:

$$f(p_1, p_2) = \sum_{\delta} r_1 r_2 f(\delta_1, \delta_2)$$

where:

$$r_i = \begin{cases} 1 - p_1, & \delta_i = 0 \\ p_i, & \delta_i = 1 \end{cases}$$

⇛ the sum is over all pairs (in this case 4).

Function interpolation

⇛ For n-dim we get a monstrous:

$$f(p_1, p_2, ..., p_n) = \sum_{\delta} r_1 r_2 ... r_n f(\delta_1, ..., \delta_n)$$

the sum is over all combinations $(\delta_1, ..., \delta_n)$, where $\delta_i = 0, 1$.
⇛ The above sum is over 2^n terms and each of it has $(n+1)$ temrs. It's easy to imagine that for large n this is hard... Example $n = 50$ then we have 10^{14} ingredients.
⇛ There has to be a better way to do this!
⇛ From construction:

$$0 \leqslant r_1 r_2 ... r_n \leqslant 1, \qquad \sum_{\delta} r_1 r_2 ... r_n = 1$$

⇛ We can treat the r_i as probabilities! We define a random varaible: $\xi = (\xi_1, ..., \xi_n)$ such that:

$$\mathcal{P}(\xi_i = 0) = 1 - p_i, \qquad \mathcal{P}(\xi_i = 1) = p_i$$

The extrapolation value is then equal:

$$f(p_1, p_2, ..., p_n) = E\left[f(\xi_1, ..., \xi_n)\right]$$

Travelling Salesman Problem

- Salesman starting from his base has to visit $n-1$ other locations and return to base headquarters. The problem is to find the shortest way.
- For large n the problem can't be solver by brutal force as the complexity of the problem is $(n-1)!$
- There exist simplified numerical solutions assuming factorizations. Unfortunately even those require anonymous computing power.
- Can MC help? YES :)
- The minimum distance l has to depend on 2 factors: P the area of the city the Salesman is travelling and the density of places he wants to visit: $\frac{n}{P}$
- Form this we can assume:

$$l \sim P^{a}\left(\frac{n}{P}\right)^{b} = P^{a-b}n^{b}.$$

Traveling Salesman Problem

- From dimension analysis:

$$a - b = \frac{1}{2}.$$

- To get l we need square root of area.
- From this it's obvious:

$$l \sim P^a (\frac{n}{P})^b = P^{0.5} n^{a-0.5}.$$

- Now we can multiply the area by alpha factor that keeps the density constant then:

$$l \sim \alpha^0.5\alpha 6a - 0.5 = \alpha^a$$

- In this case the distance between the clients will not change, but the number of clients will increase by α so:

$$l \sim \alpha$$

- In the end we get: $a = 1$

Traveling Salesman Problem

- In total:

$$l \sim k(nP)^{0.5}$$

- Of course the k depends on the shape of the area and locations of client. However for large n the k starts loosing the dependency. It's an asymptotically free estimator.
- To use the above formula we need to somehow calculate k.
- How to estimate this? Well make a TOY MC: take a square put uniformly n points. Then we can calculate l. Then it's trivial:

$$k = l(nP)^{-0.5}$$

Traveling Salesman Problem

- This kind of MC experiment might require large CPU power and time. The adventage is that once we solve the problem we can use the obtained k for other cases (it's universal constant!).
- It turns out that:

$$k \sim \frac{3}{4}$$

- Ok, but in this case we can calculate l but not the actual shortest way! Why the hell we did this exercise?!
- Turns out that for most of the problems we are looking for the solution that is close to smallest l not the exact minimum.

War Games

- S. Andersoon 1966 simulated for Swedish government how would a tank battle look like.
- Each of the sides has 15 tanks. that they allocate on the battle field.
- The battle is done in time steps.
- Each tank has 5 states:
  - OK
  - Tank can only shoot
  - Tank can only move
  - Tank is destroyed
  - Temporary states
- This models made possible to check different fighting strategies.

Backup