Name: CHAN King Yeung
SID: 1155119394
STAT3011 Individual Report

Root finding (Bisection, Newton-Raphson)

The root of an equation has a wide variety of applications in science subjects. We often interested in finding the root, a value of x such that $f(x) = 0$, to solve problems. For example, we need to find the maxima to ensure whether the maximum likelihood estimator (MLE) is maximised. Here we can adopt root-finding techniques to find the maxima of the inequality. During our life in secondary schools, we have learnt the simplest algorithm to find the roots of a quadratic equation. However, in practice, things are more complicated and such an algorithm may not apply to other equations. Hence, we will discuss 2 root-finding algorithms, which are the bisection method and the Newton-Raphson method, and understand how they work.

Bisection method relies on the intermediate value theorem (IVT), which guarantees that there is at least one value c_0 in $[a, b]$ such that $f(c_0) = 0$, given $f(x)$ is a continuous function on close internal $[a, b]$ and $f(a) \cdot f(b) < 0$. The procedure of find root is very similar to binary search. The method iteratively divides the interval into 2 parts and checks if the solution is satisfied by the true root or the defined tolerance ε. More precisely, the method calculates the middle point of the interval $c \coloneqq \frac{a+b}{2}$ and checks whether the function at $x = c$ satisfy the condition of $f(c) = 0$ or $|f(c)| < \varepsilon$. If the condition is not satisfied, then the algorithm redefines the middle point by fulfilling the condition of VIT, which based on the fact that either $f(a) \cdot f(c) < 0$ or $f(c) \cdot f(b) < 0$. The above process is repeated until the true root is found or the solution is accepted by the defined tolerance. The bisection method ensures solution will fall in between the range of $[a, b]$ whatever how many iterations. It also guarantees that there is a solution output from the algorithm if the function satisfies IVT. Although the method is robust, however, the algorithm can only return a single solution whatever there are multiple roots in the function. We need to define the range $[a, b]$ before running the programme, and such a result will only give us a rough insight into the solution. Moreover, the time complexity is $O(n)$, which may not as good as other root-finding algorithms.

By saying that, the Newton-Raphson (NR) method has a better performance than the bisection method. The NR method finds a value x_i in a non-linear equation $f(x)$, given $f(x)$ is continuous and differentiable, such that $f(x_i) = 0$ or approximate to the true root. For $f(x)$ at the initial guess point x_0, there is a tangent line pass through the point $\big(x_0, f(x_0)\big)$. By Taylor expansion, we obtain the result of $f(x_i) \approx f(x_{i-1}) + (x_i - x_{i-1}) f'(x_{i-1})$. The NR method uses the tangent line at x_{i-1} and rearranges the equation into $x_i \coloneqq x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$ to find the next possible root. Simply put, the NR method sets x_0 as the initial guess of the root and computes x_i to test $f(x_i)$ repeatedly until the true root is found or the solution is agreed by the defined tolerance. The NR method is more popular as the time complexity is $O\big(log(n)\, F(n)\big)$, $F(n)$ for calculating $\frac{f(x_{i-1})}{f'(x_{i-1})}$ with n-digit precision, which the performance is much faster than the bisection method to obtain the result. The NR method is important since it can easily generalise to a complicated bivariate function. For instance, the NR method converges exactly the true maximum and to a local minimum. However, the NR method is only applicable for differentiable function, and it may diverge if the initial guess of the root is far away from the true root. Such an issue requires a precise initial guess before running the programme. Moreover, points of inflection, local maxima or minima, will lead to an infinite circle of tangent lines during the iterations, and it will oscillate without converging to a root.

As mentioned previously, MLE can be reduced to a root-finding problem. Root finding plays an important role in verifying the theories of statistical inference. For example, we need to find the maxima to establish MLE for further statistical inference. Without finding the root, or maxima, we cannot make use of MLE to construct point estimate or confidence interval for parameter easily.

Optimisation (genetic algorithm)

Genetic Algorithms (GA) are a robust mathematical model that makes use of the concept of natural selection, parts of evolutionary algorithms. Natural selection is the mechanism of evolution. By saying that, GA evaluates a set of outputs based on the process of natural selection, and it often appears in solving the optimisation problems. For those species (solutions) who can adapt to changes in the environment (obtaining the highest value of fitness) will have the rights to survive and reproduce (retain itself and generate similar solutions) in the next generation. In other words, the process of evolution is a method to search for extrema of a function.

In the beginning, the population consists of a set of species come from nowhere. Each of them has its chromosomes. The population is the source to reproduce the next generation, and the next generation is expected to have better abilities in surviving in the environment. However, as Darwin mentioned, there are 3 criteria for the evaluation, which are selection, crossover, and mutation. Species are being selected based on the survival of the fittest they have. Those selected species have more change to reproduce more offspring, and they will obtain some chromosomes from their parents during the process of crossover. In addition to crossover, each offspring has a small change to get a mutation. The ideal reason for mutation is to discover some new ways to obtain better fitness in a change of environment.

In reality, it is not very often to find the optimum solution to a problem, but we still want to find the relative best solution to the problem. Such an issue motivates how GA works to achieve feasible solutions. GA starts with a set of random solutions, which represents points in search space and namely individuals. Each individual is evaluated with a score of fitness to the function. To obtain a better set of solutions, GA generates another set of solutions as the process of optimising the function. Individuals with high scores of fitness will have a higher chance to reproduce new solutions into the next population. With a crossover probability defined beforehand, new solutions (offspring) will have some chromosomes from the original solutions (parents). There is a case that some original solutions are directly copied into the next population. Moreover, there is a mutation probability that new solutions may have some new structure of chromosomes which do not appear in the previous set of solutions. After processing "natural selection", GA obtains a new set of solutions, and it will be repeated until the condition, obtaining the extrema to the function, is satisfied.

Indeed, there are many more details about how GA works with the operations of selection, crossover and mutation. For example, the Roulette Wheel is one of the most widely adopted selection methods in GA. The structure of chromosomes is also a crucial idea to understand how new solutions are generated. However, things are not easy to explain in a short paragraph. We are going to omit the details here since the abstract functions in `library(GA)` allow us to ignore the mechanisms behind the algorithm.

GA differs from traditional optimisation algorithms. It generates solutions at random, which guarantees that there is a set of solutions being outputted from the algorithm. Those solutions are evaluated by their fitness to the function, which is realisable even though the solutions are at random. However, GA takes a reference to evolution, which is related to the time for evaluating a new set of solution. The time complexity is $O(Fitness \times mutation) + O(crossover)$. It is time-consuming for complicated functions to obtain a set of solutions. The performance of GA depends on what operators it adopted. For example, as mentioned previously, the Roulette Wheel select solution based on their fitness score. In some cases when individuals have a significant difference in their fitness, than those with low fitness score are facing the problem of extinction. It may result in some imperfect of solutions. If GA adopts the Rank Selection, all individuals have a chance of being selected. However, time consumption of such method will become a concern for the researchers.

Finding an extremum in a function is very important to statistical inference. Without understanding, we may not construct some statistical model. For instance, function $g(\cdot)$ is a convex function in $\boldsymbol{\beta}$, and we need to find a point to minimise the $g(\cdot)$, such that we can obtain OLS estimate $\widehat{\boldsymbol{\beta}}$ for regression analysis. Optimisation algorithms are useful for such applications or some other advanced statistical models.

Numerical integration (Quadrature)

During the first year of study in calculus, we learnt definite integral is used to calculate the total signed area under some intervals. For one dimensional integral, numerical integration derives the integrand into many small rectangles and combines their values to obtain the approximation to the integral. By underlining that, the quadrature rule is a kind of method to make such approximation to the definite integral. The quadrature rule is defined as $\int_a^b f(x)\,dx = \sum_{j=1}^{n+1} w_j F(s_j) + E(F)$, where w_j is the weighted coefficient, s_j is the quadrature point, and $E(F)$ is the error term due to the approximation. The integrand is derived into n number of integration points, such that the quadrature rule can make use of polynomial interpolation for those integration points to approximate the integral. Here we will focus on the rectangle rule and the trapezoidal rule to evaluate the approximation of the integrand.

The rectangle rule is the naïve way to evaluate the integrand by defining each interpolating function as a constant function, where the polynomial has a degree of zero. It can be stated as $\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{a+b}{2}\right)$ and passes through the midpoint of each interval, which is $\left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right)$. As the rectangle rule is approximating the signed area of an interval $[a, b]$, applying the rectangle rule to each subinterval from the derived integrand and summing all the results will return the final approximation of the integrand.

The trapezoidal rule is very similar to the rectangle rule, but the trapezoidal rule evaluates the integrand by an affine function instead of constant function for interpolating function, which has one degree in the polynomial. It can be simply as $\int_a^b f(x)\,dx \approx (b-a)\frac{f(a)+f(b)}{2}$ and passes through the points at $\big(a, f(a)\big)$ and $\big(b, f(b)\big)$. Again, since the integrand has been derived into n portions, employing the trapezoidal rule to each subinterval will generate the final approximation of the integrand.

The quadrature rule is the easiest way to approximate the definite integral and obtaining the optimal for improper integrals. To find the approximate of the definite integral, we can even calculate with our pen and paper for the simple numerical integration. However, the approximate error depends on the number of interpolation functions. The more interpolation functions, the smaller approximation error. Since no one wants a solution that is not precise, a large number of subintervals is required in the algorithm. The time complexity is $O(n^2)$ which requires a quite long time to run the algorithm. Indeed, the performance of the quadrature rule depends on the number of subintervals. The weights of the rule are the major coefficient that determines how large the error term. To reduce the amount of error, a large number of interpolation functions is needed.

Statistical distributions, such as Normal distribution and t-distribution, are said to be the core part of the statistic. Numerical integration plays an important role here to evaluate the statistic for those distributions and make an inference to the parameter. It evaluates how the mean and variance function looks like, even calculating the probability under some continuous function.

Monte Carlo integration (distribution sampling, thus need to learn one or two sampling methods)

The quadrature rule is very good in approximating the integrand in a user-friendly way. However, it is limited to one-dimensional problems since it is difficult to generalise the concept into multidimensional problems. Such an issue motivates how Monte Carlo (MC) evaluate the approximation for integration. Rather than using interpolation functions, MC simulates a set of random points, which agreed by the target function, to provide generally approximate solutions. The average of those points estimates the definite integral, given the fact that the sample mean approximates the population mean by the strong law of large number. Samples are drawn from some distribution $P(X)$ which is normalised such that $\int P(X)\,dX = 1$ to ensure all points within the support of $f(X)$ have a chance being selected. Thus, MC integration can be written as $\int f(X)P(X)\,dX = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$. So far so good we know the basic procedure of MC integration, but what is the distribution for sampling? Here we will consider the general MC method and Markov Chain MC (MCMC) method.

The general MC method uses either transformation method or rejection method to simulate random points. The meaning of transformation is to make use of the inverse cumulative distribution function F^{-1} to draw a sample from the function $f(x)$. By restricting $F(x) = \int_{-\infty}^{x} f(t)\,dt$ into $0 \leq F(y) \leq 1$, it allows the computer to simulate random points from $u \sim Unif(0,1)$ such that $X = F^{-1}(u)$ is the desired points of the generation from $f(x)$. However, the transformation is applicable only when invertible. Thus, the rejection method is used instead. The general idea of the method is to discover a comparison function $g(x)$, where $g(x) \geq f(x)\ \forall x$, to simulate random points. It generates a random sample from $g(x)$, says x_0. The algorithm draws another random sample from $Uinf(0, g(x_0))$, says y_0. The random sample x_0 is said to be accepted when the condition, $y_0 < f(x_0)$, is satisfied. The method simulates random samples until it has enough accepted x_0's to evaluate the definite integral. If $g(x)$ is far away from $f(x)$, the rejection method may be a good choice to adopt.

MCMC is a robust method to generate a set of consistent samples from any given distribution. There is no requirement for the specific knowledge about the distribution and support of the function. Suppose $f(x)$ is the function that we are interested to find the definite integral. The method defines a proposal density $g(x)$ with an initial value x_0 to generate new steps for the Markov chain. Metropolis-Hasting MCMC is a common method used to simulate a sequence of random samples from $g(x)$. It computes a candidate step which defined as $x' = x_i + \varepsilon$ and $\varepsilon \sim g(x)$. Again, the algorithm simulates a random point from $u \sim Unif(0,1)$ to determine whether x' is accepted as the next step of not with a probability $A(x', x_t) = \min\left(1, \frac{f(x')}{f(x_t)}\frac{g(x_t|x')}{g(x'|x_t)}\right)$. If $u \leq A(x', x_t)$, the next step will be x'. Otherwise, the next step will be the old state forward, say $x_{t+1} = x_t$.

MC is very useful when the distribution is complicated to evaluate the definite integral, even in one-dimensional problems. It is less hampered by the complex domain and high-dimensional situation. However, MC requires a large sample size to make the approximation accurate. In other words, the time complexity is $O(n)$ which means the more random points being simulated, the longer time in running the algorithm. One thing needs to be emphasised here is that the error of the output can be reduced in terms of $\frac{1}{\sqrt{N}}$ which the performance of MC can be measured.

Similar to the quadrature rule, MC acts very important in the view of statistic. MCMC is part of a series on Bayesian statistics. In calculating multidimensional integrals, such as hierarchical models mentioned in Bayesian statistics, can apply MCMC to find the approximation of the model.