Tarea #10

Joel Chacón Castillo

Abstract—El método simplejo de Nelder-Mead es un proceso de optimización que trabaja bien con funciones objetivo de forma regular. Para una función de n parámetros, compara la función objetivo de los $n+1$ vértices de un simplejo y actualiza el peor vértice a traves búsqueda de pasos del simplejo.

1 Introduction

El método de Nelder y Mead (1965) es un procedimiento de optimziación el cual se utiliza en funciones irregulares. Es un método robusto que puede superar los mínimos locales, indeterminaciones y discontinuidades en funciones que han sido evaluadas.

2 Teoria

2.1 Algoritmo Nelder-Mead

La idea detràs del algoritmo Nelder-Mead es rastrear hacia el valor mínimo moviendo en un simplejo un vértice a la vez. Los vértices se mueven realizando una operación a ala vez. Los vértices son movidos efectuando cuatro operaciones básicas: Reflexión, Expansión, Contracción y Múltiple Contracción Figura (2.1).

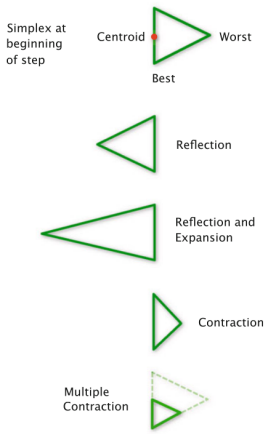

Fig. 1. Operaciones básicas del método simplex.

2.2 Operaciones bàsicas

Suponiendo que se quiere minimizar f, una funciòn de dos paràmetros θ_1 y θ_2

$$f(\theta_1, \theta_2) = |\theta_1| + |\theta_2| \tag{1}$$

La soluciòn es trivial con los valores $(\theta_1, \theta_2) = (0, 0)$.

2.2.1 *Inicializar el simplejo*

Dado un valor inicial, el mètodo crea un simplejo agregando un paso de la magnitud dada que pertenece a cada paràmetro. Para n paràmetros, un simplejo es un politopo el cual es convexo en sus $n+1$ vértices. Para dos parámetros de minimzación el simplejo es un triángulo. Para el punto inicial el método puede crear otros vértices agregando un ruído mínimo en su respectivo componente del punto.

2.2.2 *Ordenar lso vèrtices y calcular el centroide*

Evaluar la funciòbjetivo en todos los vèrtices y ordernarlos en orden creciente. Posteriormente se calcula el centroide de todos los vèrtices excepto del peor.

2.2.3 *Actualizar el peor simplejo o reducir el simplejo*

Reflejar el peor vèrtice, si la funciòn objetivo tiene un valor menos al vèrtice reflejado que el mejor vèrtice actual, entonces el algoritmo refleja y expande el vértice el cual regresa al paso 2. Si no es una mejora sobre el mejor vértice pero es mejor que el segundo peor vértice, entonces el vértice reflejado reemplaza el peor vértice el algoritmo regresa al paso 2. Si el vértice reflejado no es una mejora sobre el segundo peor vértice, el algoritmo trata de contraerse. Si la contracción es mejor que el peor vértice, este reemplaza el peor vértice con el vértice contraído y el algoritmo regresa al paso 2, de otra forma si el vértice contraído es peor que el peor vértice, el algoritmo contrae el simplejo entero alrededor del mejor vértice. El algoritmo regresa al paso 2 e itera hasta que algún criterio de convergencia sea cumplido.

2.3 Función objetivo

La función objetivo es conocida como la función de Rosenbrock

$$f(x) = \sum_{i=1}^{d-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \qquad dondex = [x_1, ..., x_N] \in \Re^d \tag{2}$$

2.4 Algoritmo

El criterio de paro que se utiliza regularmente es que la norma de la diferencia entre la peor solución X_n y el promedio de las restantes en menos que una tolerancia ($\epsilon = 10^{-4}$.

2.4.1 Paràmetros

n = Nùmero de variables.
δ_i = Tamaño de paso (se escoge suficientemente grande deacuerdo al dominio del problema)
α = Paràmetro de reflexiòn (se sugiere 1).
γ = Paràmtro de expansiòn (se sugiere 1).
β = Paràmetro de contracciòn (se sugiere 0.5),
τ = Paràmetro de contracciòn al generar el nuevo simplejo (se sugiere 0.5).

2.5 Pseudoódigo

Algorithm 1 Algoritmo de Nelmer Mead

Require: Se propone un punto inicial $X_0 = [x_1, x_2, ..., x_n]$
  Formar el simplejo inicial con los demas puntos
  $X_1 = [x_1 + \delta_1, x_2, ..., x_n]$
  $X_2 = [x_1 + \delta_1, x_2, ..., x_n]$
  $X_n = [x_1 + \delta_1, x_2, ..., x_n]$
  Evaluar la función en los puntos n+1
  while Se cumpla criterio de paro **do**
    $\overline{X} = \frac{1}{n}\sum_{i=0}^{n-1} X_i$ es decir se obtiene el centroide
    $X^R = \overline{X} + \alpha(\overline{X} - X_n)$
    if $f(X^R) > f(X_0)$ **then**
      $X^E = X^R + \gamma(X^R - \overline{X})$
      if $f(X^E) > f(X_0)$ **then**
        $X_n = X^E$
      else if $f(X^E) < f(X_0)$ **then**
        $X_n = X^R$
    else if $f(X^R) > f(X_{n-1})$ **then**
      $X_n = X^R$
    else
      if $f(X^R) > f(X_n)$ **then**
        $\widehat{X} = X^R$
      else
        $\widehat{X} = X_n$
      $X^C = \overline{X} + \beta(\widehat{X} - \overline{X})$
      if $f(X^C) > f(X_n)$ **then**
        $X_n = X^C$
      else
        $Simplejo = [X_0, \tau X_0 + (1-\tau)X_1, \tau X_0 + (1-\tau)X_2, ..., \tau X_0 + (1-\tau)X_{n-1}, \tau X_0 + (1-\tau)\widehat{X}]$

3 Resultados

Se realizaron 3 series de 30 ejecuciones, donde en cada serie se promedió el número de iteraciones, evaluaciones y tiempos, los resultados se presentan en la tabla, en la primer columna se presenta el número de ejecuciones en la configuración de la correspondiente fila, en la segunda columna se muestra el promedio del número total de iteraciones, en la tercer columna se presenta el promedio de las 30 evaluaciones, en la quinta y sexta columna indican los límites con que se proporcionó el punto inicial en el algoritmo, es decir $X_0 = U(XMin, XMax)$. Se puede observar que conforme el rango de X_0 sea mayor se necesita un número mayor de iteraciones para que el algoritmo obtenga el valor óptimo.

Tamaño Muestra	Iteraciones	Evaluaciones	Tiempos	X Min	Xmax
30	94	-8.34E-07	0.2761862333	-3	3
30	114	-2.42E-06	0.3015899667	-5	5
30	168	-9.78E-18	0.4143950333	-10	10

Fig. 2. Resultado de promediar tres muestras en la función objetivo de Rosenbrock en dos dimensiones.

4 Conclusions

Se presentó el algoritmo de Nelder-Mead el cual es útil en diversas situaciones, por ejemplo en el caso en que no se pueda obtener el gradiente de la función objetivo, también se sabe por experiencia que no es el mejor algoritmo en situaciones donde el número de parámetros a optimizar sean mayor a diez dimensiones.

References

[1] Zapatero Moreno, Ma José; Alegre Martínez, Jesús y Pacheco Bonrostro, Joaquín. *Análisis de algunas metaheurísticas creadas a partir de "Optimización Gravitatoria"*.
[2] Kyle Klein · Julian Neira *Nelder-Mead Simplex Optimization Routine for Large-Scale Problems: A Distributed Memory Implementation*