Algoritmo de búsquedas

Juan F. Olivares Pacheco*

Índice

1 Búsqueda por fuerza bruta

1.1 Fundamentos teóricos

El algoritmo de búsqueda por fuerza bruta es una técnica de optimización que examina sistemáticamente todas las soluciones posibles dentro de un espacio de búsqueda definido. A través de una evaluación exhaustiva, selecciona la solución que optimiza una función objetivo preestablecida.

*jfolivar@uda.cl

A diferencia de métodos más avanzados que explotan la estructura del problema, la fuerza bruta no hace suposiciones sobre la topología del espacio de soluciones. Esta característica garantiza la localización del óptimo global, siempre que el espacio de búsqueda sea finito y computacionalmente tratable. Su principal limitación es la complejidad computacional, que crece exponencialmente con la dimensionalidad del problema, restringiendo su uso a problemas de escala reducida.

1.2 Conceptos fundamentales

- **Búsqueda exhaustiva:** Implica la evaluación de cada punto en el espacio de soluciones sin descartar ninguna región prematuramente. Esto asegura la identificación del óptimo global.
- **Espacio de Búsqueda:** Es el conjunto de todas las soluciones candidatas. Puede ser de naturaleza discreta o continua. Para la implementación computacional, los espacios continuos deben ser discretizados.
- **Criterio de evaluación (función objetivo):** Es la función $f(\cdot)$ que se desea minimizar o maximizar. Permite cuantificar la calidad de cada solución candidata.
- **Complejidad computacional:** Para un problema con n variables, donde cada una puede tomar k valores, la complejidad es de orden $O(k^n)$. Este crecimiento exponencial es conocido como la "maldición de la dimensionalidad".

1.3 Implementación para problemas con dominio discreto

Consideremos el problema de minimización de la función:

$$f(x) = (-x + 10)^2 - 20,\ x \in \{0, 1, 2, \ldots, 20\}$$

A continuación se muestra la implementación en `R`.

```r
# Definir la función objetivo
f <- function(x) {
    (-x + 10)^2 - 20
}

# Definir el espacio de búsqueda discreto
x <- seq(from = 0, to = 20, by = 1)
```

```r
# Evaluar la función en todo el dominio (búsqueda exhaustiva)
y <- f(x)

# Identificar el índice del valor mínimo
indice <- which.min(y)

# Obtener la solución óptima
x_min <- x[indice]
y_min <- y[indice]

# Mostrar los resultados
cat("El valor minimo de f(x) es:", y_min,
    "\nEl valor optimo de x es:", x_min)
```

```
## El valor minimo de f(x) es: -20
## El valor optimo de x es: 10
```

Este ejemplo demuestra la simplicidad y efectividad del método en problemas de baja dimensionalidad.

1.4 Adaptación para dominios continuos

Para dominios continuos, es necesario discretizar el espacio de búsqueda. Este proceso introduce un error de aproximación controlado por la granularidad de la discretización (el tamaño del paso).

Consideremos el problema de maximización:

$$f(x) = 20x(1-x)^3,\ x \in (0,1)$$

A continuación se muestra la implementación en R.

```r
# Definir la función objetivo
f <- function(x) {
    20 * x * (1 - x)^3
}

# Discretizar el dominio continuo
# Un paso más pequeño aumenta la precisión y el costo computacional
step <- 0.01
x <- seq(from = 0, to = 1, by = step)
```

```r
# Evaluar la función en el dominio discretizado
y <- f(x)

# Identificar el índice del valor máximo
indice <- which.max(y)

# Obtener la solución óptima aproximada
x_max <- x[indice]
y_max <- y[indice]

# Mostrar los resultados
cat("El valor maximo de f(x) es:", y_max,
    "\nEl valor optimo de x es:", x_max)
```

```
## El valor maximo de f(x) es: 2.109375
## El valor optimo de x es: 0.25
```

El parámetro `step` define el balance entre la precisión de la solución y el costo computacional.

1.5 Extensión a problemas multivariados

La generalización a múltiples variables requiere la construcción de una grilla multidimensional que represente el espacio de búsqueda.

Consideremos el problema de minimización:

$$f(x, y) = x^2 + 2y^2 + 10,\ x,\ y \in (-10, 10)$$

A continuación se muestra la implementación en `R`.

```r
# Definir la función objetivo para que acepte un vector como entrada
f <- function(v) {
    x <- v[1]
    y <- v[2]
    x^2 + 2 * y^2 + 10
}

# Definir y discretizar el dominio para cada variable
step <- 0.1
```

```r
x <- seq(from = -10, to = 10, by = step)
y <- seq(from = -10, to = 10, by = step)

# Construir la grilla del espacio de búsqueda con todas las combinaciones
search_grid <- expand.grid(x, y)

# Evaluar la función en toda la grilla
# Usamos apply para aplicar la función a cada fila (combinación x, y)
# de la grilla
z <- apply(search_grid, 1, f)

# Identificar el índice del mínimo
indice <- which.min(z)

# Obtener la solución óptima
punto_optimo <- search_grid[indice, ]
z_min <- z[indice]

# Mostrar resultados
cat("El valor minimo de f(x, y) es:", z_min,
    "\nEl valor optimo de (x, y) es:", paste(punto_optimo, collapse = ", "))
```

```
## El valor minimo de f(x, y) es: 10
## El valor optimo de (x, y) es: 0, 0
```

1.6 La maldición de la dimensionalidad

El principal obstáculo de la fuerza bruta es el crecimiento exponencial del espacio de búsqueda con el número de variables. Consideremos la minimización de una función de 10 variables:

$$f(x_1, x_2, \ldots, x_{10}) = \sum_{i=1}^{10} (x_i - i)^2, \; x_i \in [-10, 10]$$

El objetivo es encontrar el conjunto de valores $(x_1, x_2, \ldots, x_{10})$ que minimizan la función f. Si discretizamos cada variable con un paso de 0.1 en el intervalo $[-10, 10]$, obtenemos:

$$\text{Puntos por variable} = \frac{(10 - (-10))}{0.1} + 1 = 201$$

Luego, el número total de combinaciones es:

$$\text{Total de combinaciones} = 201^{10} \approx 2.57 \times 10^{23}$$

Ahora supongamos, de forma optimista, que nuestro computador puede evaluar un millón (10^6) de combinaciones por segundo. Entonces, el tiempo total para evaluar todas las combinaciones es:

$$\frac{2.57 \times 10^{23} \text{ combinaciones}}{10^6 \text{ combinaciones / seg}} = 2.57 \times 10^{17} \text{ segundos}$$

Si este resultado lo convertimos en años:

$$\frac{2.57 \times 10^{17} \text{ segundos}}{3.154 \times 10^7 \text{ segundos / año}} \approx 8.15 \times 10^9 \text{ años}$$

Por lo tanto, tomaría aproximadamente 8150 millones de años completar la evaluación.

Esta limitación fundamental hace que el algoritmo de fuerza bruta sea viable únicamente para problemas de baja dimensionalidad o con espacios de búsqueda muy restringidos. Para problemas de mayor escala, es necesario recurrir a algoritmos más sofisticados que exploten la estructura del problema o utilicen estrategias heurísticas para reducir el espacio de búsqueda efectivo.

2 Búsquedas aleatorias

2.1 Principios fundamentales

El algoritmo de búsquedas aleatorias representa una alternativa estocástica al enfoque determinista de la fuerza bruta. En lugar de evaluar sistemáticamente todas las soluciones posibles, este método genera un conjunto de soluciones candidatas mediante procesos aleatorios, evaluándolas para identificar la mejor encontrada durante el proceso de búsqueda.

Esta estrategia sacrifica la garantía de encontrar el óptimo global a cambio de una reducción significativa en el costo computacional. La efectividad del método depende tanto del número de muestras generadas como de la distribución de probabilidad utilizada para el muestreo del espacio de búsqueda.

2.2 Implementación básica

Ilustremos la implementación del algoritmo mediante el problema de minimización:

$$f(x) = x^2 - 4x,\ x \in [0, 4]$$

La función objetivo se define como:

```
f <- function(x) {
  x^2 - 4 * x
}
```

La generación aleatoria de soluciones candidatas se realiza mediante:

```
set.seed(0)
n <- 10
x <- runif(n, min = 0, max = 4)
```

La función `runif()` genera n valores distribuidos uniformemente en el intervalo especificado. El uso de `set.seed()` garantiza la reproducibilidad de los resultados.

El proceso de evaluación y selección del mejor candidato sigue la misma estructura que en el algoritmo de fuerza bruta:

```
y <- f(x)
indice <- which.min(y)
x_min <- x[indice]
y_min <- y[indice]
```

Mostramos los resultados.

```
cat("El valor minimo de f(x) es:", y_min,
  "\nEl valor optimo de x es:", x_min)
```

```
## El valor minimo de f(x) es: -3.915078
## El valor optimo de x es: 2.291413
```

2.3 Consideraciones sobre la convergencia

La convergencia del algoritmo de búsquedas aleatorias hacia el óptimo global es probabilística y depende del número de muestras generadas. Según el teorema del límite, cuando el número de muestras tiende a infinito, la probabilidad de encontrar una solución arbitrariamente cercana al óptimo global tiende a uno, siempre que la distribución de muestreo tenga soporte en todo el espacio de búsqueda.

En la práctica, el número de muestras requerido para alcanzar una aproximación satisfactoria del óptimo puede ser considerablemente menor que el número total de puntos en una discretización fina del espacio, especialmente en problemas de alta dimensionalidad donde la búsqueda exhaustiva es computacionalmente prohibitiva.

3 Ejercicios

3.1 Búsqueda por fuerza bruta

Implemente el algoritmo de búsqueda por fuerza bruta para resolver los siguiente problemas de optimización. Para dominios continuos, experimente con diferentes valores de discretización y compare los resultados obtenidos con las soluciones analíticas cuando sea posible.

1. Encuentre el mínimo de la función:

$$f(x) = x^2,\ -10 \leq x \leq 10$$

2. Encuentre el máximo de la función:

$$f(x) = \sin(x),\ 0 \leq x \leq 2\pi$$

3. Encuentre el mínimo de la función:

$$f(x, y) = x^2 + y^2,\ -5 \leq x, y \leq 5$$

4. Encuentre el máximo de la función:

$$f(x, y) = \sin(x) \cdot \cos(y),\ -2\pi \leq x, y \leq 2\pi$$

5. Encuentre el mínimo de la función

$$f(x, y) = e^{x+y} - x^2 - y^2,\ -3 \leq x, y \leq 3$$

3.2 Búsqueda aleatoria

Implemente el algoritmo de búsqueda aleatoria para aproximar las soluciones de los siguientes problemas. Experimente con diferentes tamaños de muestra y analice la convergencia de las soluciones.

1. Encuentre una aproximación al mínimo de la función:

$$f(x) = x^2,\ -10 \leq x \leq 10$$

2. Encuentre una aproximación al máximo de la función:

$$f(x) = \sin(x),\ 0 \leq x \leq 2\pi$$

3. Encuentre una aproximación al mínimo de la función:

$$f(x, y) = x^2 + y^2,\ -5 \leq x, y \leq 5$$

4. Encuentre una aproximación al máximo de la función:

$$f(x, y) = \sin(x) \cdot \cos(y),\ -2\pi \leq x, y \leq 2\pi$$

5. Encuentre una aproximación al mínimo de la función:

$$f(x, y, z) = x^2 + y^2 + z^2 - xyz,\ -3 \leq x, y, z \leq 3$$