Búsqueda Guiada por Regresión: un algoritmo determinista para la optimización global

Miquel Oliver-Vert

Universidad Nacional de Educación a Distancia

19 de octubre de 2025

Resumen

Se presenta el algoritmo Búsqueda Guiada por Regresión (BGR), un método determinista de optimización global diseñado para localizar el mínimo de funciones tanto reales como vectoriales definidas en dominios multidimensionales. El algoritmo integra modelos de regresión local en el proceso de exploración, aprovechando la información obtenida en evaluaciones previas para orientar la búsqueda hacia las regiones más prometedoras del dominio. Este enfoque permite reducir significativamente el número de evaluaciones necesarias, manteniendo una exploración sistemática del dominio. El artículo describe formalmente el procedimiento e ilustra su funcionamiento mediante un ejemplo.

1. Introducción

La optimización global de funciones definidas sobre dominios multidimensionales es un problema fundamental en matemáticas aplicadas y en numerosas áreas de la ingeniería, la economía y la inteligencia artificial. En muchos casos, la función objetivo es no convexa, multimodal y costosa de evaluar, lo que hace necesario desarrollar algoritmos que logren un equilibrio entre exploración efectiva del dominio y eficiencia computacional.

Los métodos deterministas clásicos con garantías teóricas, como los algoritmos Lipschitzianos (Galperin, 1985; Hansen & Jaumard, 1995; Piyavskii, 1972; Sergeyev, 1995; Shubert, 1972), ofrecen una exploración sistemática del espacio de búsqueda, pero su aplicación práctica suele verse limitada por la dificultad de conocer una constante de Lipschitz eficaz. Otras aproximaciones, como el método DIRECT (Jones et al., 1993), eliminan este requisito, aunque tienden a requerir un número elevado de evaluaciones para alcanzar soluciones precisas en dimensiones moderadas o altas. Por último, los métodos heurísticos —entre ellos, los algoritmos genéticos (Bremermann, 1958; Goldberg, 1989; Holland, 1975), el *Particle Swarm Optimization* (Kennedy & Eberhart, 1995) o el Simulated Annealing (Kirkpatrick et al., 1983)— emplean reglas diseñadas para obtener un buen rendimiento en la práctica, pero sin garantías formales de convergencia al mínimo global.

En el presente artículo, se propone un nuevo algoritmo, denominado Búsqueda Guiada por Regresión (BGR), un método determinista de optimización global que incorpora modelos de regresión local para dirigir la búsqueda hacia las regiones más prometedoras del dominio. En cada iteración, el algoritmo selecciona un nivel de paso y un punto asociado, estima mediante regresión las imágenes en distintas direcciones y evalúa el punto más prometedor, que se incorpora a la lista de puntos explorados. Este enfoque permite reducir el número de evaluaciones necesarias, manteniendo una exploración estructurada del espacio de búsqueda.

El resto del artículo se organiza como sigue. En la Sección 2 se describe formalmente el algoritmo BGR y el procedimiento de estimación por regresión. La Sección 3 ilustra su funcionamiento mediante un ejemplo, y la Sección 4 presenta las conclusiones.

2. Descripción del algoritmo

Empiezo definiendo la notación y objetos usados durante la ejecución de este método, aplicado a una función $f : \mathbb{R}^m \longrightarrow \mathbb{R}^n$:

- Sean $\ell, u \in \mathbb{R}^m$ las cotas inferiores y superiores que definen el hipercubo $D := \prod_{i=1}^{m} [\ell_i, u_i] \subset \mathbb{R}^m$, que es nuestra región de búsqueda, y sea $d = u - \ell$.

- X es una matriz $N \times m$ que almacena los N puntos evaluados.

- Y es una matriz $N \times n$ que almacena las imágenes de los N puntos evaluados.

- P es una matriz $N \times 1$ que almacena los *niveles de paso* de los N puntos evaluados. El nivel de paso dado a un punto es un entero que determina qué tan cerca vamos a explorar alrededor de dicho punto, en caso que este sea escogido. Es decir, si hemos decidido explorar un nuevo punto por encima del punto x en la dirección i, y el nivel de paso es p, entonces el nuevo punto z a evaluar sería $z \in \mathbb{R}^m$ con $z_{i'} = x_{i'}$ para $\forall i' \neq i$ y $z_i = x_i + 2^{-p} d_i$, donde d es un vector predefinido de longitudes de paso. Nótese que, con esta definición, un nivel de paso más pequeño implica dar un salto más grande.

- S es una matriz $N \times 1$ que almacena las *puntuaciones* de los N puntos evaluados. La puntuación dada a un punto $x \in \mathbb{R}^m$ viene determinada por una función $s : \mathbb{R}^n \longrightarrow \mathbb{R}$ que, dados los valores de $f(x)$, no da un escalar. Esta definición nos permite abarcar distintos tipos de problemas:

  - Minimización de funciones reales $f : \mathbb{R}^m \longrightarrow \mathbb{R}$: basta tomar $Z = Y$; es decir, $s = f$.

  - Minimización de funciones vectoriales $f : \mathbb{R}^m \longrightarrow \mathbb{R}^n$: para ello, necesitamos definir un vector no negativo $w \in \mathbb{R}^n_+$ con los pesos dados a cada una de las dimensiones del espacio imagen, y definimos $s(x) := w \cdot f(x)$.

- Resolver el sistema $f(x) = c$ (o, si no existe solución, buscar los puntos que más se aproximen a satisfacer las n condiciones): para ello, como en el anterior caso, definimos un vector no negativo de pesos $w \in \mathbb{R}^n$, y definimos $s(x) := \sum_{i=1}^{n} w_i |f_i(x) - c_i|$.

Con esto, podemos pasar a describir el algoritmo:

- Inicializamos el método eligiendo un punto inicial $x_0 \in D$ arbitrario y evaluando este punto, así como los m puntos $\{x_i\}_{i=1}^m$ definidos por

$$x_{i,j} = x_{0,j} \text{ para } \forall j \neq i \text{ y } x_{i,i} = \begin{cases} x_{0,i} + 2^{-1} d_i & \text{, si } x_{0,i} + 2^{-1} d_i \leq u_i, \\ x_{0,i} - 2^{-1} d_i & \text{, en caso contrario.} \end{cases}$$

  e inicializamos los objetos descritos arriba: $X = \Big(x_0, \ldots, x_m\Big)$, $Y = \Big(f(x_0), \ldots, f(x_m)\Big)$, $P = \Big(1, \ldots, 1\Big)$, $Z = \Big(s(f(x_0)), \ldots, s(f(x_m))\Big)$.

- En cada iteración, para cada $\bar{p} = 1, \ldots, 10$ (donde se fija arbitrariamente 10 como el nivel máximo de pasos permitido), y para cada nivel de paso $p = 1, \ldots, \bar{p}$, se realizan los siguientes pasos. Nótese que estos bucles, que determinan los niveles de paso, priorizan niveles de paso bajos, lo cual implica, como se desprende del paso 4, que se prioriza la exploración de regiones poco exploradas.[1]

  1. Sea $\mathcal{B} := \Big\{k \in \{1, \ldots, N\} : P(k) \leq p\Big\}$ el conjunto de puntos con nivel de paso no superior a p. Si $\mathcal{B}$ es vacío, saltamos al siguiente p; si no:

  2. De los puntos asociados a conjunto $\mathcal{B}$, escogemos el que tiene menor puntuación; es decir, elegimos $k' := \arg\min_{k \in \mathcal{B}} S(k)$. Sea $x' = X(:, k')$ tal punto y $p' = P(k')$ su nivel de paso.

[1] Digo "prioriza" en el sentido de que, para $p < p'$, p' se escoge para $\bar{p} \in P' := \{p', \ldots, 10\}$, mientras que p se escoge para $P := \{p, \ldots, p' - 1\} \cup P'$.

3. Aumentamos su nivel de paso $P(k') = p' + 1$. Esto implica que (i) este punto va a ser elegible en menos iteraciones (dada la definición de los bucles de p y $\bar{p}$); y (ii) de ser elegido, la exploración se realizará en una región más cercana a dicho punto.

4. Definimos $2m$ puntos candidatos a evaluar: $\{z_{(i,a)}\}_{i\in\{1,\dots,m\}a\in\{-1,1\}}$, definidos como $z_{(i,a),j} = x_j$ para $\forall j \neq i$ y $z_{(i,a),i} = x_i + a2^{-p'}d_i$, de los cuales eligiremos a lo sumo uno. Para ser elegible, el punto $z = z_{(i,a)}$, además de hallarse dentro del dominio, debe cumplir que no haya puntos *cercanos* a z en X; es decir, que el siguiente conjunto sea vacío:

$$\mathcal{L} := \Big\{x \in X : |z_i - x_i| \leq 2^{-|p|}d_i,\ \forall i \in \{1,\dots,m\}\Big\}.$$

   Esto implica que si ya hay algún punto en el hipercubo de centro z asociado al nivel de paso p, entonces el punto z solo podrá ser elegido para niveles de paso mayores a p, lo cual hace su elección más difícil.

5. Si en el paso previo no se ha obtenido ningún candidato elegible, volvemos al paso 1. De lo contrario, de entre los candidatos elegibles, calculamos una estimación $\hat{f}(z_{(i,a)})$ de $f(z_{(i,a)})$ (una posibilidad es usar el método de regresión local propuesto en la Sección 2.0.1), calculamos la puntuación asociada a dicha estimación $s(\hat{f}(z_{(i,a)}))$ y elegimos el punto mínimo.

6. Evaluamos el punto z elegido en el paso previo; añadimos z a X, $f(z)$ a Y, $s(f(z))$ a S, y $p'' = \text{máx}(1, p-1)$ a P. Nótese que la reducción en el nivel de paso de los nuevos puntos evaluados tiene como objetivo: (i) aumentar la probabilidad de que dichos puntos sean seleccionados, y (ii) acelerar la exploración hacia regiones nuevas *prometedoras*.

2.0.1. Regresión

Para la estimación por regresión de la imagen $f(z)$ de un punto z, siendo p el nivel de paso de la iteración, seguimos el siguiente procedimiento. Inicializamos $p' = p$ y hacemos:

1. Consideramos el subconjunto $X' \subset X$ definido seguidamente, y denotamos Y' el conjunto de imágenes de X'.

$$X' := \{x \in X |\ |x_i - z_i| \leq d_i 2^{-p'}\}.$$

2. Sea N el número de puntos de X' y $\bar{g} \in \mathbb{N}$.[2] Si $N > N(m, \bar{g}) = \frac{(m+\bar{g})!}{m!\bar{g}!}$, reducimos $p' = p' - 1$ y volvemos al paso 1; de lo contrario, vamos al siguiente paso.[3] Nótese que al reducir p' estaremos haciendo la regresión aún más local. Inicializamos el grado del polinomio $g = \bar{g}$.

3. Definimos el conjunto de multiindices $\mathcal{A} := \left\{\vec{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{N}^M : \sum_{i=1}^m \alpha_i \leq g\right\}$, al cual le damos un orden lexicográfico $\mathcal{A} = \{\vec{\alpha}_1, \ldots, \vec{\alpha}_{N(m,g)}\}$.[4]

4. Definimos $\hat{X}$ como una matriz $N \times N(m, g)$, definiendo el elemento de la fila i (asociada al punto $x_i \in X'$) y columna j (asociada al multiindice $\vec{\alpha}_j \in \mathcal{A}$) como $\hat{X}_{i,j} = \prod_{k=1}^{m} x_{i,k}^{\alpha_{j,k}}$.

5. Para dar más peso a los puntos de X' cercanos al punto z, definimos los pesos $W = (w_1, \ldots, w_N)I_N$, donde $w_i = e^{-||x_i - z||_1 \lambda}$, donde $\lambda \in [0, \infty)$. Nótese que para $\lambda = 0$ todos los puntos tendrán el mismo peso, mientras que cuanto mayor sea λ mayor peso se dará a los puntos próximos a z. Este enfoque sigue la idea del aprendizaje localmente ponderado descrito en Atkeson et al. (1997).

6. Se obtienen los coeficientes del polinomio que minimiza la suma de residuos cuadrados: $\beta = \arg\min_{\hat{\beta}} \sum_{i=1}^{N} w_i (Y_i - \sum_{j=1}^{N(m,g)} \hat{X}_{i,j}\beta_j)^2$. En forma matricial, tenemos que el óptimo cumple $(W\hat{X})^t WY = (W\hat{X})^t W\hat{X}\beta$, por lo que si $(W\hat{X})^t W\hat{X}$ es invertible, obtenemos el vector de coeficientes β. En caso que $(W\hat{X})^t W\hat{X}$ sea singular, entonces reducimos el grado del polinomio, $g = g - 1$, y volvemos al paso 3.

[2] Por ejemplo, $\bar{g} = 4$.

[3] Nótese que $N(m, \bar{g})$ es el número de coeficienes a estimar para un polinomio de grado $\bar{g}$.

[4] En particular, dados dos $\alpha_{j_1}, \alpha_{j_2} \in \mathcal{A}$, será $j_1 < j_2$ si y solo si $\alpha_{j_1,i^*} < \alpha_{j_2,i^*}$ para $i^* := \text{máx}\{i \in \{1, \ldots, m\} : \alpha_{j_1,i} \neq \alpha_{j_2,i}\}$.

7. La estimación de $f(z)$ es $\sum_{j=1}^{N(m,g)} z^{\vec{\alpha}_j}\beta_j$.

3. Ilustración del algoritmo

Para ilustrar el funcionamiento del algoritmo, la Figura 1 muestra las seis primeras iteraciones del mismo aplicado a la función *peaks* en MATLAB. La iteración 0 corresponde al paso de inicialización, en el que se evalúa el punto inicial (recuérdese que puede elegirse cualquier punto del dominio) junto con un punto adicional en cada dirección. Las cinco iteraciones siguientes corresponden a los valores $\hat{p} = 1, \ldots, 5$.

En la iteración 1 se selecciona el punto $(0, -3)$, que pasa a nivel de paso 2 (color rojo), y se evalúa el punto $(3, -3)$, el cual adquiere el nivel de paso mínimo 1 (azul).

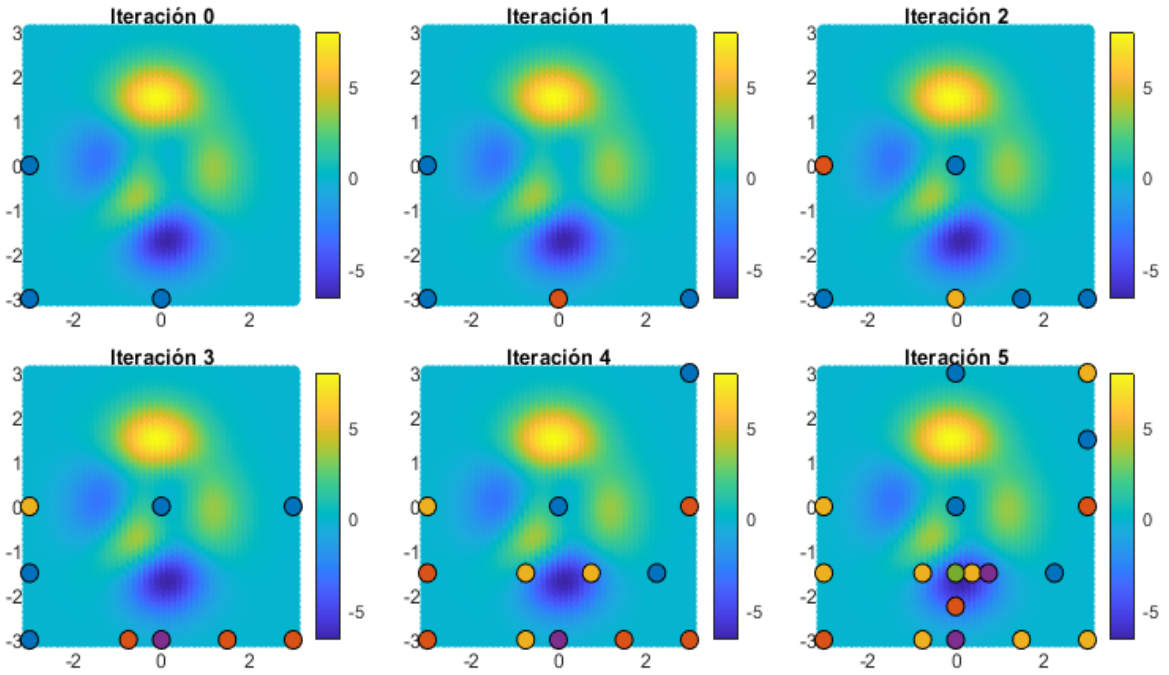

Figura 1: La figura muestra la función *peaks* de MATLAB en el dominio $[-3, 3]^2$, donde la imagen de la función en cada punto se representa mediante la escala cromática indicada. Además, se muestran superpuestos los puntos evaluados por el algoritmo hasta el momento, con distintos colores que indican el nivel de paso de cada punto al finalizar la iteración: $p = 1$ (azul), $p = 2$ (rojo), $p = 3$ (amarillo), $p = 4$ (violeta) y $p = 5$ (verde).

En la iteración 2 ($\hat{p} = 2$), para $p = 1$ se elige el punto $(-3, 0)$, que pasa a nivel de paso

2 (rojo), y se evalúa el punto $(0, 0)$, con nivel de paso mínimo 1 (azul); mientras que para $p = 2$ se vuelve a seleccionar el punto $(-3, 0)$, que asciende a nivel de paso 3 (amarillo), y se evalúa el punto $(-3, 1{,}5)$, que adquiere el nivel de paso $p' = p - 1 = 1$ (azul).

Finalmente, en la iteración 3, para $p = 1$, obsérvese que se elige el punto $(-3, 1{,}5)$, que pasa a nivel de paso 2 (rojo), pero ninguno de sus candidatos resulta elegible, ya sea por encontrarse fuera del dominio o por estar demasiado próximos (a nivel $p = 1$) a puntos previamente evaluados. Dado que no se evalúa ningún nuevo punto, el algoritmo selecciona otro punto con $p = 1$, que en este caso es $(3, -3)$.

4. Conclusiones

En este artículo se ha presentado un nuevo algoritmo, un método determinista de optimización global que integra técnicas de regresión local en el proceso de exploración. Su diseño permite aprovechar la información obtenida en evaluaciones previas para orientar la búsqueda hacia las regiones más prometedoras del dominio, reduciendo el número de evaluaciones necesarias.

En conjunto, el BGR constituye una propuesta especialmente útil en contextos donde el coste de evaluación de la función objetivo es elevado. En este sentido, resulta interesante considerar una versión del algoritmo estructurada en dos fases iterativas: en la primera, se ejecutarían algunas iteraciones del BGR sobre la función original; en la segunda, el algoritmo operaría sobre una aproximación de dicha función obtenida por regresión a partir de los puntos evaluados en la fase inicial. Finalmente, también cabe mencionar que, en el caso de resolución de sistemas, puede resultar más efectivo definir un bucle adicional para cada una de las ecuaciones, de manera que en cada iteración se elija el candidato cuya estimación se acerque más al cumplimiento de dicha condición.

Referencias

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally Weighted Learning. *Artificial Intelligence Review*, *11*(1), 11-73. https://doi.org/10.1023/A:1006559212014

Bremermann, H. J. (1958). The evolution of intelligence: the nervous system as a model of its environment [Also reprinted in: Self-Organizing Systems (1962), pp. 93–108]. *Technical Report No. 1, Department of Mathematics, University of Washington.*

Galperin, E. A. (1985). The Cubic Algorithm. *Journal of Mathematical Analysis and Applications*, *112*(2), 635-640.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Hansen, P., & Jaumard, B. (1995). Lipschitz optimization. *Handbook of Global Optimization*, 407-493.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Jones, D. R., Perttunen, C. D., & Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, *79*(1), 157-181.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942-1948.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680.

Piyavskii, S. A. (1972). An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, *12*(4), 57-67.

Sergeyev, Y. D. (1995). An information global optimization algorithm with local tuning. *SIAM Journal on Optimization*, *5*(4), 858-870.

Shubert, B. O. (1972). A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, *9*(3), 379-388.