

Introducción a la Optimización

Módulo 4 : Técnicas computacionales avanzadas para modelar fenómenos sociales
Concentración en Economía Aplicada y Ciencia de Datos
ITESM

13 de marzo de 2024



Todxs optimizamos



Introducción a la Optimización

- La optimización es una herramienta importante en las ciencias de la decisión y en el análisis de sistemas físicos.
- Primero debemos identificar un **objetivo**, esto es una medida cuantitativa del desempeño del sistema bajo estudio.
- El objetivo depende de ciertas características del sistema, llamadas **variables**.
- Con frecuencia, las variables están **restringidas** (*constrained*) en alguna manera.

Introducción a la Optimización

- El proceso de identificar el objetivo, las variables y las restricciones para un problema dado, se le conoce como **modelación**.
- Una vez que se ha formulado el modelo, podemos utilizar un *algoritmo optimización* para encontrar su solución.
- **No hay** un algoritmo de optimización universal: hay una colección de algoritmos, cada uno de los cuales se adapta a un tipo particular de problema de optimización.

Introducción a la Optimización

- Despues de aplicar un algoritmo de optimización al modelo, debemos ser capaces de reconocer si ha sido exitoso en su tarea de encontrar una solución.
- En algunos casos hay expresiones matemáticas conocidas como *condiciones de optimalidad* para cerciorarse que el conjunto actual de variables es la solución del problema.
- El modelo puede ser mejorado al aplicar técnicas como el *análisis de sensibilidad* que revela la sensibilidad de la solución a los cambios en el modelo y los datos.

Formulación matemática

En términos formales, la optimización es la minimización o maximización de una función sujeta a restricciones en sus variables. En notación :

- x es el vector de variables, también conocido como *parámetros*.
- f es la *función objetivo*, una función (escalar¹) de x que queremos maximizar o minimizar.
- c_i son funciones de restricción, que son funciones escalares de x que definen ciertas ecuaciones y desigualdades que el vector x debe satisfacer.

Usando esta notación, el problema de optimización puede ser escrito como:

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} f(x) & \text{sujeto a} \\ & c_i(x) = 0, \quad i \in \mathcal{E} \\ & c_i(x) \geq 0, \quad i \in \mathcal{I} \end{array} \quad (1)$$

donde \mathcal{E} y \mathcal{I} son conjuntos de índices para restricciones de igualdad y desigualdad.

¹Es decir, una función que va de \mathbb{R}^n a \mathbb{R}

Ejemplo

Considere el siguiente problema ,

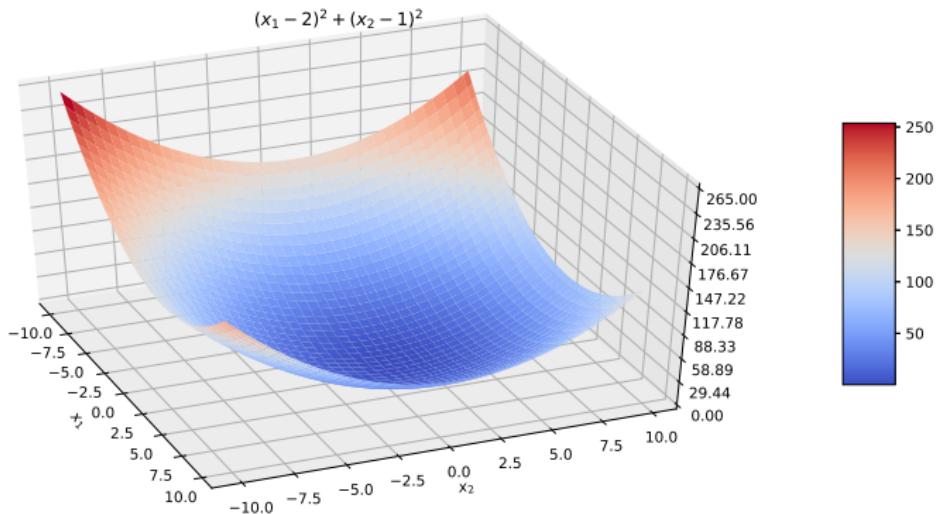
$$\min (x_1 - 2)^2 + (x_2 - 1)^2 \quad \text{sujeto a} \quad \begin{aligned} x_1^2 - x_2 &\geq 0, \\ x_1 + x_2 &\geq 2. \end{aligned} \quad (2)$$

Podemos escribir este problema en la forma (1) al definir

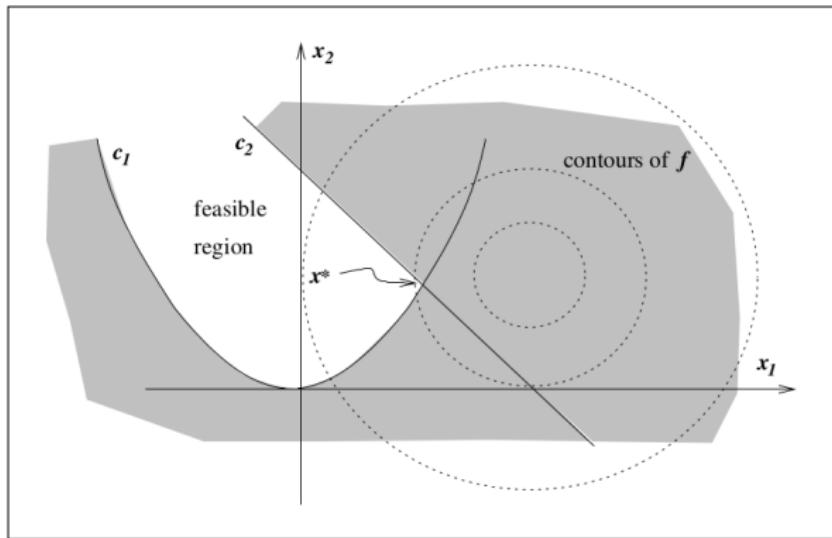
$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$c(x) = \begin{bmatrix} c_1(x) \\ c_2(x) \end{bmatrix} = \begin{bmatrix} -x_1^2 + x_2 \\ -x_1 - x_2 + 2 \end{bmatrix}, \quad \mathcal{I} = \{1, 2\}, \mathcal{E} = \emptyset$$

Conozcamos la función objetivo



Contorno de la función objetivo



- La *región factible* es el conjunto de puntos que satisfacen todas las restricciones.
- El punto x^* es la solución al problema.

Maximización

Un problema de maximización de una función $f(x)$ puede ser resrito de acuerdo al modelo (1) como :

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} -f(x) & \text{sujeto a} \\ & c_i(x) = 0, \quad i \in \mathcal{E} \\ & c_i(x) \geq 0, \quad i \in \mathcal{I} \end{array} \quad (3)$$

Optimización discreta y continua

- Los problemas de *optimización discreta* pueden contener variables enteras, binarias y objetos variables más abstractos, como permutaciones de un conjunto ordenado. Ejemplos:
 - ▶ TSP (Travelling salesman problem).
- La característica que define a los problemas de optimización discreta es que x pertenece a un conjunto finito y numerable.
- Por su parte, el conjunto factible para los problemas de *optimización continua* es infinito no numerable, como cuando los componentes de x pueden ser números reales.

Optimización restringida y no restringida

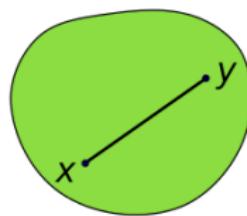
- En los problemas de *optimización no restringida* tenemos que en (1), $\mathcal{E} = \mathcal{I} = \emptyset$.
- Este tipo de problemas surgen también como problemas de optimización restringida, en los que las restricciones son reemplazadas por términos de penalización agregados a la función objetivo.
- Los problemas de *optimización restringida* surgen de modelos en los que las restricciones tienen un rol importante, como por ejemplo la restricción presupuestaria en el problema de consumidor.

Programación lineal y no lineal

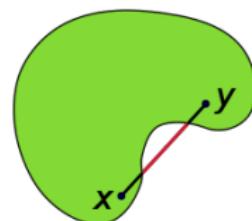
- Cuando la función objetivo y todas las restricciones son funciones lineales de x , el problema se le conoce como de *programación lineal*.
- Los problemas de *programación no lineal* son aquellos en los que al menos una restricción o la función objetivo son no lineales.

Convexidad

- El término *convexo* puede aplicarse tanto a conjuntos como a funciones.
- Un conjunto $S \in \mathbb{R}^n$ es un *conjunto convexo* si podemos trazar una línea que conecte a dos puntos en S y que se encuentre por completo en dentro de S .



(a) Convexo



(b) No Convexo

- Formalmente, para cualquier dos puntos $x \in S$ y $y \in S$, tenemos $\alpha x + (1 - \alpha)y \in S$ para todo $\alpha \in [0, 1]$

- La función f es una *función convexa* si su dominio S es un conjunto convexo y si para cualquier dos puntos x y y en S , se satisface la siguiente propiedad:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y), \text{ para todo } \alpha \in [0, 1] \quad (4)$$

- Decimos que f es *estRICTAMENTE CONVEXA* si las desigualdades en (4) es estricta si $x \neq y$ y α está en el intervalo abierto $(0, 1)$.
- Una función f se dice que es *concava* si $-f$ es convexa.

Algoritmos de Optimización

- Los algoritmos de optimización son iterativos.
- Estos inician con un valor inicial de la variable x y generan una secuencia de estimaciones mejoradas (llamadas *iteraciones*) hasta que terminen, **con suerte**, en una solución.
- La estrategia utilizada para pasar de una iteración a la siguiente es lo que distingue a los algoritmos entre sí.
- Algunos algoritmos acumulan información recopilada en iteraciones anteriores, mientras que otros usan solo información local obtenida en el punto actual.

Algoritmos de Optimización

Los algoritmos de optimización deben poseer las siguientes propiedades.

- **Robustez:** Deben tener un buen desempeño en una amplia variedad de problemas en su tipo, para todos los valores razonables del punto de inicio.
- **Eficiencia:** No deben requerir un excesivo uso de tiempo de cómputo o almacenamiento.
- **Precisión:** Deben ser capaces de encontrar una solución con precisión, sin ser demasiado sensible a los errores en los datos o a los errores de redondeo aritmético.

¿Qué es una solución?

Cuando minimizamos f , deseamos encontrar un **minimizador global**, un punto donde la función alcanza su valor mínimo. Una definición formal es la siguiente:

Minimizador global

Un punto x^* es un *minimizador global* si $f(x^*) \leq f(x)$ para todo x , donde x ranges over all \mathbb{R}^n .

El minizadador global puede ser difícil de encontrar porque nuestro conocimiento de f es usualmente sólo local.

En general, encontrar un punto (maximizador o minimizador) global es computacionalmente intratable([Neumaier, 2004](#)).

En tal caso, nos conformaremos con encontrar **óptimos locales**.

¿Qué es una solución?

La mayoría de los algoritmos son capaces de encontrar un **minimizador local**, el cual es un punto que alcanza el valor más pequeño de f en una vecindad. Formalmente:

Minimizador local (débil)

Un punto x^* es un *minimizador local* si hay una vecindad \mathcal{N} de x^* tal que $f(x^*) \leq f(x)$ para todo $x \in \mathcal{N}$.

(Una vecindad de x^* es un conjunto abierto que contiene a x^*)

Un punto que satisface esta definición se le denomina un **minimizador local débil**.

¿Qué es una solución?

Un **minimizador local estricto** es aquel que

Minimizador local (estricto)

Un punto x^* es un *minimizador local estricto* si hay una vecindad \mathcal{N} de x^* tal que $f(x^*) < f(x)$ para todo $x \in \mathcal{N}$ con $x \neq x^*$.

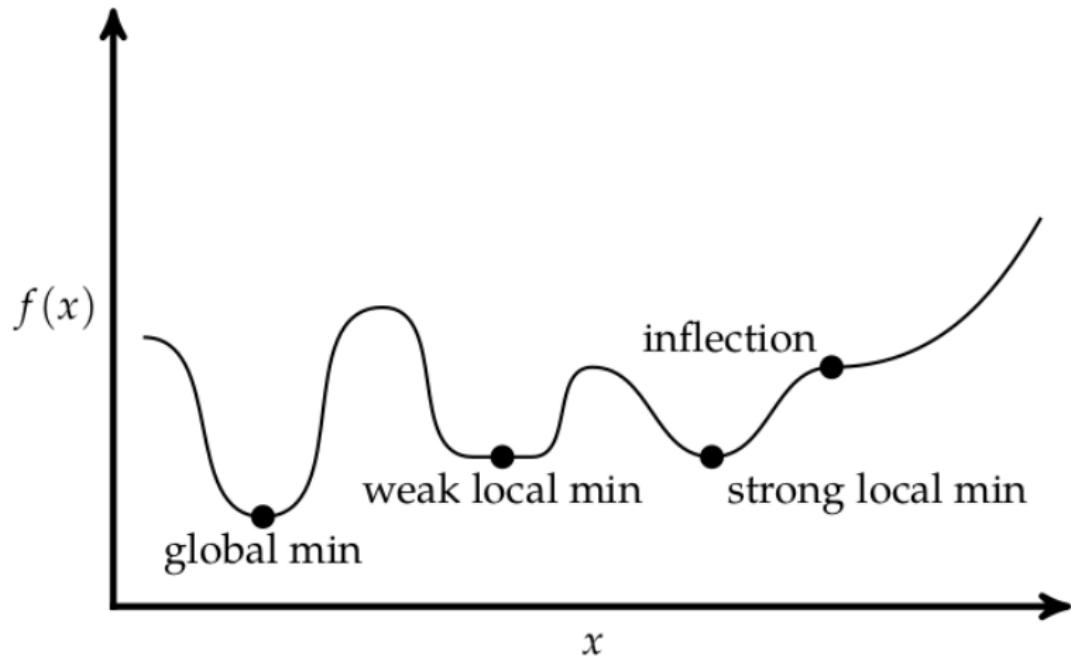


Figura: Tomado de Kochenderfer and Wheeler (2019)

Condiciones para mínimos locales (en funciones univariadas)

Asumimos que:

- La función objetivo es univariada y diferenciable.
- No hay restricciones.

Se garantiza que un punto es un mínimo local estricto si su primer derivada es igual a cero y su segunda derivada es positiva²

- $f'(x^*) = 0$
- $f''(x^*) > 0$

²Si $f'(x^*) = 0$ y $f''(x^*) < 0$ entonces x^* es un máximo local.

Condiciones **necesarias** de mínimos locales

Un punto es un mínimo local si su primera derivada es igual a cero y la segunda derivada es no negativa:

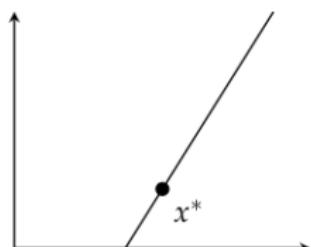
- $f'(x^*) = 0$, es la *condición necesaria de primer orden* (FONC)³.
- $f''(x^*) \geq 0$, es la *condición necesaria de segundo orden* (SONC)

Estas condiciones se denominan *necesarias* porque todos los mínimos locales las cumplen.

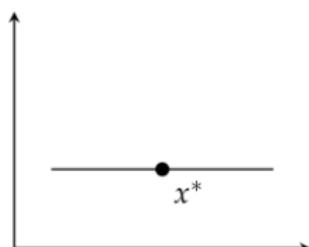
³Un punto que satisface la condición necesaria de primer orden se le denomina *punto estacionario*

Condiciones necesarias de mínimos locales

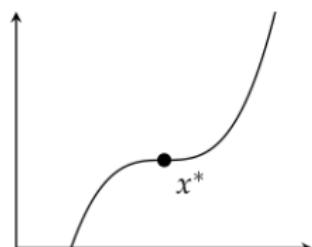
Desafortunadamente, no todos los puntos con primera derivada y segunda derivada igual a cero son mínimos locales, como se muestra a continuación:



SONC but not FONC



FONC and SONC



FONC and SONC

Derivadas en múltiples dimensiones: Gradiente

- El *gradiente* es la generalización de la derivada en funciones multivariadas.
- Captura la pendiente local de la función, permitiéndonos predecir el efecto de dar un pequeño paso desde un punto en cualquier dirección.
- El gradiente apunta en la dirección del ascenso más pronunciado del *hiperplano tangente*.
- El gradiente apunta en la dirección de máximo cambio en el valor de la función.

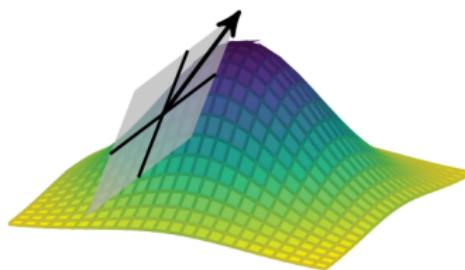


Figura: Tomado de Kochenderfer and Wheeler (2019)

Derivadas en múltiples dimensiones: Gradiente

El *gradiente* de f en \mathbf{x} es denotado como $\nabla f(\mathbf{x})$ y es un vector.

Cada entrada del vector es la *derivada parcial*⁴ de f con respecto a cada variable:

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right] \quad (5)$$

Cálculo de gradiente en un punto particular

Calcula el gradiente de $f(\mathbf{x}) = x_1x_2$ en $\mathbf{c} = [2, 0]$.

$$f(\mathbf{x}) = x_1x_2$$

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2} \right] = [x_2, x_1]$$

$$\nabla f(\mathbf{c}) = [0, 2]$$

⁴La derivada parcial de una función con respecto a una variable es la derivada asumiendo que el resto de las variables se mantienen constantes. Se denota como $\frac{\partial f}{\partial x}$ ↗ ↘ ↙

Derivadas en múltiples dimensiones: Hessiano

El *Hessiano* de una función multivariada es una matriz que contiene todas las segundas derivadas con respecto a la entrada.

La matriz Hessiana se interpreta como la segunda derivada de una función multivariada, de manera que captura información acerca de la curvatura local de la función.

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ & \vdots & & \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_n} \end{bmatrix} \quad (6)$$

Condiciones para mínimos locales (en funciones multivariadas)

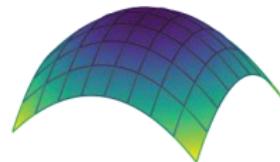
Las siguientes condiciones son necesarias para que \mathbf{x} sea un mínimo local de f :

- $\nabla f(\mathbf{x}) = \mathbf{0}$, es la *condición necesaria de primer orden* (FONC).
- $\nabla^2 f(\mathbf{x})$ sea positiva semi definida, es la *condición necesaria de segundo orden* (SONC).

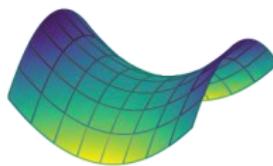
FONC y SONC son generalizaciones del caso univariado. FONC no dice que la función no está cambiando en \mathbf{x} .

Condiciones para mínimos locales (en funciones multivariadas)

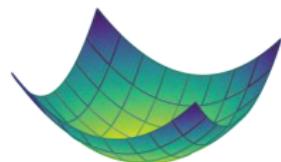
Las siguientes figuras son ejemplos de funciones multivariadas donde se satisface la FONC.



A *local maximum*. The gradient at the center is zero, but the Hessian is negative definite.



A *saddle*. The gradient at the center is zero, but it is not a local minimum.



A *bowl*. The gradient at the center is zero and the Hessian is positive definite. It is a local minimum.

Figura: Tomado de Kochenderfer and Wheeler (2019)

Caso Multivariado. Repaso de Álgebra Lineal

Dado un vector $x \in \mathbb{R}^n$, usamos x_i para denotar a la i -ésima entrada. Se asume que x es un vector *columna*:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

La transpuesta de x , denotada como x^T es un vector *fila*:

$$x = [x_1 \quad x_2 \quad \dots \quad x_n]$$

Usamos $x \geq 0$ para indicar que todas las entradas son no negativas, esto es, $x_i \geq 0$ para todo $i = 1, 2, \dots, n$, mientras que $x > 0$ indica $x_i > 0$ para todo $i = 1, 2, \dots, n$.

Dados $x \in \mathbb{R}^n$ y $y \in \mathbb{R}^n$, el producto punto es $x^T y = \sum_{i=1}^n x_i y_i$.

Caso Multivariado. Repaso de Álgebra Lineal

- Dada una matriz $A \in \mathbb{R}^{m \times n}$, sus entradas las especificamos con los subíndices i,j , es decir, A_{ij} , para $i = 1, 2, \dots, m$ y $j = 1, 2, \dots, n$.
- La transpuesta de A , denotada como A^T , es una matriz $n \times m$ cuyos componentes son A_{ji} .
- Se dice que la matriz A es *cuadrada* si $m = n$.
- Una matriz es simétrica si $A = A^T$.

Caso Multivariado. Repaso de Álgebra Lineal

Una matriz cuadrada A es *positiva definida* si hay un escalar positivo α tal que

$$x^T A x \geq \alpha x^T x, \text{ para todo } x \in \mathbb{R}^n$$

Es *positiva semidefinida* si

$$x^T A x \geq 0, \text{ para todo } x \in \mathbb{R}^n$$

- Una matriz simétrica es **positiva definida** si todos sus eigenvalores son positivos.
- Una matriz simétrica es **positiva semidefinida** si todos sus eigenvalores son no negativos.

Eigen vectores y eigenvalores

Un vector columna $\bar{x} \in \mathbb{R}^n$ es un **eigen vector** de una matriz $A \in \mathbb{R}^{n \times n}$ si se cumple la siguiente expresión para un escalar λ :

$$A\bar{x} = \lambda\bar{x}$$

el escalar λ es llamado **eigen valor**.

Polinomio característico

El polinomio característico de una matriz $A \in \mathbb{R}^{n \times n}$ es el polinomio de grado n en λ obtenido de expandir $\det(A - \lambda I)$

Note que este es un polinomio de grado n siempre tiene n raíces (ya sea repetidas o complejas).

Los eigenvalores son las n raíces del polinomio característico de cualquier matriz $n \times n$.

Observación

El polinomio característico $f(\lambda)$ de una matriz $A \in \mathbb{R}^{n \times n}$ es un polinomio en λ que toma la siguiente forma, donde $\lambda_1, \dots, \lambda_n$ son eigenvalores de A :

$$\det(A - \lambda I) = (\lambda_1 - \lambda)(\lambda_2 - \lambda) \dots (\lambda_n - \lambda)$$

Eigen vectores y eigenvalores: ejemplo

De esta forma, los eigenvalores y eigen vectores de una matriz A pueden calcularse al expandir $\det(A - \lambda I)$, igualar a cero y resolver para λ .

Considera la siguiente matriz :

$$B = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

La matriz $B - \lambda I$ puede ser escrita como:

$$B - \lambda I = \begin{bmatrix} 1 - \lambda & 2 \\ 2 & 1 - \lambda \end{bmatrix}$$

El determinante $\det(B - \lambda I)$ es igual a :

$$\begin{aligned}\det(B - \lambda I) &= (1 - \lambda)(1 - \lambda) - 2 * 2 \\ &= (1 - \lambda)^2 - 4 \\ &= \lambda^2 - 2\lambda - 3\end{aligned}$$

Eigen vectores y eigenvalores: ejemplo

La expresión $\lambda^2 - 2\lambda - 3$ puede ser reescrita como $(3 - \lambda)(-1 - \lambda)$. Al igualar esta expresión a cero,

$$(3 - \lambda)(-1 - \lambda) = 0$$

tenemos que los eigenvalores (las raíces del polinomio) son 3 y -1.

Condiciones para mínimos locales: ejemplo

Considera la función de Rosenbrock:

$$f(\mathbf{x}) = (1 - x_1)^2 + 5(x_2 - x_1^2)^2$$

El punto $(1, 1)$ satisface las condiciones de primer y segundo orden?

El gradiente es:

$$\begin{aligned}\nabla f(\mathbf{x}) &= \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -2(1 - x_1) + 20(x_2 - x_1^2)x_1 \\ 10(x_2 - x_1^2) \end{bmatrix} \\ &= \begin{bmatrix} 2(10x_1^3 - 10x_1x_2 + x_1 - 1) \\ 10(x_2 - x_1^2) \end{bmatrix}\end{aligned}$$

Condiciones para mínimos locales: ejemplo

Sustituyendo el punto $(1, 1)$ en el gradiente, tenemos $\nabla f([1 \ 1]) = [0 \ 0]$ de manera que satisface las condiciones de primer orden.

El hessiano de la función es

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^f}{\partial x_1 \partial x_1} & \frac{\partial^f}{\partial x_1 \partial x_2} \\ \frac{\partial^f}{\partial x_2 \partial x_1} & \frac{\partial^f}{\partial x_2 \partial x_2} \end{bmatrix} = \begin{bmatrix} 2(30x_1^2 - 10x_2 + 1) & 2(-10x_1) \\ 10(-2x_1) & 10 \end{bmatrix}$$

Sustituyendo $(1, 1)$ en el hessiano,

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 42 & -20 \\ -20 & 10 \end{bmatrix}$$

Condiciones para mínimos locales: ejemplo

Que es positiva definida, de forma que cumple con la condición de segundo orden.

$$\nabla^2 f(\mathbf{x}) - \lambda I = \begin{bmatrix} 42 - \lambda & -20 \\ -20 & 10 - \lambda \end{bmatrix}$$

El determinante $\det(\nabla^2 f(\mathbf{x}) - \lambda I)$ es igual a :

$$\begin{aligned}\det(\nabla^2 f(\mathbf{x}) - \lambda I) &= (42 - \lambda)(10 - \lambda) - (-20) * (-20) \\ &= 420 - 42\lambda - 10\lambda + \lambda^2 - 400 \\ &= \lambda^2 - 52\lambda + 20\end{aligned}$$

Con raíces $\lambda_1 = 51,61249695$ y $\lambda_2 = 0,38750305$.

Métodos de Optimización (Murphy, 2022)

- **Métodos de primer orden:** Utilizan información del gradiente⁵, pero ignoran información sobre la curvatura de la función.
- **Métodos de segundo orden:** Utilizan el gradiente e incorporan información sobre la curvatura de la función mediante distintas vías (por ejemplo, mediante el Hessiano), ayudando a tener una convergencia más rápida.
 - ▶ Método de Newton : $\theta_{t+1} = \theta_t - \rho_t H_t^{-1} g_t$
 - ▶ Métodos de Quasi-Newton:
 - ★ BFGS (nombrado así por sus inventores Broyden, Fletcher, Goldfarb y Shanno) ($B_t \approx H_t$, $C_t \approx H_t^{-1}$).
 - ★ Limited memory BFGS (L-BFGS).⁶

Los métodos de quasi-Newton iterativamente construyen una aproximación del Hessiano a partir de información obtenida del gradiente.

⁵Primer derivada en el caso univariado

⁶El paquete de Python sklearn usa L-BFGS como método por default para resolver regresiones logísticas

Métodos de primer orden

Todos los métodos de iteración requieren especificar un punto de inicio θ_0 . En cada iteración t realizan una actualización siguiendo la siguiente regla:

$$\theta_{t+1} = \theta_t + \rho_t d_t \quad (7)$$

donde ρ_t se le conoce como **tamaño de paso** o **tasa de aprendizaje**, y d_t es una **dirección de descenso**.

Métodos de primer orden

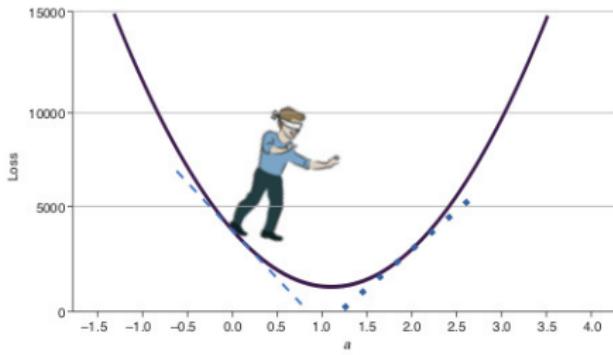


Figura: Tomado de Durr y Sick (2020). *Probabilistic deep learning*

Métodos de primer orden

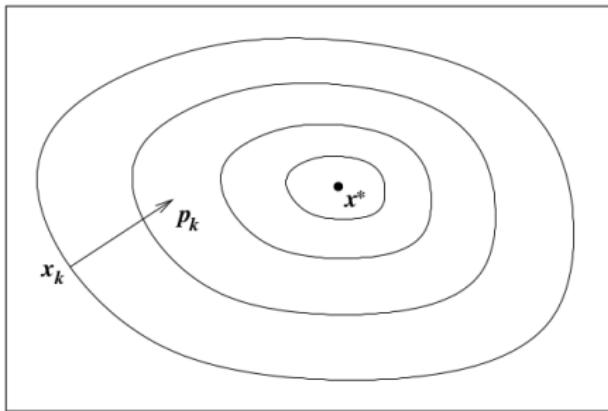


Figure 2.5 Steepest descent direction for a function of two variables.

Figura: Tomado Nocedal and Wright (2006)

Métodos de primer orden: Steepest descent

Cuando la dirección de descenso es igual al negativo del gradiente (*i.e.* $\mathbf{d}_t = -\mathbf{g}_t$)⁷, la dirección se le conoce como de **steepest descent**.

$$\theta_{t+1} = \theta_t - \rho_t \mathbf{g}_t \quad (8)$$

El método de **steepest descent** puede ser muy lento debido a problemas de convergencia.

⁷Recuerda que el gradiente apunta en la dirección de máximo incremento en f , por eso el negativo apunta en la dirección de máxima disminución.

Métodos de primer orden: Steepest descent

Tasa de aprendizaje

El método más sencillo es utilizar una tasa de aprendizaje constante $\rho_t = \rho$.

Sin embargo, si esta tasa es muy grande, el método puede fallar en converger, mientras que si es demasiada pequeña, el método convergerá pero muy lento.

Taller

Ver notebook [steepest_descent.ipynb](#)

Métodos de primer orden: Steepest descent

Problemas del método steepest descent

- Tasa de aprendizaje constante.
- Al no considerar la curvatura de la función a minimizar, el método se mueve muy lentamente en regiones planas de la función.
- Queda atrapado en mínimos locales.

Métodos de primer orden: Steepest descent

Problemas del método steepest descent

- Tasa de aprendizaje constante. Alternativas:
 - ▶ Line Search (Condiciones de Armijo y Wolfe).
 - ▶ RMSprop ([Hinton et al., 2012](#)).
 - ▶ Adagrad y Adadelta ([Duchi et al., 2011](#)).
 - ▶ Adam ([Kingma and Ba, 2014](#))
- Al no considerar la curvatura de la función a minimizar, el método se mueve muy lentamente en regiones planas de la función. Alternativas:
 - ▶ Momentum ([Bertsekas, 1997](#)).
 - ▶ Nesterov momentum ([Nesterov, 2013](#)).
- Queda atrapado en mínimos locales.
 - ▶ Descenso de gradiente estocástico.

Métodos de Momentum

- El método de descenso por gradiente puede moverse muy lentamente en regiones planas de la función.
- Como alternativa, se ha propuesto agregar un término de *momentum* el cuál utiliza una fracción del paso de actualización previo.
- Al incluir este *momentum* se agrega **inercia** al movimiento del método de descenso por gradiente.
- La intuición de la heurística es que el método se mueva más rápido en direcciones que fueron previamente buenas, mientras que se moverá más lento en direcciones donde el gradiente ha cambiado repentinamente.

Métodos de Momentum (Kneusel, 2021)

- En física, el *momentum* de un objeto en movimiento se define como la masa multiplicada por la velocidad, $p = mv$.
- Sin embargo, la velocidad es la primera derivada de la posición del objeto con respecto al tiempo, $v = \frac{dx}{dt}$.
- De forma que el *momentum* es la masa por qué tan rápido está cambiando la posición del objeto en el tiempo.

Métodos de Momentum (Kneusel, 2021)

- Para el método de descenso por gradiente, la *posición* es el valor de la función, y el *tiempo* es el argumento de la función.
- La *velocidad* sería qué tan rápido el valor de la función cambia con un cambio en el argumento de la función, $\frac{\partial f}{\partial x}$.
- De manera que podemos pensar el *momentum* como un término de velocidad escalado.
- En física, el factor de escalamiento es la masa de objeto.
- En descenso por gradiente, el factor de escalamiento es $\beta \in (0, 1)$.

Métodos de Momentum (Murphy, 2022)

Sí agregamos el término de *momentum* \mathbf{m}_t al método de steepest descent el cuál tiene por ecuación de actualización

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \rho \mathbf{g}_t \quad (9)$$

El *momentum* es incorporado de la siguiente manera:

$$\mathbf{m}_{t+1} = \beta \mathbf{m}_t + \mathbf{g}_t \quad (10)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \rho \mathbf{m}_{t+1} \quad (11)$$

Un valor típico de β es 0.9. Para $\beta = 0$, el método se reduce a steepest descent. Podemos pensar β como un factor de escala o e intensidad de inercia. Inicialmente, $\mathbf{m}_t = 0$.

Métodos de Momentum (Murphy, 2022)

Los gradientes pasados exhiben alguna influencia en el presente.

Si

$$\mathbf{m}_t = \beta \mathbf{m}_{t-1} + \mathbf{g}_{t-1} \quad (12)$$

Si sustituimos (10) y (12) en (11)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \rho \beta^2 \mathbf{m}_{t-1} - \rho \mathbf{g}_{t-1} - \rho \mathbf{g}_t \quad (13)$$

Además, \mathbf{m}_t se asemeja a un promedio ponderado exponencialmente de los gradientes pasados.

$$\mathbf{m}_t = \beta \mathbf{m}_{t-1} + \mathbf{g}_{t-1} = \beta^2 \mathbf{m}_{t-2} + \beta \mathbf{g}_{t-2} + \mathbf{g}_{t-1} = \cdots = \sum_{\tau=0}^{t-1} \beta^\tau \mathbf{g}_{t-\tau-1} \quad (14)$$

Nesterov Momentum (Murphy, 2022)

- Un problema en el método estándar de *momentum* es que puede no reducir su velocidad lo suficiente en el fondo de un valle, provocando oscilaciones.
- El método de Nesterov ([Nesterov, 2013](#)) modifica el descenso de gradiente al incluir un paso de extrapolación.
- La idea del método es calcular el gradiente, no en la actual posición, sino en la posición de descenso que se daría si continuara utilizando el *momentum* actual.

Nesterov Momentum (Murphy, 2022)

El método de Nesterov ([Nesterov, 2013](#)) definido en el formato estándar de *momentum* es el siguiente:

$$\mathbf{m}_{t+1} = \beta \mathbf{m}_t - \rho \nabla f(\boldsymbol{\theta}_t + \beta \mathbf{m}_t) \quad (15)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{m}_{t+1} \quad (16)$$

Una razón por la que este método puede ser más rápido que el método estándar es que el vector de *momentum* ya está apuntando aproximadamente en la dirección correcta, por lo que calcular el gradiente en la nueva ubicación $\boldsymbol{\theta}_t + \beta \mathbf{m}_t$, en lugar que en la ubicación actual $\boldsymbol{\theta}_t$, puede ser más preciso.

Métodos adaptativos de descenso de gradiente

- Los métodos consisten en adaptar, de alguna manera, la tasa de aprendizaje durante la ejecución del algoritmo.
- Los algoritmos RMSprop ([Hinton et al., 2012](#)) y Adam ([Kingma and Ba, 2014](#)) incorporan un mecanismo de seguimiento del valor del gradiente mediante el cálculo de una media móvil de la trayectoria del algoritmo, la cual utilizan para ajustar la tasa de aprendizaje.

Métodos adaptativos de descenso de gradiente

Específicamente, calculan una media móvil ponderada exponencialmente:⁸

$$\hat{\mu}_t = \beta \mu_{t-1} + (1 - \beta) \mathbf{y}_t \quad (17)$$

donde $0 < \beta < 1$. La contribución de un punto en el paso k en el pasado es ponderado por $\beta^k(1 - \beta)$. La contribución de los primeros datos disminuye exponencialmente.

Dado que $0 < \beta < 1$, tenemos que $\beta^{t+1} \rightarrow 0$ en la medida que $t \rightarrow \inf$, de manera que a valores pequeños de β , el pasado se olvida más rápidamente, mientras que se adapta más rápido a los datos más recientes.

⁸Exponentially weighted moving average (EWMA). Ver Murphy (2022)

Métodos adaptativos de descenso de gradiente

Dado que inicialmente la estimación inicia con $\hat{\mu}_0 = \mathbf{0}$, hay un sesgo inicial. Para corregir el sesgo, Kingma and Ba (2014) sugieren escalar con la siguiente regla

$$\tilde{\mu}_t = \frac{\hat{\mu}_t}{1 - \beta^t} \quad (18)$$

Métodos adaptativos de descenso de gradiente

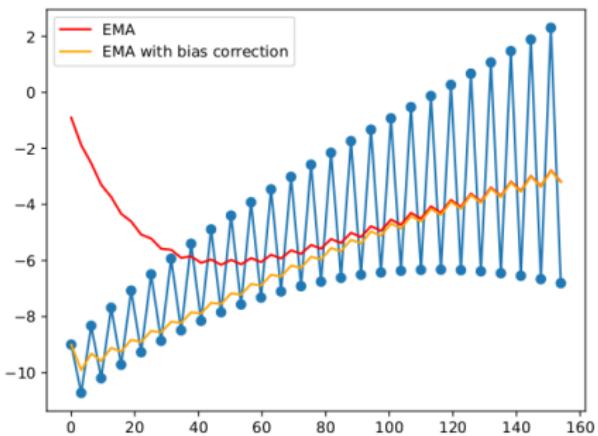


Figura: $\beta = 0,9$

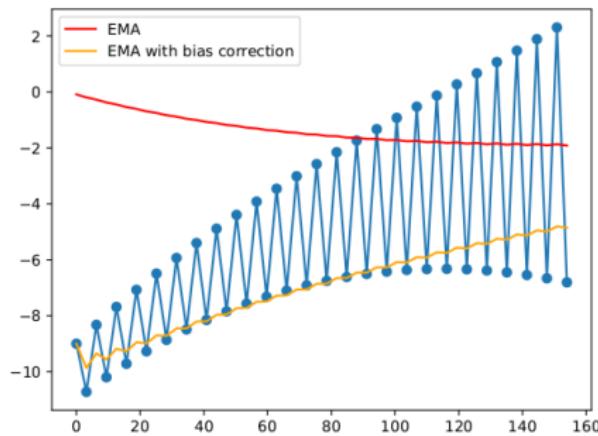


Figura: $\beta = 0,99$

Adam

Kingma and Ba (2014) propusieron el método *Adam* (Adaptive moment estimation). El método usa el cuadrado del gradiente así como un término de *momentum*. Las reglas de actualización son las siguientes:

$$\begin{aligned}\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + \beta_1(1 - \beta_1) \mathbf{g}_t \\ \mathbf{s}_t &= \beta_2 \mathbf{s}_{t-1} + \beta_2(1 - \beta_1) \mathbf{g}_t^2 \\ \tilde{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1 - \beta_1^t} \\ \tilde{\mathbf{s}}_t &= \frac{\mathbf{s}_t}{1 - \beta_2^t} \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \rho \frac{1}{\sqrt{\tilde{\mathbf{s}}_t} + \epsilon} \tilde{\mathbf{m}}_t\end{aligned}$$

Los valores estándar de los parámetros son $\beta_1 = 0,9$, $\beta_2 = 0,999$ y $\epsilon = 10^{-6}$. Si definimos $\beta_1 = 0$ y quitamos la corrección del sesgo, obtenemos el método RMSprop (Hinton et al., 2012) que no utiliza *momentum*. Se suele utilizar una tasa de aprendizaje constante igual a $\rho = 0,001$.

Métodos de primer orden : Line search

Elige una dirección p_k y busca en esa dirección desde algún valor x_k algún punto x_{k+1} tal que $f(x_{k+1}) < f(x_k)$.

La distancia a moverse puede encontrarse resolviendo:

$$\min_{\alpha > 0} \quad f(x_k + \alpha p_k)$$

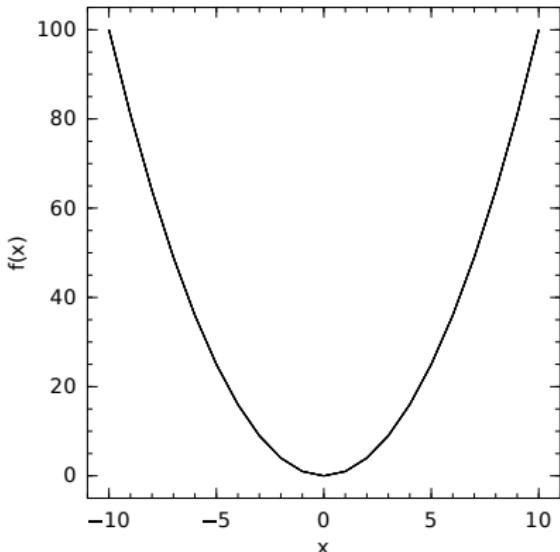


Figura: $f(x) = x^2$

Métodos de primer orden

Elige una dirección p_k y busca en esa dirección desde algún valor x_k algún punto x_{k+1} tal que $f(x_{k+1}) < f(x_k)$.

La distancia a moverse puede encontrarse resolviendo:

$$\min_{\alpha > 0} f(x_k + \alpha p_k)$$

¿Cómo calculamos α ?

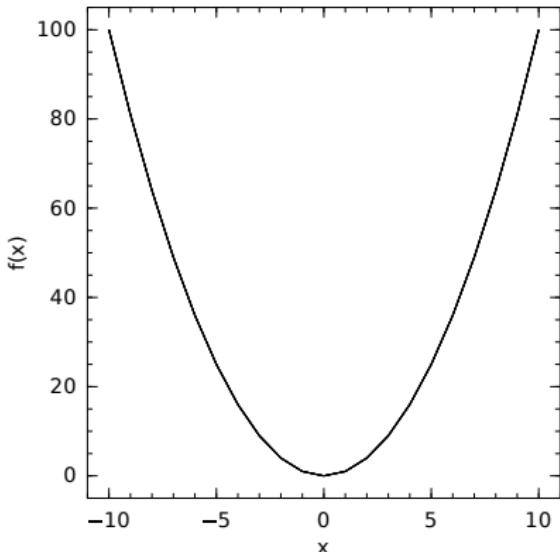


Figura: $f(x) = x^2$

El tamaño de paso

- Nos interesa elegir un valor de α_k que reduzca sustancialmente f pero también queremos decidir rápido.
- Una opción sería encontrar una alfa que diera como resultado

$$f(x_k + \alpha_k p_k) < f(x_k)$$

El tamaño de paso

- Nos interesa elegir un valor de α_k que reduzca sustancialmente f pero también queremos decidir rápido.
- Una opción sería encontrar una alfa que diera como resultado

$$f(x_k + \alpha_k p_k) < f(x_k)$$

PROBLEMA: Mala convergencia

El tamaño de paso

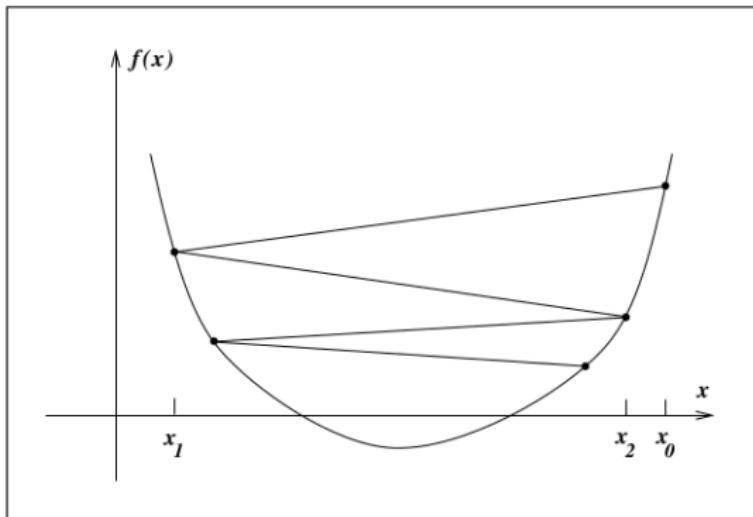


Figure 3.2 Insufficient reduction in f .

Figura: Tomado Nocedal and Wright (2006)

Condiciones de Wolfe

Las condiciones de Wolfe sugieren que α_k consiga un decremento suficiente en f medido por:

$$f(x_k + \alpha_k p_k) < f(x_k) + c_1 \alpha \nabla f_k^T p_k$$

para alguna constante $c_1 \in (0, 1)$. En la práctica, $c_1 = 10^{-4}$

Condiciones de Wolfe

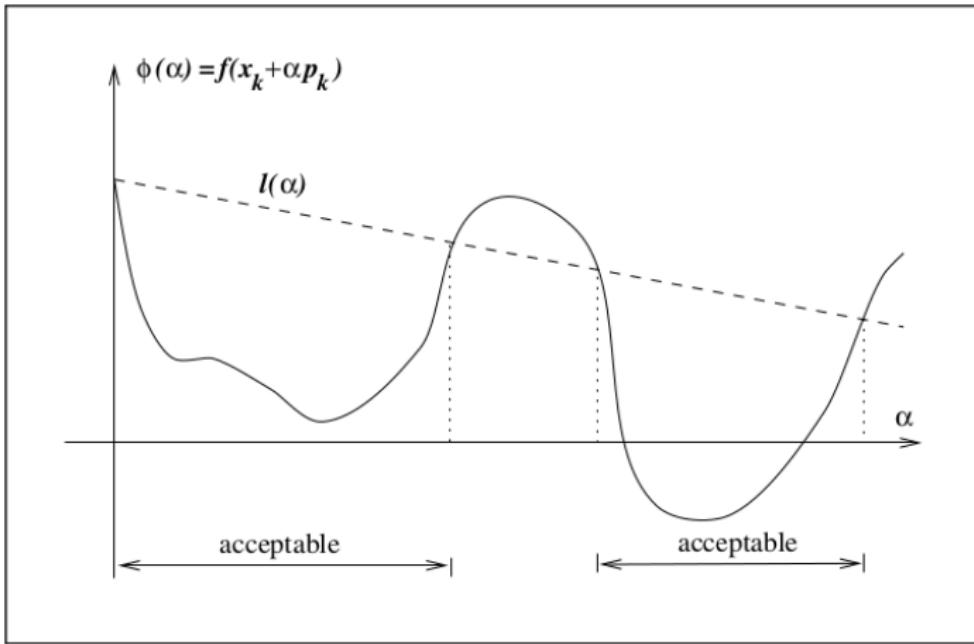


Figure 3.3 Sufficient decrease condition.

Figura: Tomado Nocedal and Wright (2006)

Gradiente descendente: el algoritmo

Algorithm 1: Gradiente descendente

Input: $f, \nabla f, \mathbf{x}_0 \in \mathbb{R}^n, \epsilon$

Output: \mathbf{x}_{k+1}

$\mathbf{x}_k \leftarrow \mathbf{x}_0$

while $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| > \epsilon$ **do**

$p_k \leftarrow -\nabla f(\mathbf{x}_k)$

Calcular tamaño de paso α_k

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k p_k$

end while

return \mathbf{x}_{k+1}

Gradiente descendente: el algoritmo

Algorithm 2: Gradiente descendente

Input: $f, \nabla f, \mathbf{x}_0 \in \mathbb{R}^n, \epsilon$

Output: \mathbf{x}_{k+1}

$\mathbf{x}_k \leftarrow \mathbf{x}_0$

while $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| > \epsilon$ **do**

$p_k \leftarrow -\nabla f(\mathbf{x}_k)$

Calcular tamaño de paso α_k *¿cómo lo calculamos?*

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k p_k$

end while

return \mathbf{x}_{k+1}

Algorithm 3: Gradiente descendente

Input: $f, \nabla f, \mathbf{x}_0 \in \mathbb{R}^n, \epsilon, \tilde{\alpha}, c, \rho$

Output: \mathbf{x}_{k+1}

$\mathbf{x}_k \leftarrow \mathbf{x}_0$

while $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| > \epsilon$ **do**

$p_k \leftarrow -\nabla f(\mathbf{x}_k)$

 /* ----- Algoritmo backtracking para tamaño de paso ----- */

$\alpha_k \leftarrow \tilde{\alpha}$

while $f(\mathbf{x}_k + \alpha_k p_k) \leq f(\mathbf{x}_k) + c \alpha_k \nabla f_k^\top p_k$ **do**

$\alpha_k \leftarrow \rho \alpha_k$

end while

 /* ----- */

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k p_k$

end while

return \mathbf{x}_{k+1}

Métodos de primer orden

Taller

Ver notebook [descenso_grad.ipynb](#)

Funciones de prueba

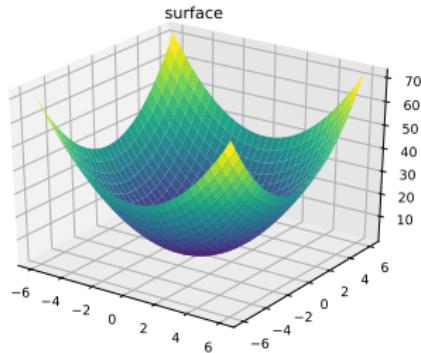


Figura: $f(x_1, x_2) = x_1^2 + x_2^2$

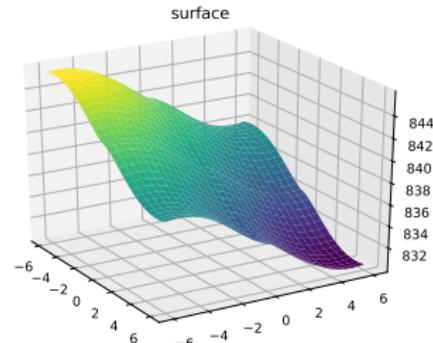


Figura:
 $f(x_1, x_2) = 418,9829 * 2 - x_1 \sin \sqrt{|x_1|} - x_2 \sin \sqrt{|x_2|}$

References I

- Aggarwal, C. C. (2020). *Linear Algebra and Optimization for Machine Learning - A Textbook*. Springer.
- Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334.
- Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).
- Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kneusel, R. T. (2021). *Math for deep learning: what you need to know to understand neural networks*. No Starch Press.

References II

- Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for optimization*. MIT Press.
- Murphy, K. P. (2022). *Probabilistic machine learning: an introduction*. MIT press.
- Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.
- Neumaier, A. (2004). Complete search in continuous global optimization and constraint satisfaction. *Acta numerica*, 13:271–369.
- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.