

Métodos de optimización

BIG DATA CON PYTHON

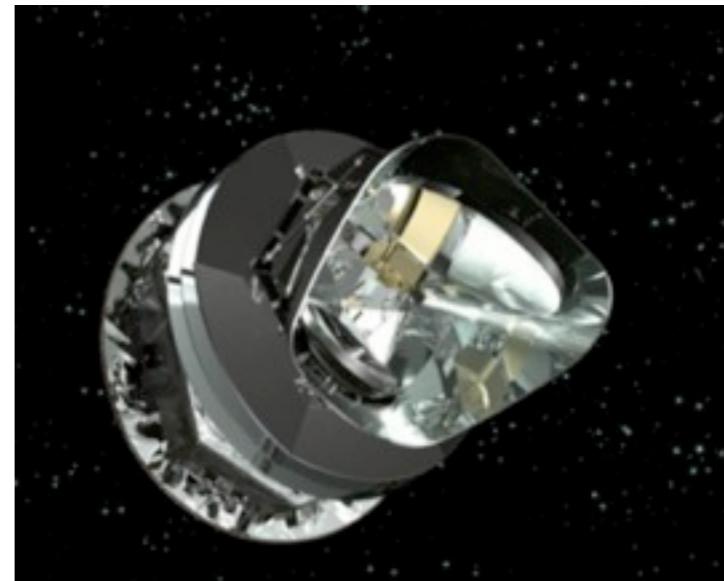
Biuse Casaponsa

Investigadora Posdoctoral IFCA

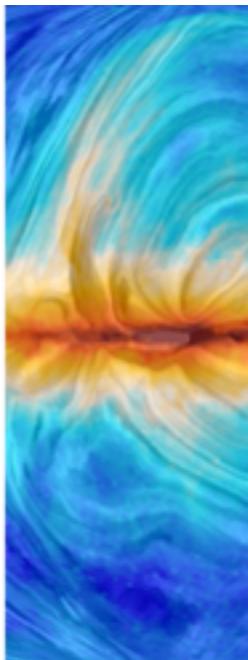
INTRODUCTION TO
PYTHON FOR BIG DATA



Doctorado en análisis del CMB

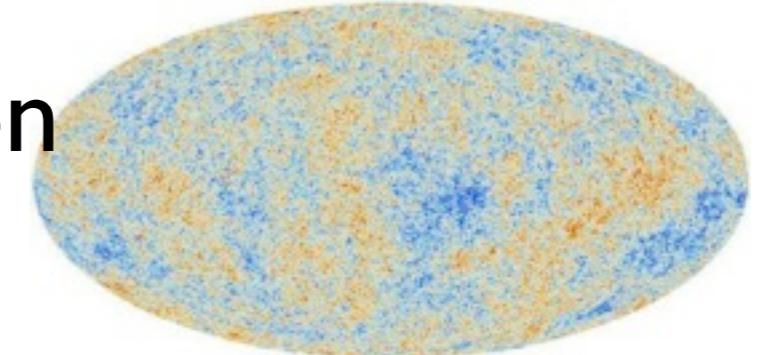


Ultimate modelling of
Radio Foregrounds:
a key ingredient for Cosmology



Planck collaboration

Radioforegrounds Project



Colaboraciones puntuales

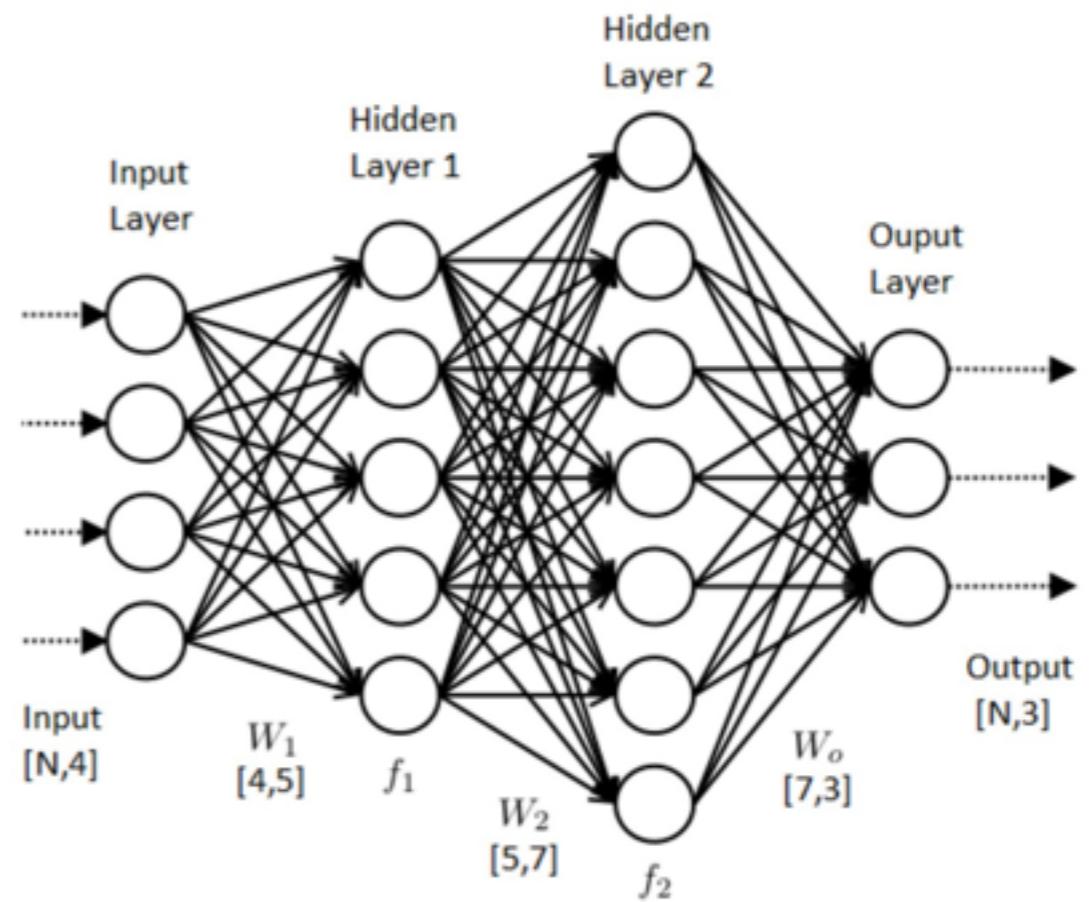
Análisis de datos

Datos cosmológicos: obtención de parámetros de modelos, estadística del CMB, separación de componentes

Redes neuronales

Python ¿Por qué?

-Es sencillo, librerías gráficas muy buenas, hay mucha documentación, muchas librerías. Lo usa mucha gente, es fácil compartir código, ...ah! y **gratuito!**



Métodos de optimización

BIG DATA CON PYTHON

1. Introducción a la optimización

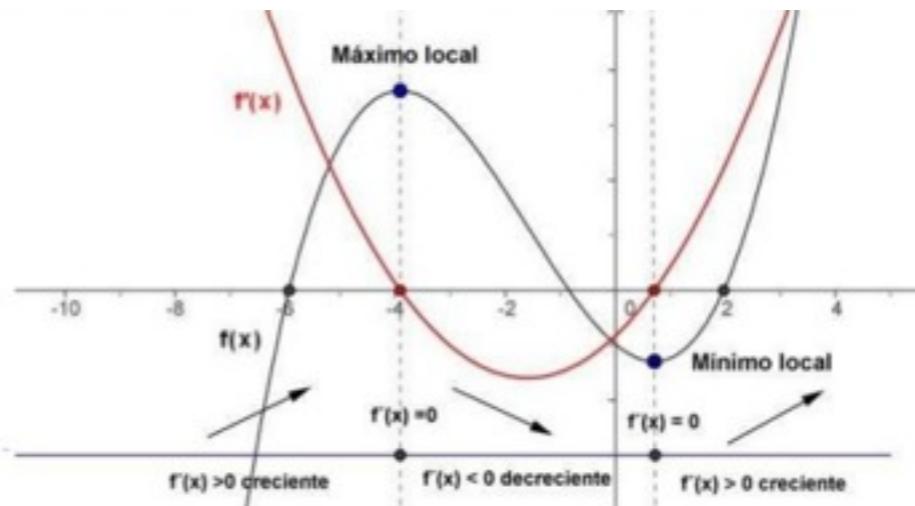
- Tipos de problemas
- Big Data problems

2. Resolver problemas que requieren optimización

- Solución analítica, grid,algoritmos
- Optimizar con Python

3. Optimización lineal

- Solución geométrica
- Métodos numéricos
- Ejemplos con python (scipy.optimize)



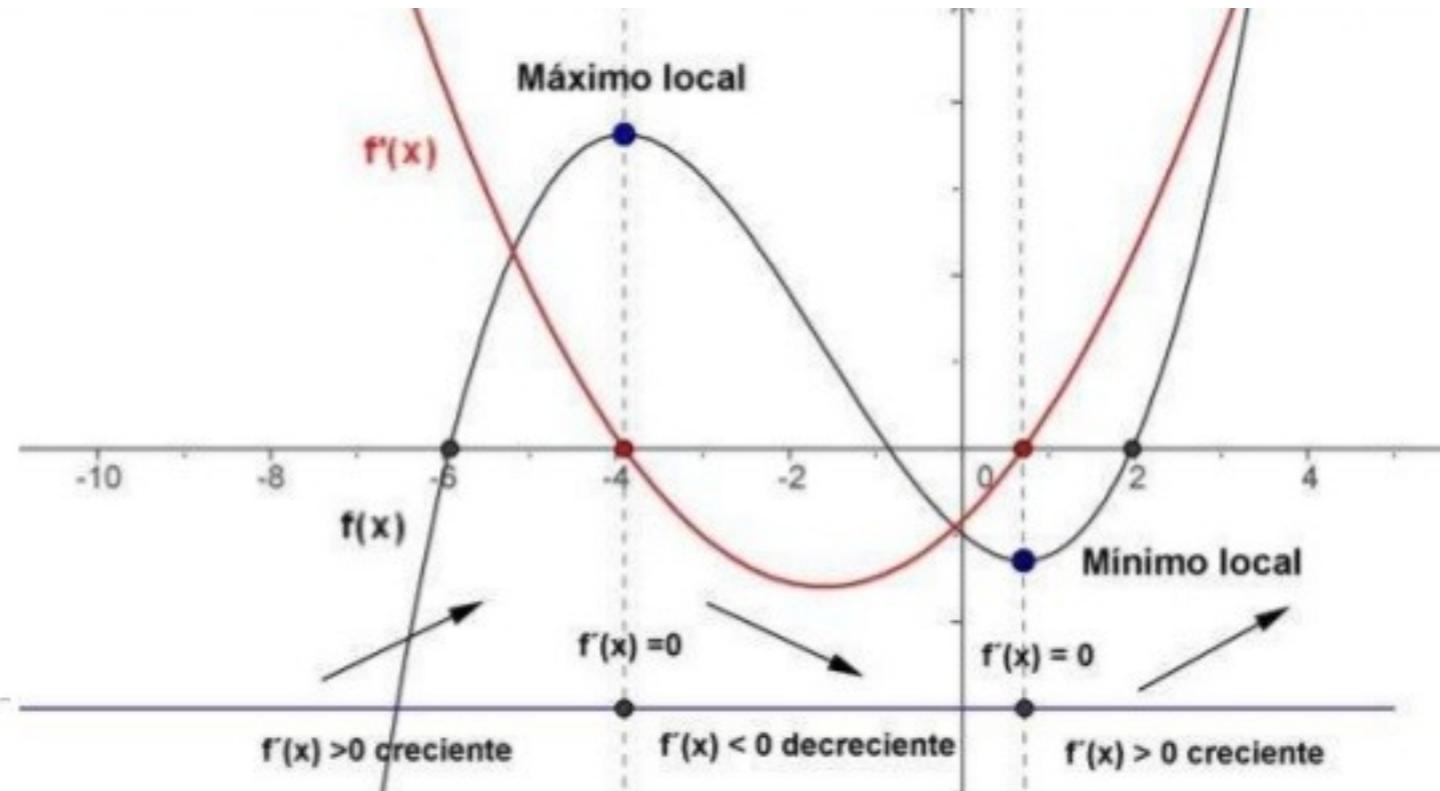
4. Optimización no lineal

- Programación cuadrática,
ejemplo con Python

5. Modelado de datos

- mínimos cuadrados
- maximum likelihood
- Ejemplos con python (numpy.polyfit and scipy.curve_fit)

Introducción: Problemas Optimización



Ejercicios colegio: lo resolvíamos de forma analítica:

$$\frac{dF}{dx} = 0$$

$$\frac{d^2F}{dx^2} > 0$$

$$\frac{d^2F}{dx^2} = 0$$

$$\frac{d^2F}{dx^2} < 0$$

Introducción: Problemas Optimización

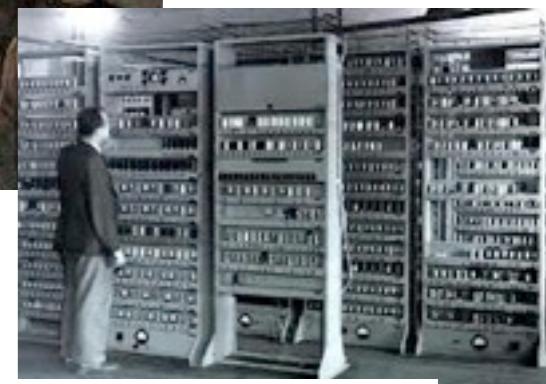
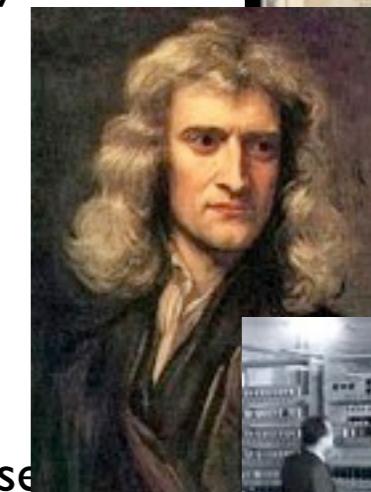
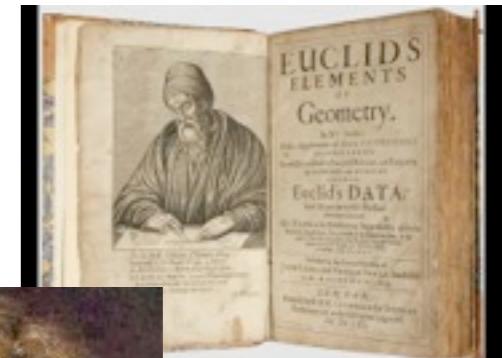
Optimización: Escoger el mejor elemento (dadas unas condiciones) de entre varias alternativas disponibles.

- Todos optimizamos (o intentamos) constantemente los problemas que nos surgen, aunque no lo formulemos matemáticamente:
 - ¿Cuál es el camino más rápido para volver a casa hoy?
 - ¿Cómo será mejor que me organice el verano, dedico tiempo a formarme o mejor me voy de vacaciones?

Lo más común es plantear problemas que requieran minimizar o maximizar una función.

Historia Optimización

- Desde que el hombre tiene uso de razón y se plantea problemas ha tenido que encontrar soluciones óptimas para resolverlos
- Los primeros problemas formulados fueron en la antigua Grecia: Euclides (300 a.c.) resuelve la distancia más corta entre una recta y un punto
- S. XVII, Fermat demuestra que la derivada en un extremo de una función es cero y más adelante Newton propone un método numérico para encontrar estos extremos. En el Siglo XVIII se desarrolla calculo diferencial (Euler, Lagrange, ...)
- En el S. XIX se presentan los primeros algoritmos de optimización, (gradient descent, minimos cuadrados,...). Se introduce concepto de optimización lineal (Fourier) y resolución de sistemas lineales (Gauss)
- En el S.XX hay un gran desarrollo de nuevos métodos y variantes de los clásicos, se desarrolla la inteligencia artificial.
 - Con la llegada de los ordenadores en los 50s, se pueden resolver problemas más complejos, con multitud de variables, con funciones no lineales
- En el S. XXI el gran avance no es tanto en los algoritmos si no en la cantidad de datos que manejamos y el avance tecnológico en rapidez y capacidad de los ordenadores. Esto tiene especial importancia para obtener modelos predictivos.



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Problemas de Logística

- Mantener stock óptimo, de tal manera que pueda satisfacer la demanda y que me quepa en el almacén
- Cargar camiones de manera más eficiente.
- Escoger ruta óptima de reparto o de ventas. (Si hay muchas ciudades o muchos repartidores, ...)
-



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Problemas de Logística

- Mantener stock óptimo, de tal manera que pueda satisfacer la demanda y que me quepa en el almacén
- Cargar camiones de manera más eficiente.
- Escoger ruta óptima de reparto o de ventas. (Si hay muchas ciudades o muchos repartidores, ...)
-



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Problemas de Logística

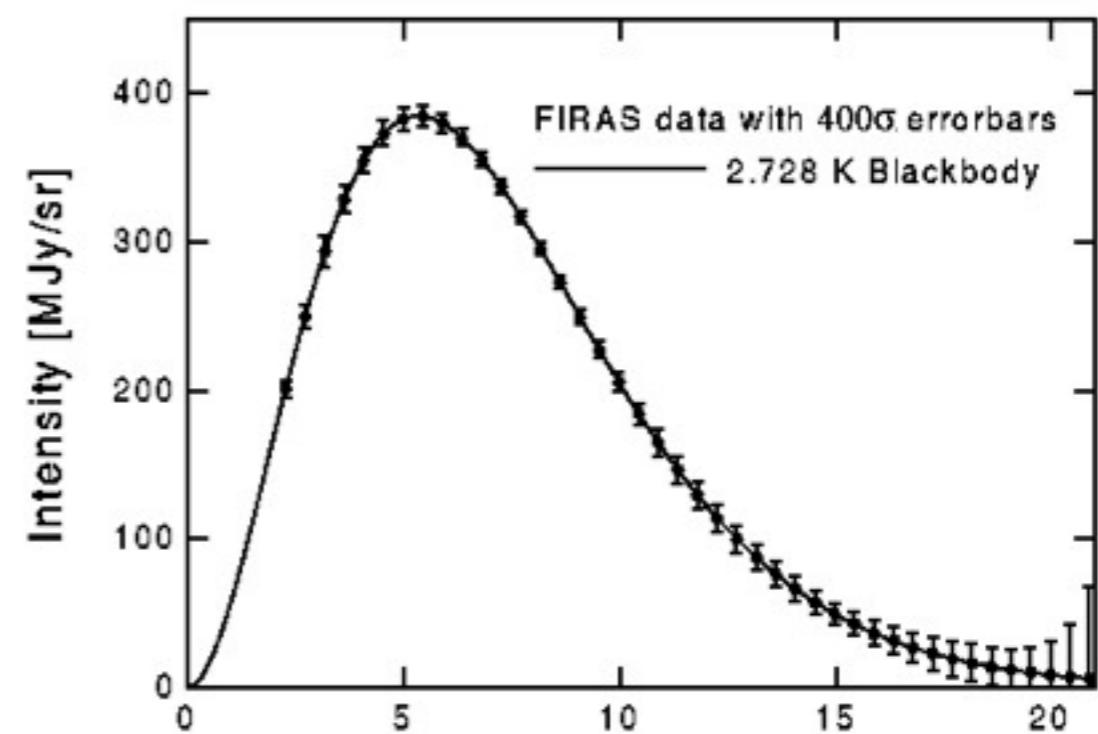
- Mantener stock óptimo, de tal manera que pueda satisfacer la demanda y que me quepa en el almacén
- Cargar camiones de manera más eficiente.
- Escoger ruta óptima de reparto o de ventas. (Si hay muchas ciudades o muchos repartidores, ...)
-



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Ciencia

- **Diseño instrumentación.** Como hacer un satélite con el mínimo coste que me permita obtener la máxima resolución. O el mínimo pero
- **Modelado de datos.** Encontrar parámetros de un modelo teórico que se ajustan más a los datos observados. Ej. la radiación de fondo de microondas se comporta como un cuerpo negro, con temperatura T



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Otros

- **Diseño puentes, carreteras, instrumentos.** Maximizar resistencia, maximizar ligereza, Minimizar costes,
- **Medio Ambiente.** Minimizar impacto ecológico de alguna medida, de alguna infraestructura. Maximizar eficiencia energética con menor contaminación,...
- **Ventas.** Maximiza probabilidad de venta, clasificar cliente,...
- **Finanzas, sociología,**



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Otros

- **Diseño puentes, carreteras, instrumentos.** Maximizar resistencia, maximizar ligereza, Minimizar costes,
- **Medio Ambiente.** Minimizar impacto ecológico de alguna medida, de alguna infraestructura. Maximizar eficiencia energética con menor contaminación,...
- **Ventas.** Maximiza probabilidad de venta, clasificar cliente,...
- **Finanzas, sociología,**



Al final en todos estos problemas lo que tenemos es un objetivo (función coste o **función objetivo**) dadas una serie de **condiciones** y un **método** para encontrar cómo satisfacer ese objetivo.

PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y cuantificarlo (poder definirlo de manera matemática, con números).
- Definir variables.
- Definir función objetivo: qué vamos a minimizar o maximizar con respecto a esas variables.
- Definir ligaduras entre las variables (si las hay)
- Escoger algoritmo para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.

Big Data

Además de los problemas tradicionales, ahora se abre un mundo de posibilidades: **modelos predictivos**

- Somos capaces de recoger muchos datos: de obtenerlos y de almacenarlos
- **Ventaja.** Gran cantidad de datos nos permite obtener **modelos predictivos**, dado un conocimiento del comportamiento anterior o de otros ser capaces de generalizar.
- **Algoritmos potentes.** Para obtener esos modelos de manera rápida y efectiva se necesitan algoritmos potentes (**MACHINE LEARNING**)
- Reconocimiento de patrones, clasificación de objetos, identificación,
- Al final tener un buen modelo predictivo también es un ejercicio de optimización: **maximizar la probabilidad de acertar.**

RESOLVER PROBLEMAS de OPTIMIZACIÓN

PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y cuantificarlo (poder definirlo de manera matemática, con números).
- Definir variables.
- Definir **función objetivo**: qué vamos a minimizar o maximizar con respecto a esas variables.
- Definir ligaduras entre los parámetros
- Escoger algoritmo para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.



PASOS BÁSICOS OPTIMIZACIÓN

- Pensar problema y cuantificarlo (poder definirlo de manera matemática, con números). **LO MAS IMPORTANTE**

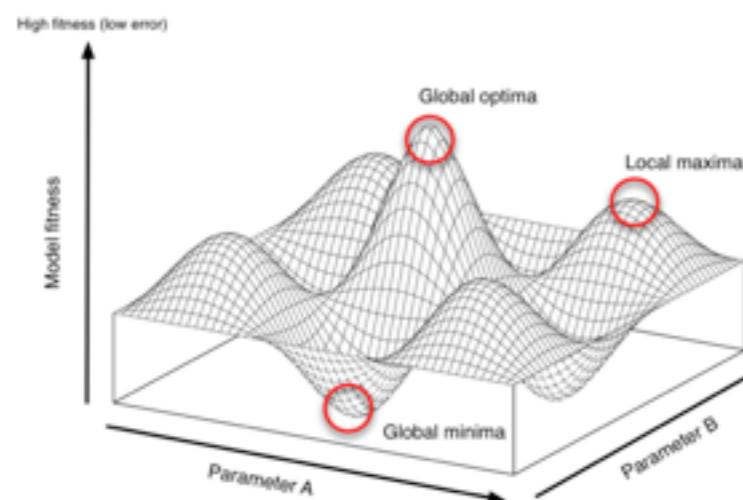
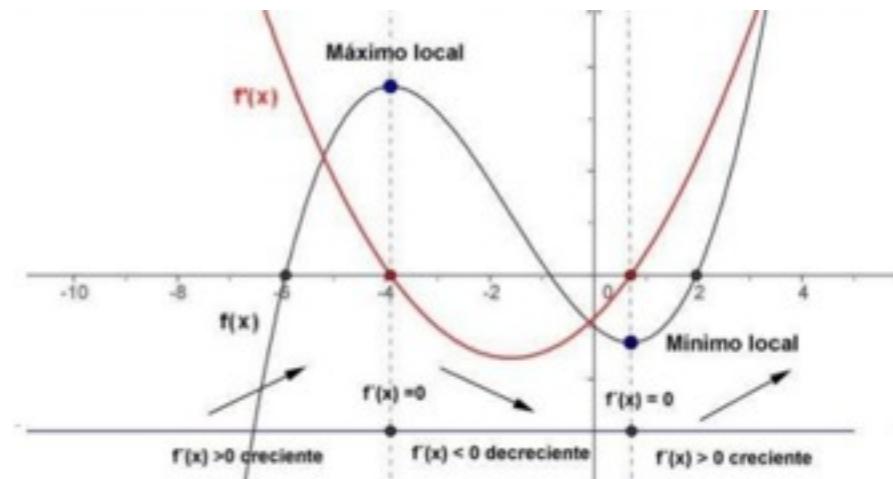
- Definir variables.
- Definir **función objetivo**: qué vamos a minimizar o maximizar con respecto a esas variables.
- Definir ligaduras entre los parámetros
- Escoger algoritmo para encontrar el valor de los parámetros que hacen mínima (o maxima) la función objetivo.



Métodos de Optimización

1. Solución analítica. Si podemos, lo más sencillo es buscar la solución analítica. Encontrar derivada e igualar a 0.

2. Grid. Podemos calcular todos los posibles resultados, y coger el más óptimo (en general no factible)



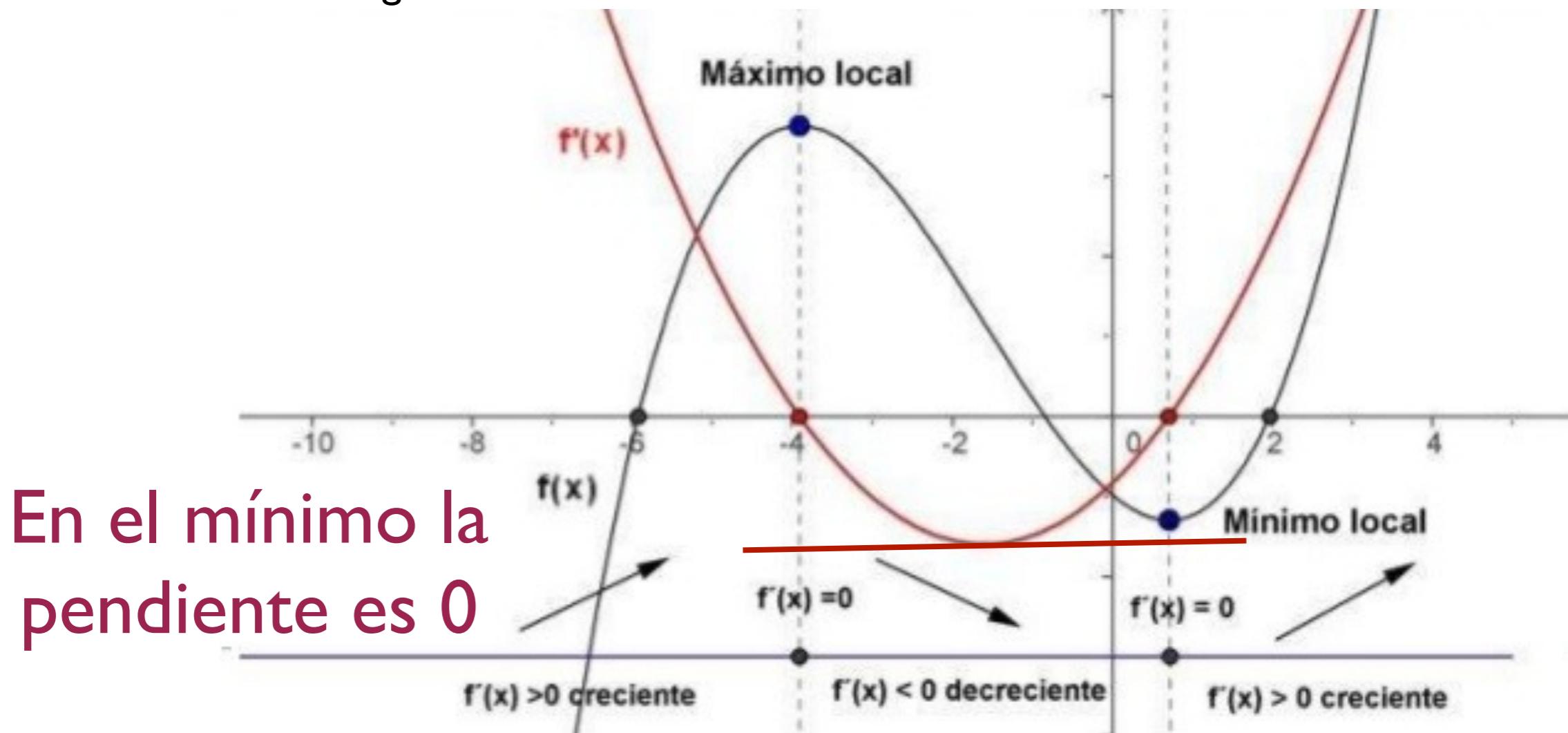
3. Algoritmo numérico. Buscar el mínimo de la función con un algoritmo iterativo que me lleve al mínimo

4. Explorar el espacio de parámetros. De manera más eficiente que calcularlos todos

Métodos de Optimización

Solución analítica.

La derivada en un punto de la parábola es la pendiente de la recta tangente.



Resolver problemas que requieren optimización.

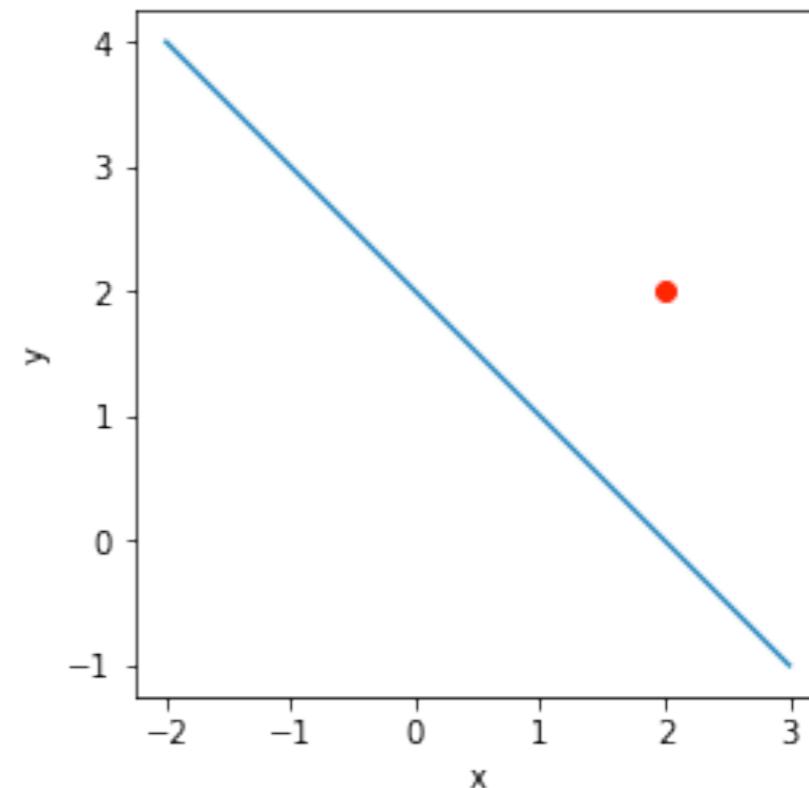
Ejemplo sencillo: Queremos encontrar el punto de la línea $y + x = 2$ que se encuentre más cerca del punto $p = (2, 2)$

- Definir **variables**: x e y
 - **función objetivo**: distancia
- $$D = (y - 2)^2 + (x - 2)^2$$
- Escogemos método para encontrar x e y que minimicen la distancia:

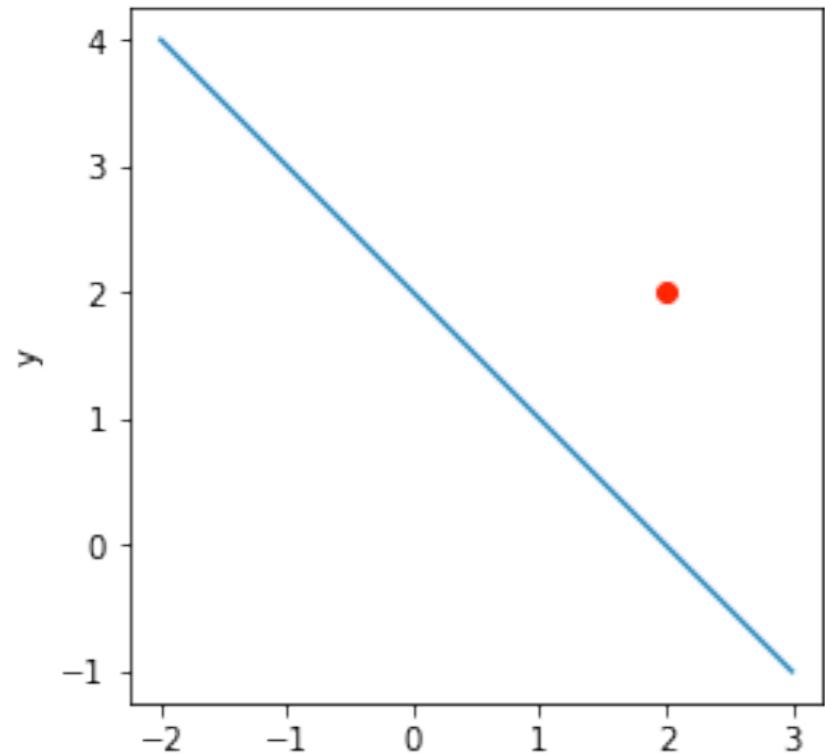
I. - Solución analítica

2. - Grid (calcular la función en todos los puntos)

3. - Algoritmo de minimización



Resolver problemas que requieren optimización. De forma analítica



Y derivamos: $\frac{dD}{dx} = 2x + 2(x - 2) = 0 \longrightarrow 4x = 4$

Dejamos la distancia en función de una sola variable:

$$D = (y - 2)^2 + (x - 2)^2$$

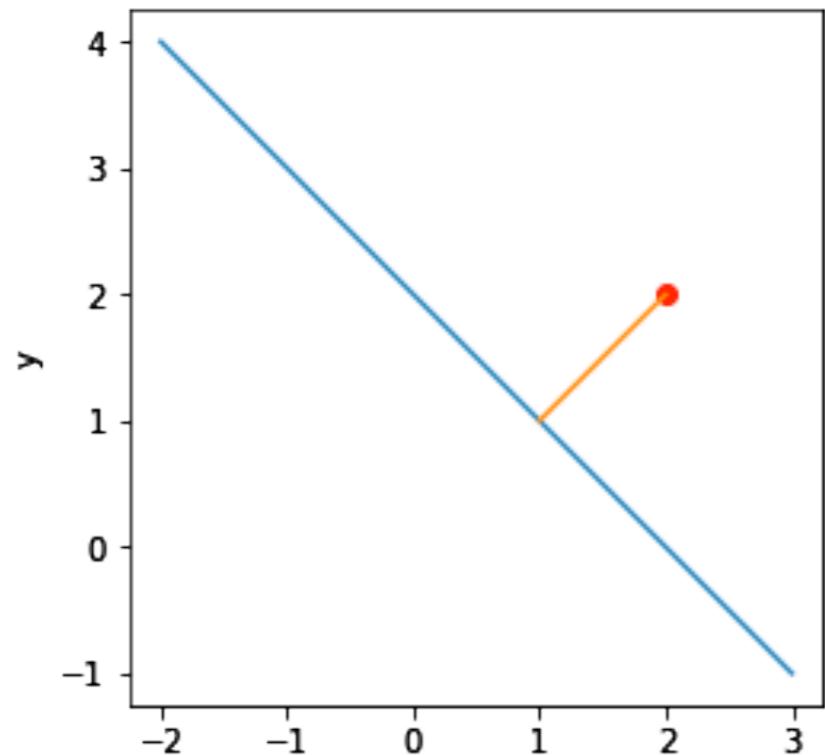
dado $y + x = 2$



$$D = (-x)^2 + (x - 2)^2$$

El punto $p = (1, 1)$ es el que resuelve el problema.

Resolver problemas que requieren optimización. De forma analítica



Y derivamos: $\frac{dD}{dx} = 2x + 2(x - 2) = 0 \longrightarrow 4x = 4$

Dejamos la distancia en función de una sola variable:

$$D = (y - 2)^2 + (x - 2)^2$$

dado $y + x = 2$

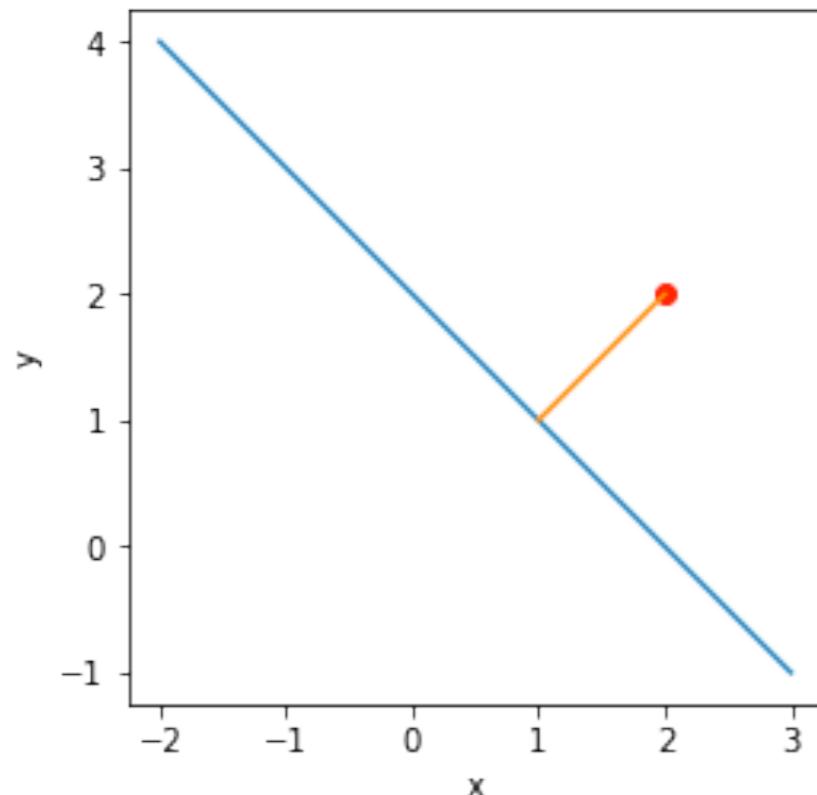


$$D = (-x)^2 + (x - 2)^2$$

El punto $p = (1, 1)$ es el que resuelve el problema.

Resolver problemas que requieren optimización. Grid

Ejemplo sencillo: Queremos encontrar el punto de la línea $y + x = 2$ que se encuentre más cerca del punto mínimo al punto $p = (2, 2)$



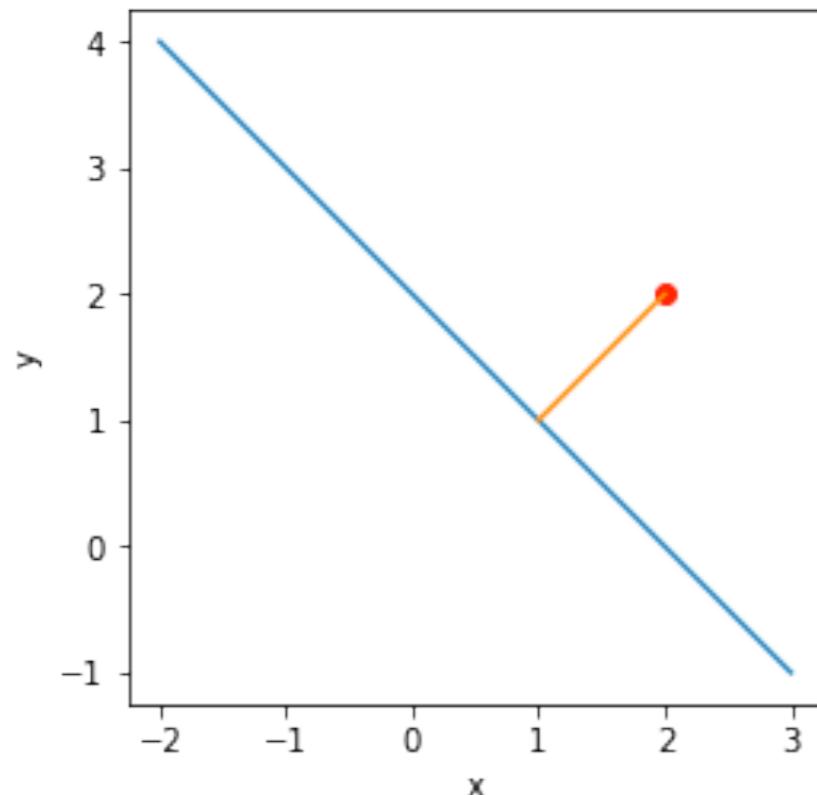
Calculamos la distancia de todos los puntos de la recta :

x	y	D
-3	5	34
-2	4	20
-1	3	10
0	2	4
1	1	2
2	0	4
3	-1	10

Llegamos a la misma solución, pero cuando hay muchas variables o el espacio de parámetros es muy amplio (en este caso estaba limitado por la recta) es imposible (computacionalmente).

Resolver problemas que requieren optimización. Grid

Ejemplo sencillo: Queremos encontrar el punto de la línea $y + x = 2$ que se encuentre más cerca del punto mínimo al punto $p = (2, 2)$



Calculamos la distancia de todos los puntos de la recta :

x	y	D
-3	5	34
-2	4	20
-1	3	10
0	2	4
1	1	2
2	0	4
3	-1	10

Llegamos a la misma solución, pero cuando hay muchas variables o el espacio de parámetros es muy amplio (en este caso estaba limitado por la recta) es imposible (computacionalmente).

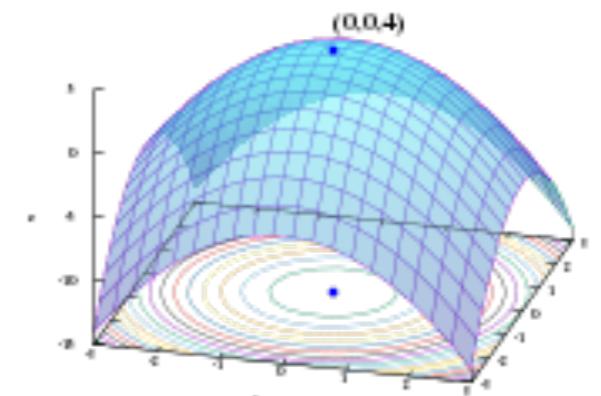
Resolver problemas que requieren optimización.

- Pero no siempre tenemos problemas tan sencillos
 - Hay veces que no se puede sacar solución analítica
 - Y hacer un grid es imposible (muchas dimensiones, espacio de parámetros muy grande)

Si la función analítica es muy complicada buscaremos el mínimo con un otro método.

Resolver problemas que requieren optimización.

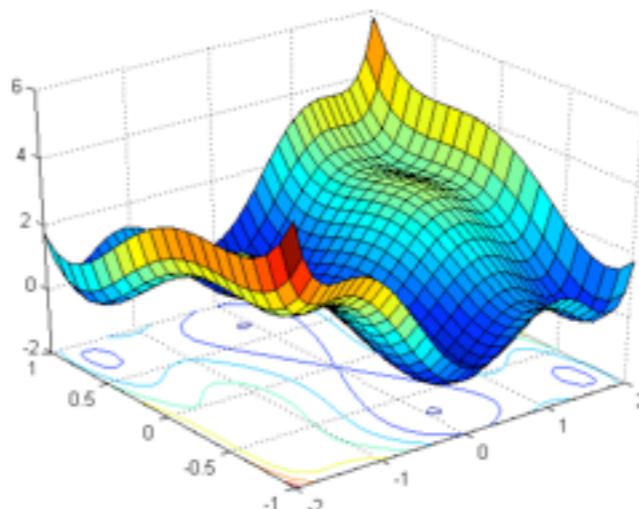
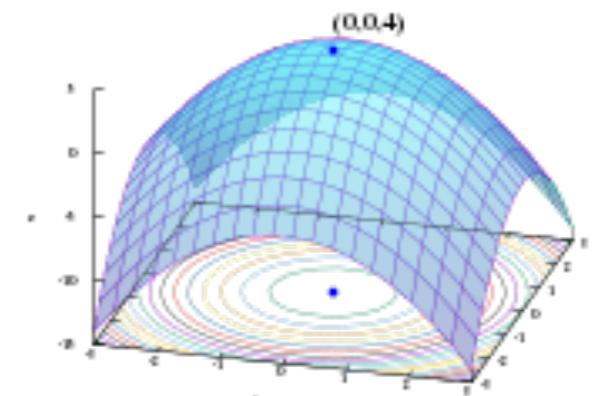
- Pero no siempre tenemos problemas tan sencillos
 - Hay veces que no se puede sacar solución analítica
 - Y hacer un grid es imposible (muchas dimensiones, espacio de parámetros muy grande)



Si la función analítica es muy complicada buscaremos el mínimo con un otro método.

Resolver problemas que requieren optimización.

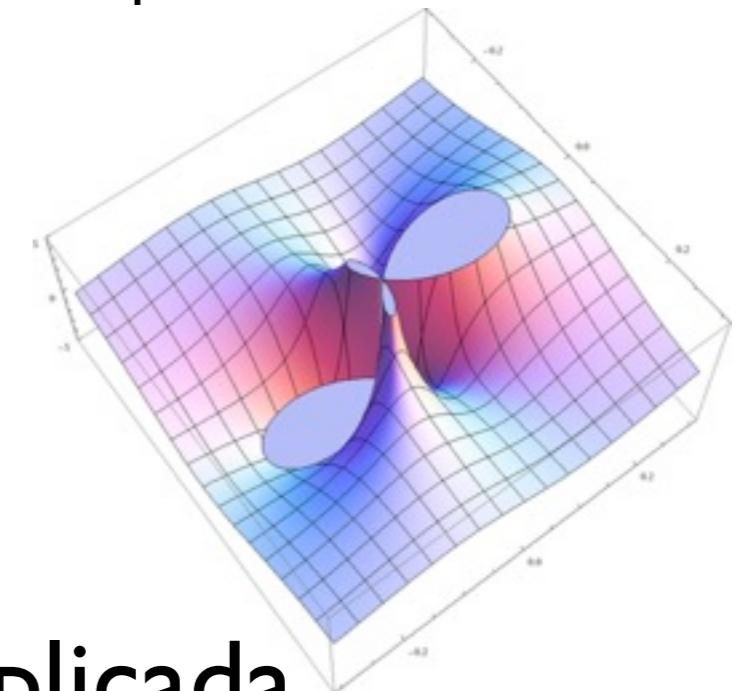
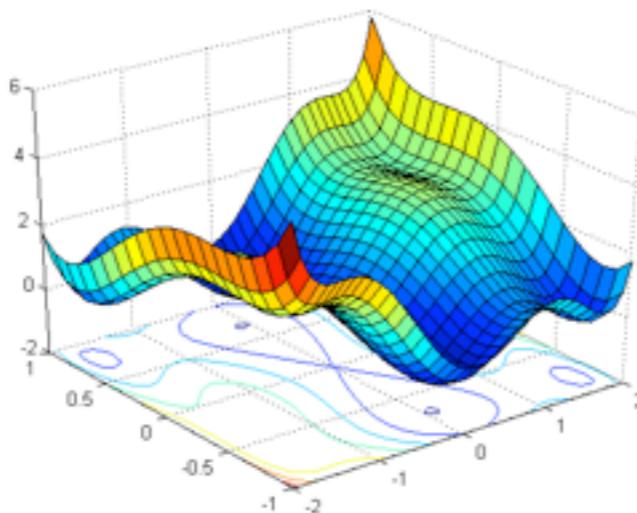
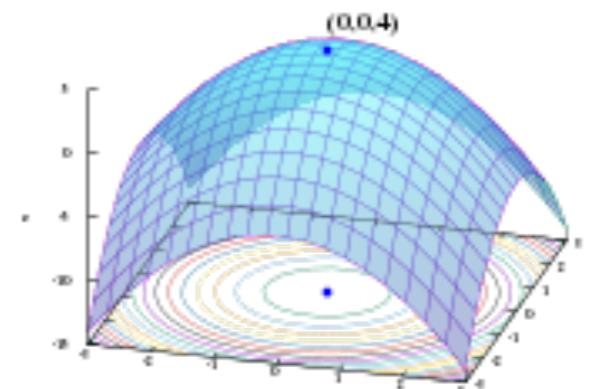
- Pero no siempre tenemos problemas tan sencillos
 - Hay veces que no se puede sacar solución analítica
 - Y hacer un grid es imposible (muchas dimensiones, espacio de parámetros muy grande)



Si la función analítica es muy complicada buscaremos el mínimo con un otro método.

Resolver problemas que requieren optimización.

- Pero no siempre tenemos problemas tan sencillos
 - Hay veces que no se puede sacar solución analítica
 - Y hacer un grid es imposible (muchas dimensiones, espacio de parámetros muy grande)



Si la función analítica es muy complicada buscaremos el mínimo con un otro método.

TIPOS de ALGORITMOS de OPTIMIZACIÓN

USAN GRADIENTE (y/o segundas derivadas)	NO USAN GRADIENTE	FANCY
<ul style="list-style-type: none">- Métodos que necesitan tener derivadas definidas- Son los más usados para funciones continuas derivables para problemas sin ligaduras-Gradient Descent-BFGS, Levenberg-Marquardt, Gauss-Newton, gradiente-conjugado-Solución analítica	<ul style="list-style-type: none">- No necesitan derivada definida- Simplex-Nelder-Mead-Random Walk-Grid-Métodos de Monte-Carlo	<ul style="list-style-type: none">-Métodos más sofisticados o creativos-Ant colony-Bat colony-Brain storming optimization-Útiles para abordar problemas de difícil solución (NP-Hard)

TIPOS de ALGORITMOS de OPTIMIZACIÓN

FAMILIA GRADIENTE :

Función objetivo tiene que ser derivable!

Gradient descent (ascent for maximum) -- Cauchy 1847

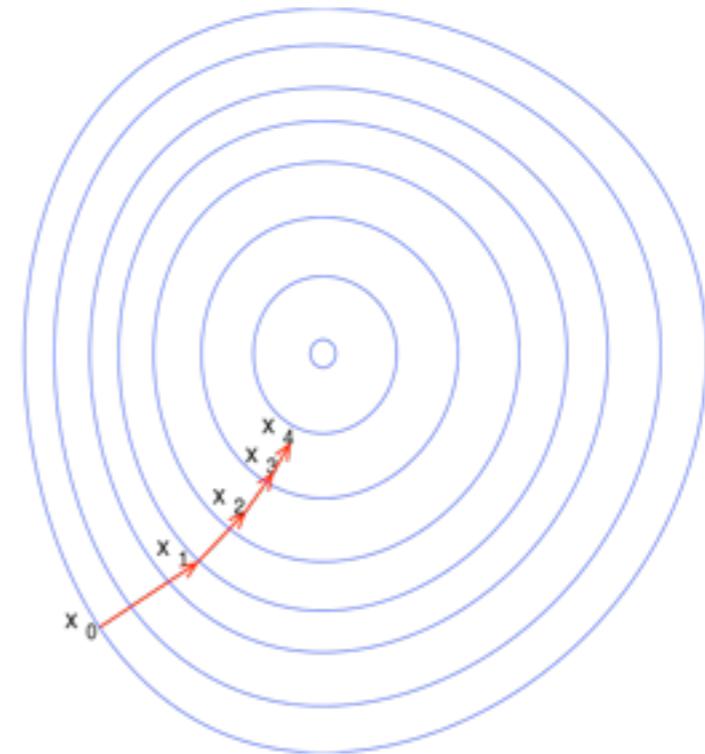
$$\vec{a}_{n+1} = \vec{a}_n - \lambda \nabla F(\vec{a}_n)$$

Método iterativo donde vamos siguiendo la dirección del gradiente hasta encontrar el mínimo. De tal manera que

$$F(a_0) > F(a_1) > F(a_2), \dots$$

λ puede ser constante o variar en cada iteración.

Es un método muy sencillo y falla en problemas complejos.



A medida que nos acercamos al mínimo puede tardar en converger.

TIPOS de ALGORITMOS de OPTIMIZACIÓN

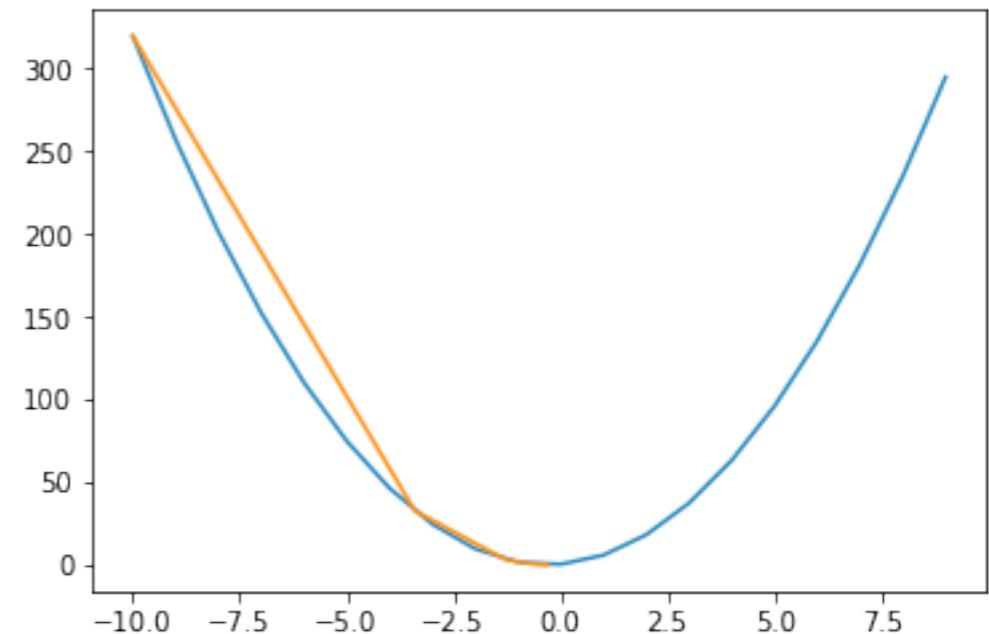
FAMILIA GRADIENTE :

Función objetivo tiene que ser derivable!

$$\vec{a}_{n+1} = \vec{a}_n - \lambda \nabla F(\vec{a}_n)$$

$$F(a_o) > F(a_1) > F(a_2), \dots$$

$$\lambda = 0.01$$



TIPOS de ALGORITMOS de OPTIMIZACIÓN

FAMILIA GRADIENTE :

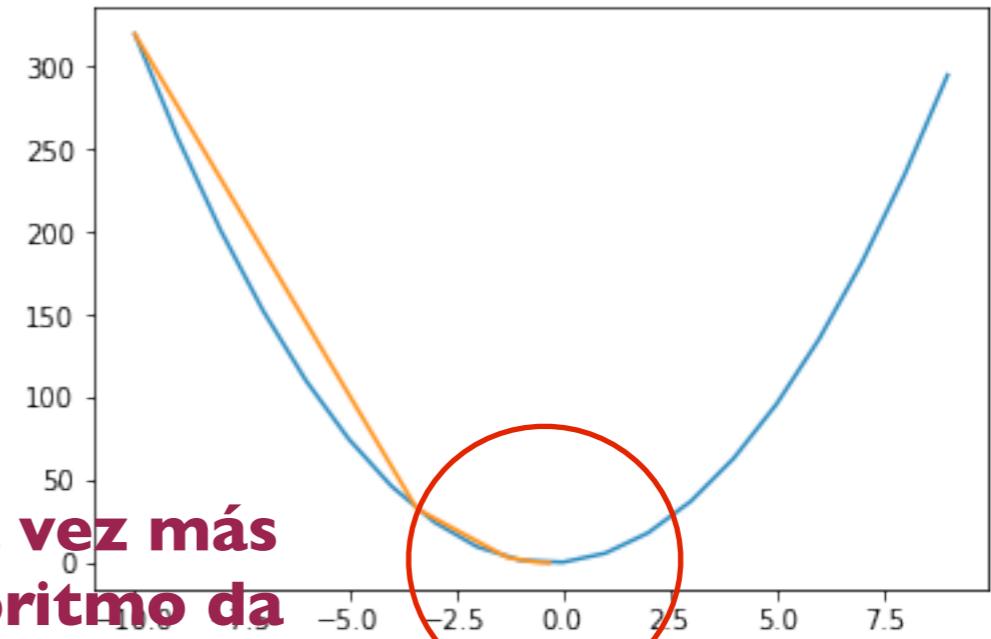
Función objetivo tiene que ser derivable!

$$\vec{a}_{n+1} = \vec{a}_n - \lambda \nabla F(\vec{a}_n)$$

$$F(a_o) > F(a_1) > F(a_2), \dots$$

$$\lambda = 0.01$$

gradiente cada vez más pequeño y algoritmo da pasos pequeños



TIPOS de ALGORITMOS de OPTIMIZACIÓN

FAMILIA GRADIENTE :

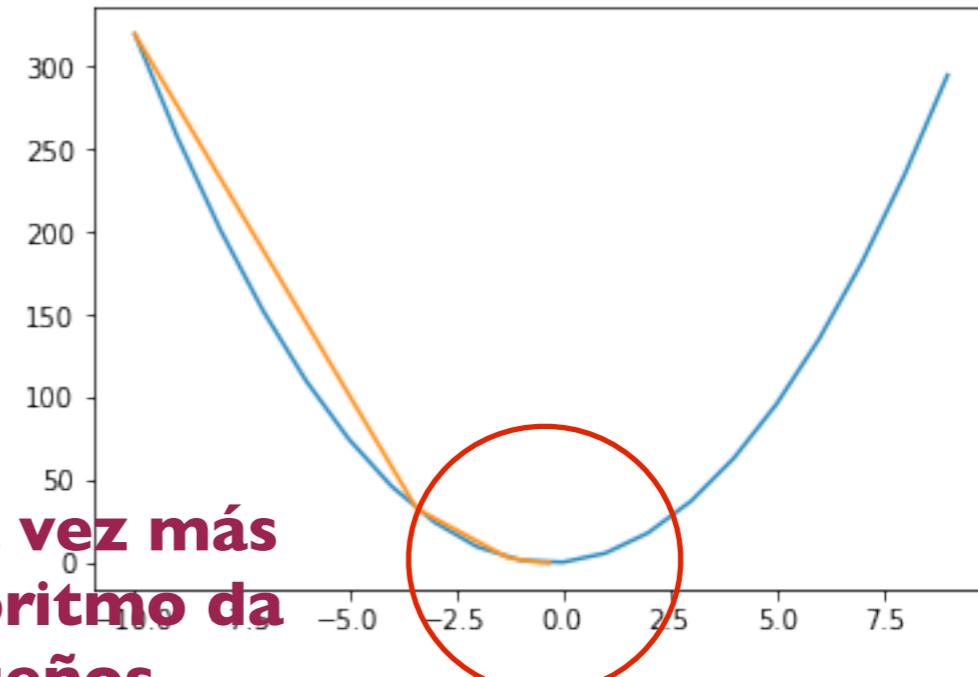
Función objetivo tiene que ser derivable!

$$\vec{a}_{n+1} = \vec{a}_n - \lambda \nabla F(\vec{a}_n)$$

$$F(a_o) > F(a_1) > F(a_2), \dots$$

$$\lambda = 0.01$$

gradiante cada vez más pequeño y algoritmo da pasos pequeños



Hay variaciones de este método que permiten mejorar la convergencia. Muchos de los otros métodos de esta familia, lo que hacen es buscar los ceros de la función gradiente. Y los buscan numéricamente (método de la secante, método de Newton,...), de los más eficientes son:

BFGS (Broyden–Fletcher–Goldfarb–Shanno) , (~70s) con varias variantes más modernas
LM (Levenberg–Marquardt method) (60s)

Dependen del valor inicial y suelen terminar cuando la variación de la función objetivo es menor a cierto valor de tolerancia.

TIPOS de ALGORITMOS de OPTIMIZACIÓN

FAMILIA BÚSQUEDAS DIRECTA :

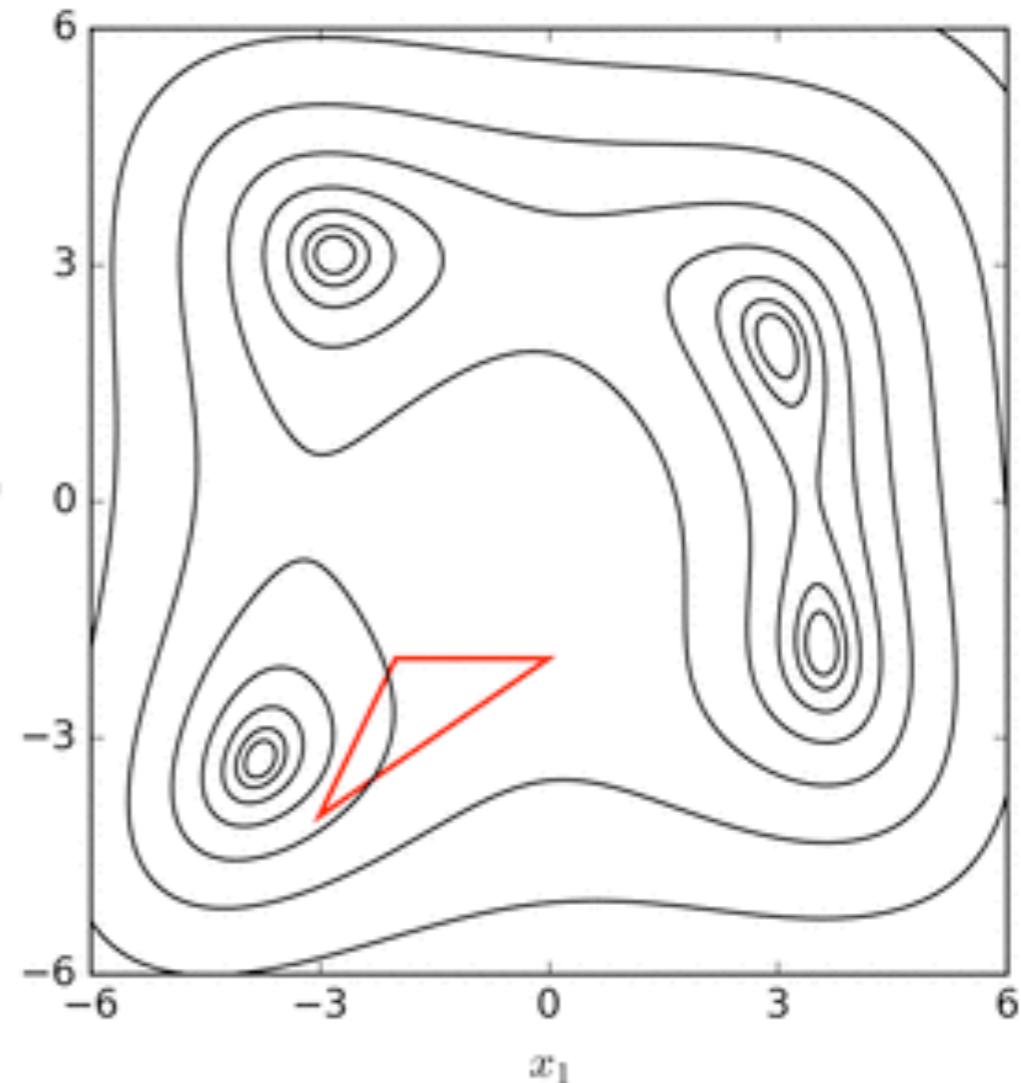
Nelder-Mead o downhill method (1965)

Método iterativo donde buscamos la dirección donde la función objetivo es menor.

En funciones derivables coincidirá con la dirección del gradiente.

Útil porque no necesita derivadas.

Depende del poliedro inicial (recta para una dimensión, triángulo para el plano, ...) y del criterio para parar.



Terminación algoritmo: Nelder-Mead proponen terminar cuando la desviación estándar de la función en los vértices del poliedro es menor a cierto valor de tolerancia previamente escogido.

TIPOS de ALGORITMOS de OPTIMIZACIÓN

- Necesitan un **punto inicial** donde empezar la búsqueda
- Criterio de convergencia, cuando parar
- Algunos requieren otras (gradiente, Hessian)



CARACTERÍSTICAS DE UN BUEN ALGORITMO:

- Tiene que converger al mínimo global
- Útil para distintos tipos de problemas
- Scalability --> (escalabilidad??) Los algoritmos tiene que ser lo más constantes en que se pueda. (Si añado más variables que no sea mucho más lento.)
- Memoria: que no necesiten mucha memoria .

TIPOS de PROBLEMAS de OPTIMIZACIÓN

dependiendo del tipo de problema irá mejor un algoritmo u otro

- **Continuos o discretos.** (Ej. cuando clasificamos estamos en un caso discreto.)
- **Con o sin ligaduras.** Algunos problemas se plantean con desigualdades, (ej. en una fábrica los recursos consumidos no pueden superar la cantidad disponible de recursos)
- **Univariable o multivariable.** (solución analítica en el caso de mutlivariables es más complicado)
- **Lineales o no-lineales.** Si las ecuaciones son lineales es más sencillo, y se necesitan algoritmos menos complejos
- **Determinista o estocástico.** Muchas veces tenemos errores en los datos por errores en la medida, por suposiciones imposibles de contrastar, distintos sujetos,... (Hay métodos que permiten añadir información del error)
- **Más de un objetivo.** A veces tenemos más de un objetivo (funciones a minimizar o maximizar) --> en general los reformulamos para que tengan una sola función objetivo)

Optimizar con Python

Hay varios paquetes en python para hacer optimización (scipy.optimize, PyOpt, cvxopt, Optunity..)

- En general los algoritmos para optimizar minimizan una función. Por lo que si tenemos un problema de maximización, lo reformulamos para que busque el mínimo.

maximiar $F(x)$ es lo mismo que minimizar $-F(x)$

Optimizar con Python

Hay varios paquetes en python para hacer optimización ([scipy.optimize](#),
PyOpt, cvxopt, Optunity..)

- En general los algoritmos para optimizar minimizan una función. Por lo que si tenemos un problema de maximización, lo reformulamos para que busque el mínimo.

maximiar $F(x)$ es lo mismo que minimizar $-F(x)$

Optimizar con Python

Hay varios paquetes en python para hacer optimización (**scipy.optimize**, PyOpt, cvxopt, Optunity..)

- En general los algoritmos para optimizar minimizan una función. Por lo que si tenemos un problema de maximización, lo reformulamos para que busque el mínimo.

maximiar $F(x)$ es lo mismo que minimizar $-F(x)$

- **scipy.optimize.fmin(fun, x0, args=(), xtol=0.0001, ...)**
usa un algoritmo clásico Nelder-Mead, que busca la dirección donde la función es menor (downhill method)

Optimizar con Python

Hay varios paquetes en python para hacer optimización (**scipy.optimize**,
PyOpt, cvxopt, Optunity..)

- En general los algoritmos para optimizar minimizan una función. Por lo que si tenemos un problema de maximización, lo reformulamos para que busque el mínimo.

maximiar $F(x)$ es lo mismo que minimizar $-F(x)$

- **scipy.optimize.fmin(fun, x0, args=(), xtol=0.0001, ...)**
usa un algoritmo clásico Nelder-Mead, que busca la dirección donde la función es menor (downhill method)
- **scipy.optimize.minimize(fun, x0, args=(), method=None, jac=None, hess=None, ...)**

Optimizar con Python

Hay varios paquetes en python para hacer optimización ([scipy.optimize](#),
PyOpt, cvxopt, Optunity..)

- En general los algoritmos para optimizar minimizan una función. Por lo que si tenemos un problema de maximización, lo reformulamos para que busque el mínimo.

maximiar $F(x)$ es lo mismo que minimizar $-F(x)$

- `scipy.optimize.fmin(fun, x0, args=(), xtol=0.0001,....)`

usa un algoritmo clásico Nelder-Mead, que busca la dirección donde la función es menor (downhill method)

Vemos ejemplo con Jupyter:
`intro.ipynb`

- `scipy.optimize.minimize(fun, x0, args=(), method=None, jac=None, hess=None, ...)`

OPTIMIZACIÓN LINEAL

(o Programación Lineal)

OPTIMIZACIÓN LINEAL

- Conjunto de problemas donde las funciones involucradas son lineales. En general una función a minimizar y una serie de desigualdades.

$$\text{minimiza} \quad z = -x_1 - 2x_2$$

$$\text{sujeto a} \quad -2x_1 + x_2 \leq 2$$

$$-x_1 + x_2 \leq 3$$

$$x_1 \leq 3$$

$$x_1, x_2 \geq 0$$

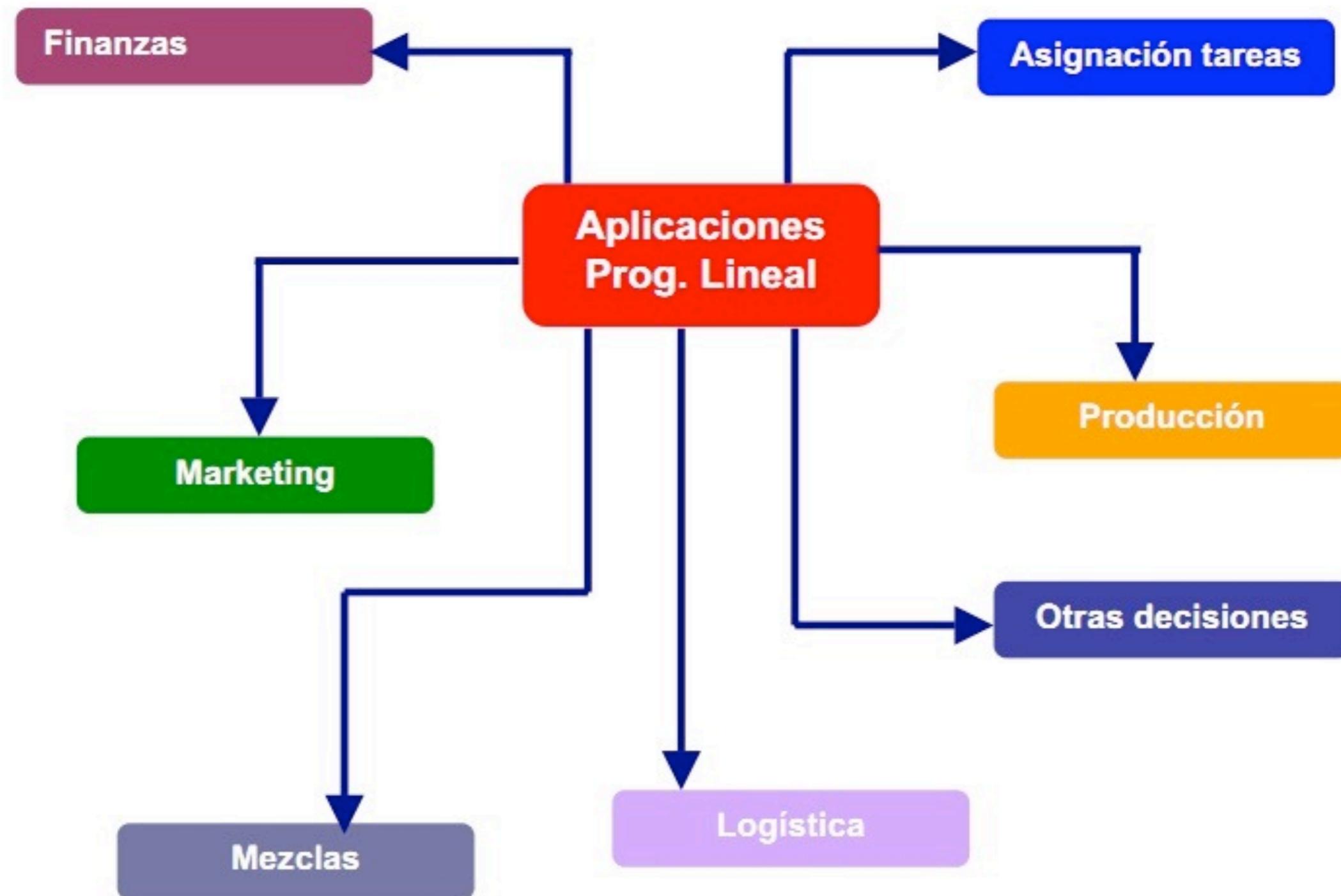
OPTIMIZACIÓN LINEAL

- Conjunto de problemas donde las funciones involucradas son lineales. En general una función a minimizar y una serie de desigualdades.

minimiza $z = -x_1 - 2x_2$ PERO...¿ALGUIEN USA
sujeto a $-2x_1 + x_2 \leq 2$ OPTIMIZACIÓN
 $-x_1 + x_2 \leq 3$ LINEAL??

$$x_1 \leq 3$$
$$x_1, x_2 \geq 0$$

OPTIMIZACIÓN



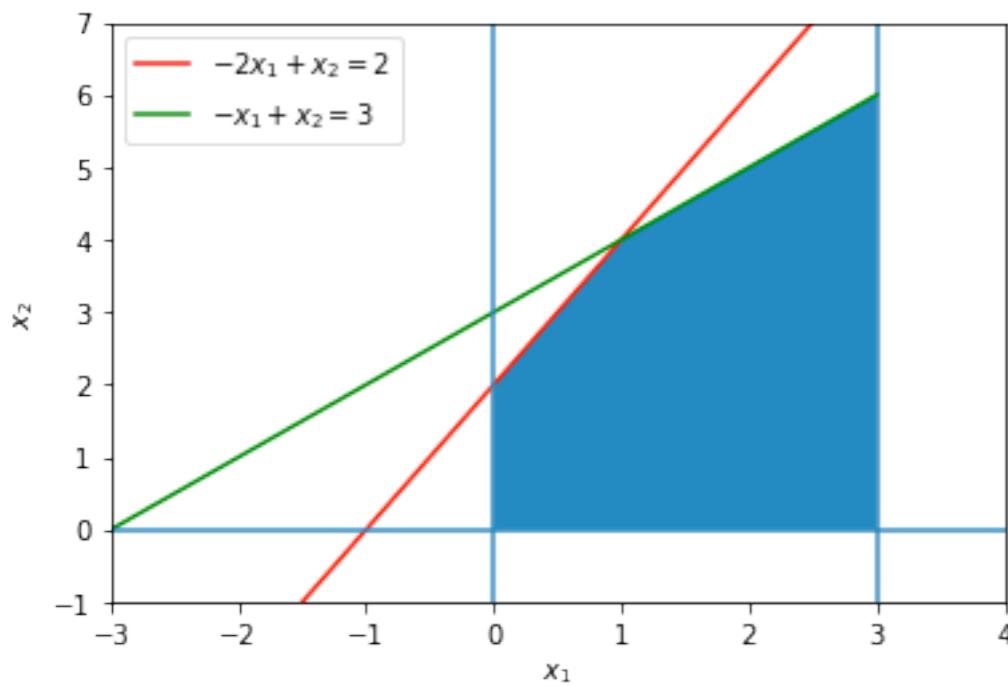
OPTIMIZACIÓN LINEAL

Análisis geométrico

minimiza $z = -x_1 - 2x_2$
sujeto a $-2x_1 + x_2 \leq 2$
 $-x_1 + x_2 \leq 3$
 $x_1 \leq 3$
 $x_1, x_2 \geq 0$

- Se hace un gráfico con las ecuaciones
- Se define la zona donde se cumplen todas las condiciones (espacio de soluciones factibles)
- Se calcula el valor de la función en distintos puntos dentro de ese espacio.

$$z_{min} = -15 \text{ en el punto } p=(3,6)$$



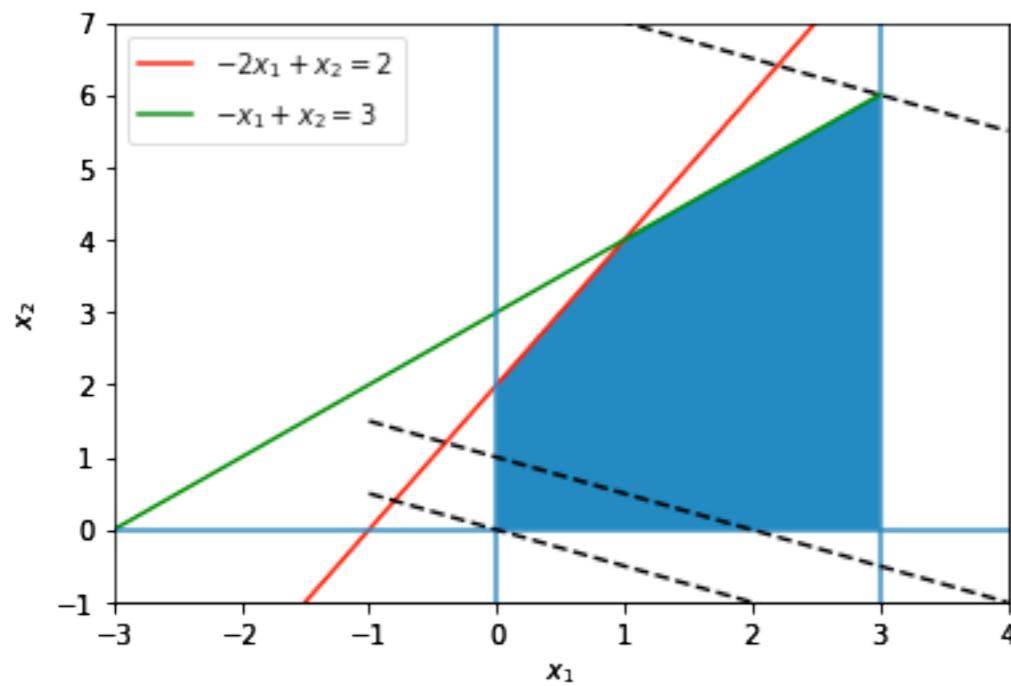
OPTIMIZACIÓN LINEAL

Análisis geométrico

minimiza $z = -x_1 - 2x_2$
sujeto a $-2x_1 + x_2 \leq 2$
 $-x_1 + x_2 \leq 3$
 $x_1 \leq 3$
 $x_1, x_2 \geq 0$

- Se hace un gráfico con las ecuaciones
- Se define la zona donde se cumplen todas las condiciones (espacio de soluciones factibles)
- Se calcula el valor de la función en distintos puntos dentro de ese espacio.

$$z_{min} = -15 \text{ en el punto } p=(3,6)$$



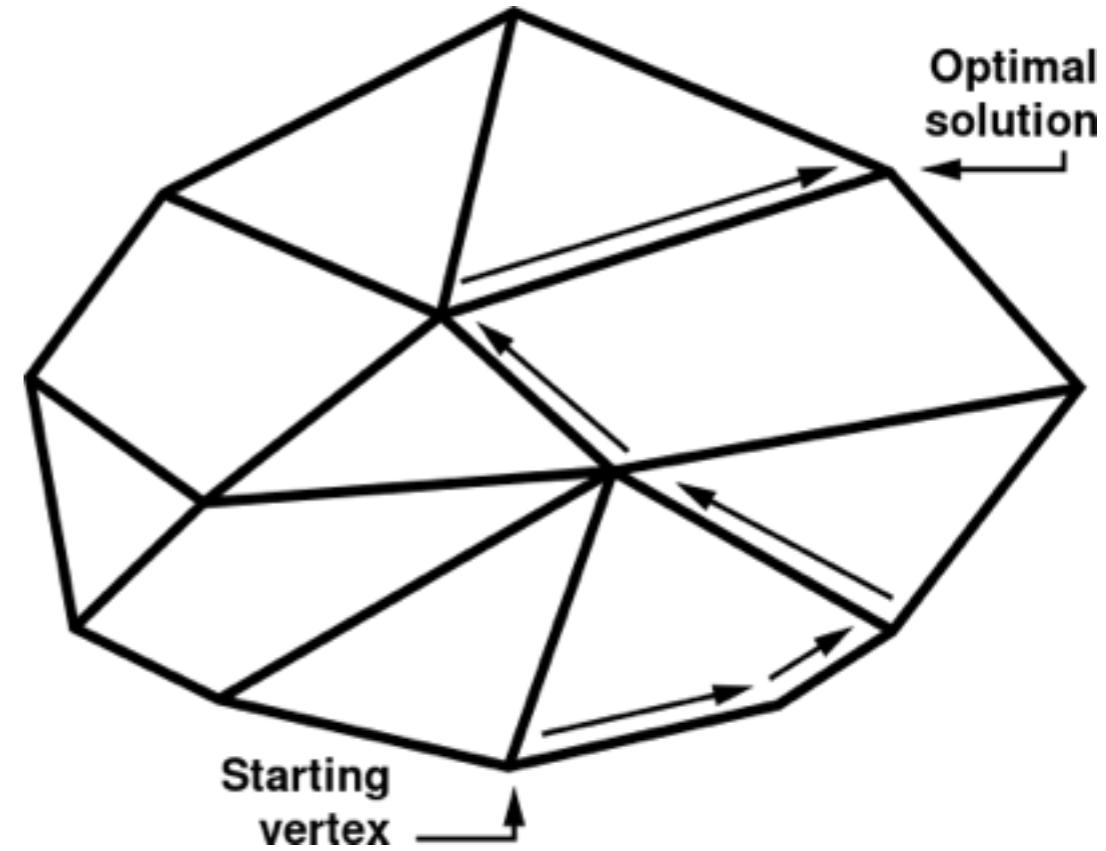
PROGRAMACIÓN LINEAL

SIMPLEX. (G. Dantzig en 1947)

En más parámetros, hacer un gráfico ya no es factible y Dantzig propuso un método iterativo para encontrar el mínimo de una función.

Si conseguimos reformular el problema inicial, de tal manera que quede:

$$\begin{array}{ll} \text{minimiza} & z = c^T x \\ \text{sujeto a} & Ax = b \\ & x \geq 0 \end{array}$$



De tal manera que el punto (o puntos) del espacio de parámetros que minimiza la función z está en uno de los vértices.

Recorremos los vértices para encontrar el punto donde la función es mínima. (Puede haber variantes de como hacer ese recorrido)

Optimizar con Python

- Para métodos con desigualdades, usamos el mismo paquete

```
scipy.optimize.minimize(fun, x0, args=(), method=None,  
jac=None, hess=None, ...)
```

pero es un poco más complicado, y depende del método que usemos tendrá una estructura u otra. Vemos ejemplo con Jupyter (LinealProgramming.ipynb) un ejemplo y pasáis a hacer vosotros ejercicios.

TIPOS de ALGORITMOS de OPTIMIZACIÓN

Para funciones derivables:

Usamos información del gradiente!

Levenberg-Marquardt (1944-1963)

Función objetivo mínimos cuadrados:

$$E = \sum_i^N \frac{(y_i - f(x_i))^2}{N}$$

siguiendo la misma idea de antes

$$f(x_i, \beta + \delta) \approx f(x_i, \beta) + J_i \delta$$

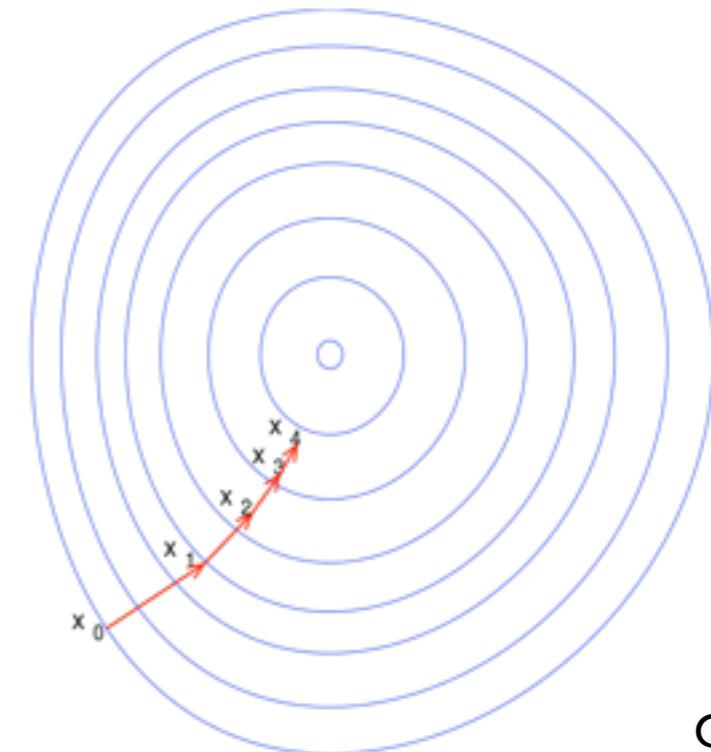
donde el vector J es el gradiente

$$J_i = \frac{df(x_i, \beta)}{d\beta}$$

$$S(\beta + \delta) \approx \sum_{i=1}^m [y_i - f(x_i, \beta) - J_i \delta]^2, \text{ derivando respecto a delta : } (\mathbf{J}^T \mathbf{J}) \delta = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\beta)],$$

$$[\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})] \delta = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\beta)].$$

Levenberg-
Marquardt

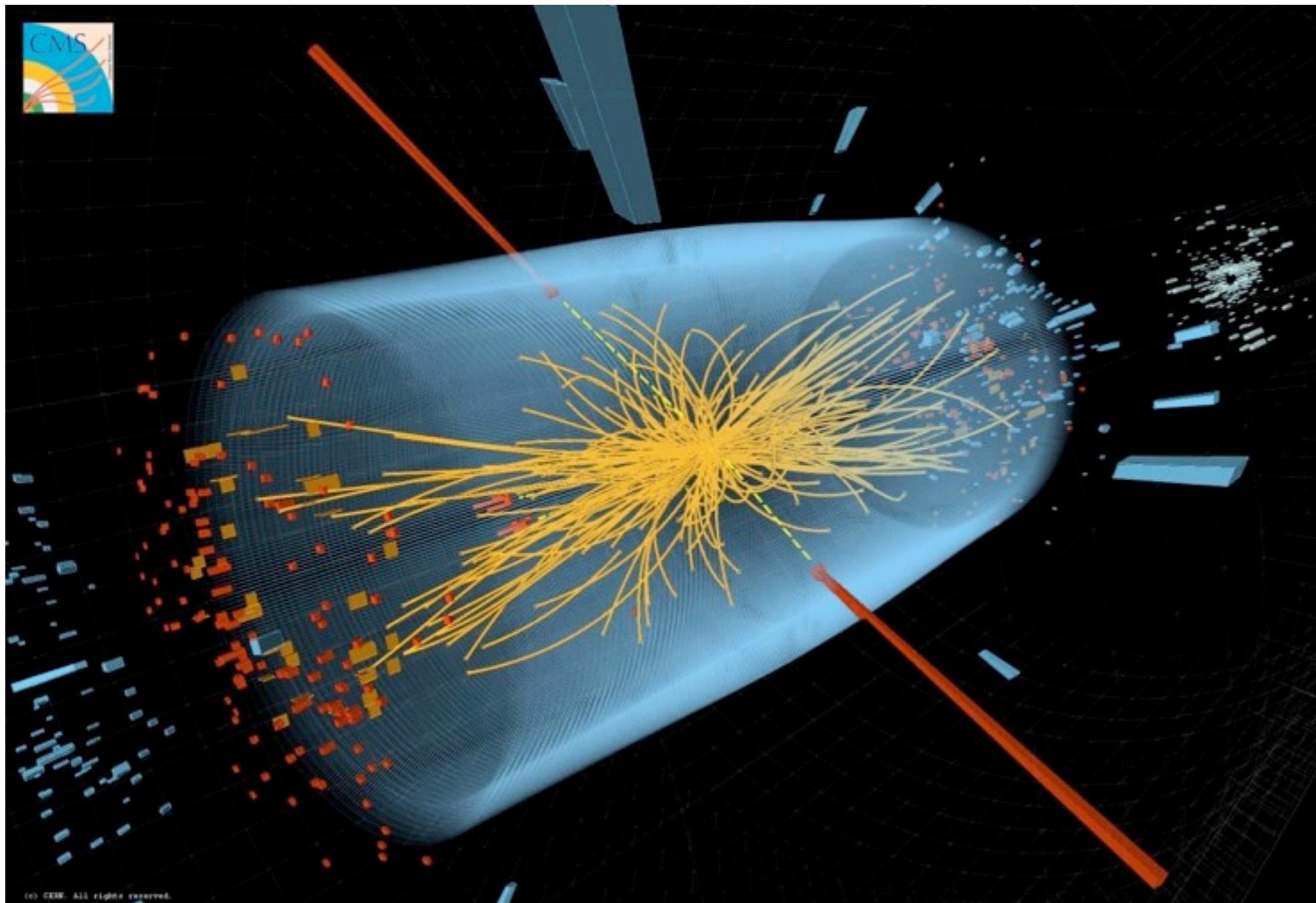


Gauss-Newton
method



Big Data.

Experimentos científicos: LHC

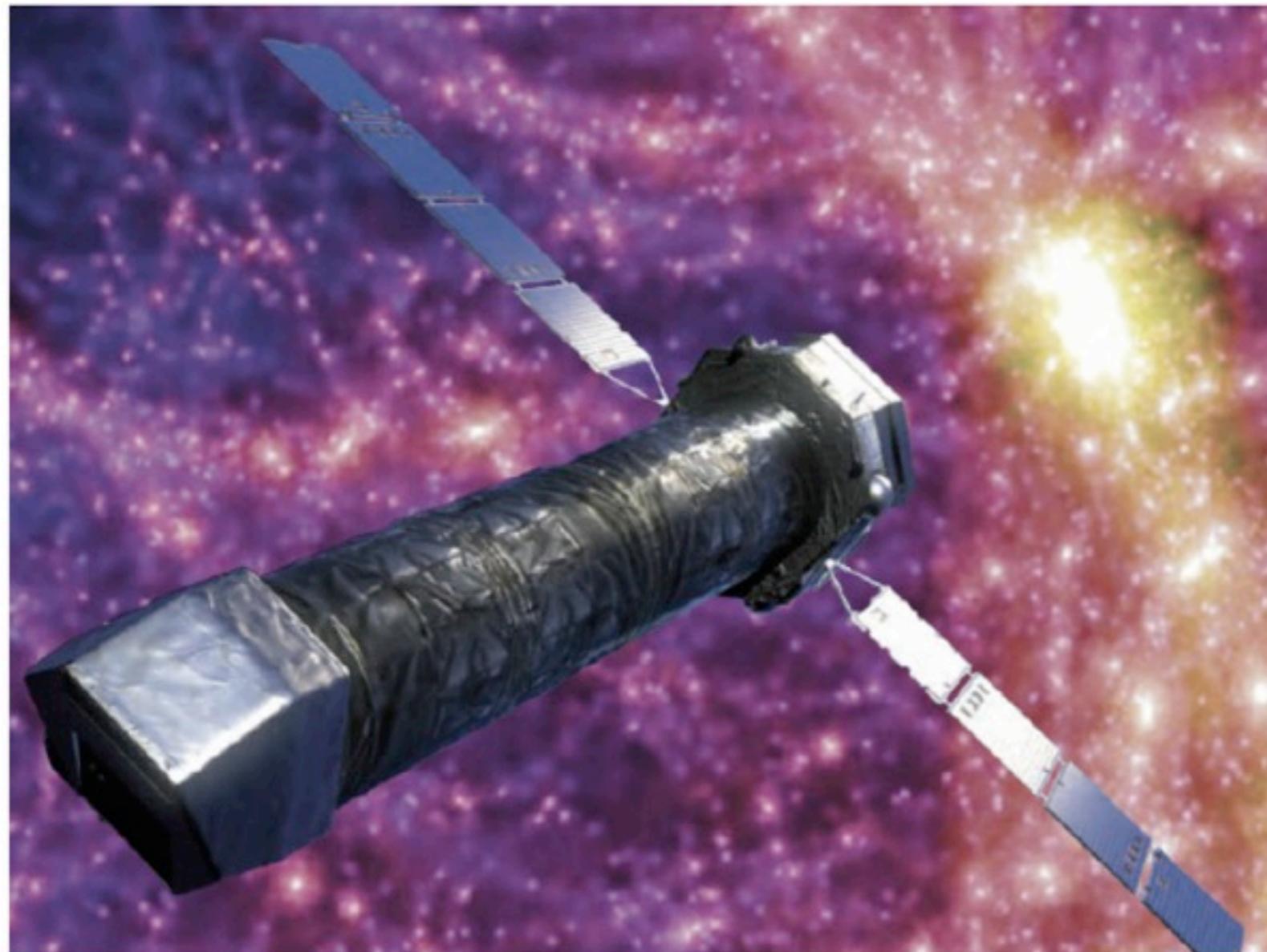


- 600TB/s
- Almacenan 25GB/s

Big Data.

Experimentos científicos:

Athena X Ray Observatory



- más de 500,000 galaxias
- Imposible tratarlas una a una
- Importante clasificarlas con algoritmos rápidos
- Analizarlas (sacar parámetros de algun modelo) con métodos rápidos

Big Data.

Redes sociales:

- Según el informe 2018 Global Digital (<https://wearesocial.com>) más 4.0 millones de personas usan internet habitualmente (de 7 mil millones de habitantes!!) y más de 3 mil millones son usuarios una red social.

- La cantidad de datos que manejan es increíble.



-Se pueden hacer muchísimos estudios con esos datos, obtener modelos para hacer predicciones, clasificar usuarios, encontrar patrones.

-Importante políticas de protección de datos.

Big Data.

Tarjetas, móviles:

- Las grandes empresas recogen muchísimos datos al día que tienen que gestionar.

Necesitan algoritmos para sacar el máximo provecho de esos datos

- Nosotros mismo guardamos muchos más datos que hace pocos años.

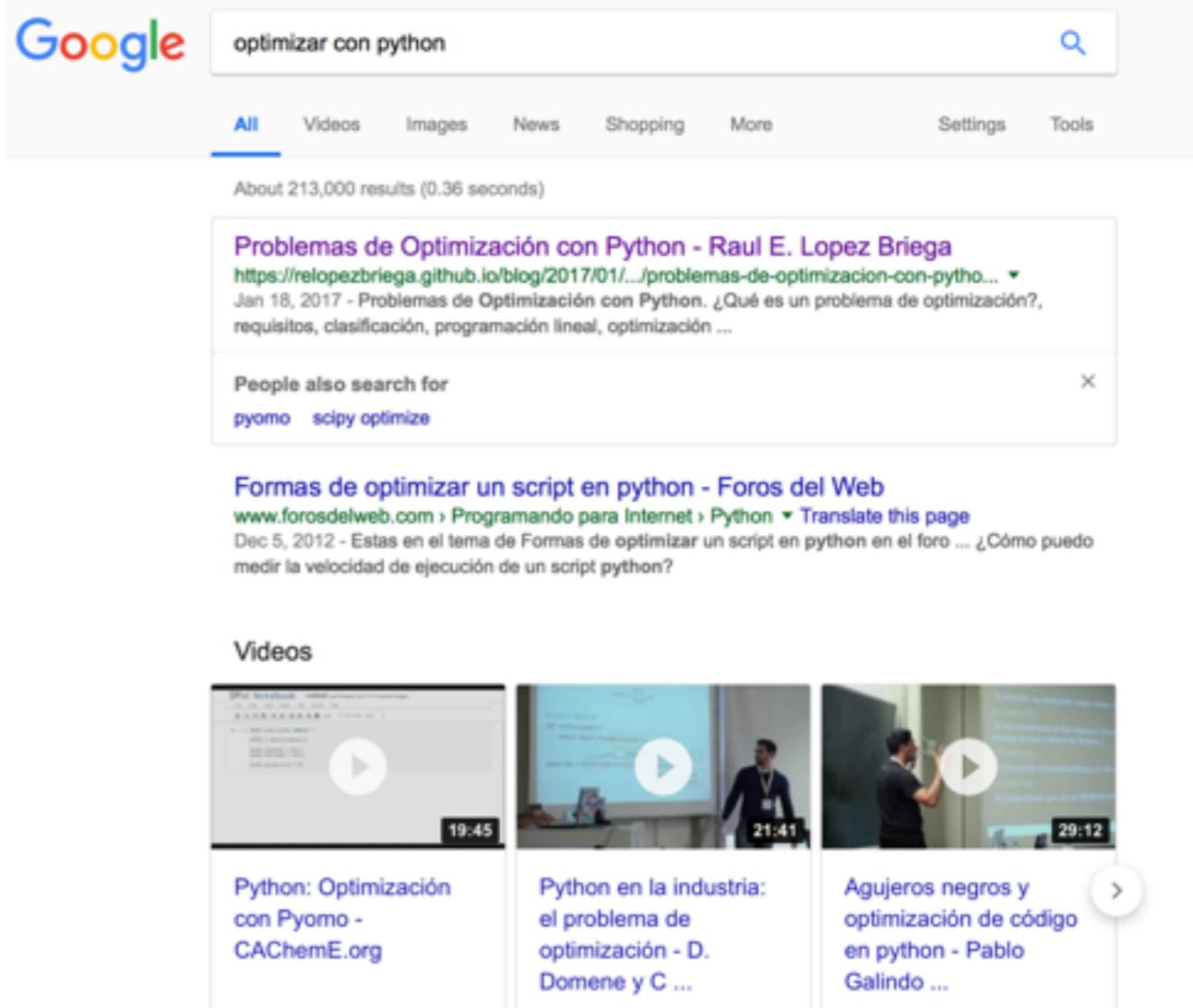
- Los datos son muy valiosos, no tanto individualmente, que también, para obtener modelos predictivos.



TIPOS de PROBLEMAS de OPTIMIZACIÓN

Ej. Internet y redes sociales

- **Publicidad.** Maximizar la probabilidad de que vas a “clickar” en un anuncio , o de que vas a comprar algo,...
- **Buscadores.** Maximizar el grado de satisfacción del usuario con los resultados de una búsqueda



A screenshot of a Google search results page. The search query "optimizar con python" is entered in the search bar. The results are filtered under the "All" tab. The top result is a blog post titled "Problemas de Optimización con Python - Raul E. Lopez Briega" from "relopezrieg.github.io". Below it, there's a section titled "People also search for" with terms like "pyomo", "scipy", and "optimize". Further down, there's a section titled "Formas de optimizar un script en python - Foros del Web" with a link to "www.forosdelweb.com". At the bottom, there's a "Videos" section showing three video thumbnails: "Python: Optimización con Pyomo - CACHEM.E.org", "Python en la industria: el problema de optimización - D. Domene y C ...", and "Agujeros negros y optimización de código en python - Pablo Galindo ...".