Métodos III - Derivadas

February 18, 2019

7 Mínimos y máximos de funciones escalares

Jose A. Hernando
Departamento de Física de Partículas. Universidade de Santiago de Compostela
Enero 2019

```
In [1]: import time
        print(' Last version ', time.asctime() )
```

```
 Last version  Sun Feb 17 23:59:06 2019
```

7.1 Objectivos

Recordar el concepto de mínimo y máximo de una función real y cómo se calculan.
Introducir el desarrollo de Taylor de 2º orden de funciones escalares y la matriz hessiana.

- Comentar la condición suficiente.

Relación entre los mínimos y máximos y los autovalores de la matriz hessiana.

- Ejemplos de cálculos de mínimos y máximos.

Introducir el concepto de mínimos condicionados.

- Presentar el método de los multiplicadores de Lagrange.

```
In [2]: # general imports
        %matplotlib inline
        %reload_ext autoreload
        %autoreload 2

        # numpy and matplotlib
        import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        matplotlib.style.use('ggplot')
        import graph_utils as gf

        figsize = 6, 3.8
        cmap    = 'hot'
```

7.2 Desarrollo de Taylor de 2º orden de funciones escalares.

Revisión de los conceptos de extremos, mínimos y máximos. Si recuerdas, para las funciones reales, $f(x)$, habíamos definido como mínimo local, aquel valor x_0 del dominio tal que en los valores en un intervalo en torno suyo, de anchura h, la función es mayor o igual que en x_0. De forma similar definimos el máximo local.

Llamamos mínimo local:

$$x \in [x_0 + h, x_0 + h] \Rightarrow f(x) \geq f(x_0)$$

Llamamos máximo local:

$$x \in [x_0 + h, x_0 + h] \Rightarrow f(x) \leq f(x_0)$$

Dijimos que x_0 es un mínimo global si todos el valor de la función en todos los puntos del dominio es mayor que en x_0, y de forma similar para el máximo.

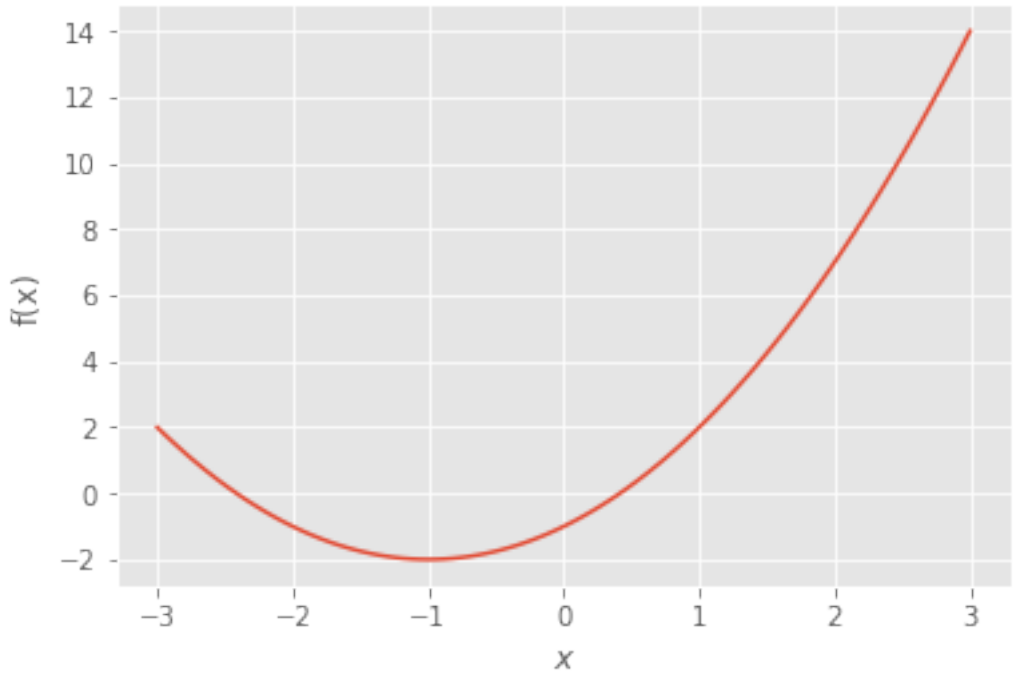
Ejemplo: La siguiente función real presenta un mínimo local en el $x_0 = -1$.

¿Recuerdas cómo calculábamos los mínimos o máximos? Ambos cumplían la condición de que su pendiente era nula y el signo de segunda derivada nos indicaba si era positiva que era un mínimo, y si era negativa, que era un máximo.

Condición de extremos: $f'(x_0) = 0$

Codición de máximo: $f''(x_0) < 0$ y de mínimo $f''(x_0) > 0$

```
In [3]: fun = lambda x: x**2 + 2*x - 1
        gf.fun1d(fun);
```



En este caso, con $f(x) = x^2 + 2x - 1$, la condición de extremo: $f'(x_0) = 2x_0 + 2 = 0$, que resulta en $x_0 = -1$.

Se trata de un mínimo, ya que $f''(x_0) = 2$

Si recuerdas, podíamos obtener los mínimos y máximos, gracias a que esta función es diferenciable, admite desarrollo de Taylor de segundo orden.

El desarrollo de Taylor de segundo orden de una función real, $f(x)$, es:

$$f(x_0 + h) \simeq f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2$$

donde h es un tamaño 'pequeño'.

Recuerda también que la condición suficiente para que una función real tenga desarrollo de Taylor de segundo orden en un valor x_0 es que su derivada segunda en un entorno de ese valor sea continua.

Interpretando $f'(x_0)$ como la pendiente, vemos que para que la función no cambie de valor entorno a x_0, su pendiente tiene que ser nula.

Y después, el segundo sumando, $\frac{1}{2}f''(x_0)h^2$, es positivo o negativo dependiendo del signo de la derivada segunda, $f''(x_0)$. De ahí la condición de mínimo o máximo.

Igualdad de las derivadas segundas cruzadas Antes de abordar el desarrollo de Taylor de segundo orden de funciones escalares, vamos a ver este teorema importante:

Teorema: Sea $f(\mathbf{x})$ una función escalar de $\mathrm{R}^n \to \mathrm{R}$, si las derivadas segundas son continuas en una bola centrada en un punto, $\mathbf{x}_0$, se cumple que las derivadas cruzadas son iguales:

$$\frac{\partial^2 f(\mathbf{x}_0)}{\partial x_i \partial x_j} = \frac{\partial^2 f(\mathbf{x}_0)}{\partial x_j \partial x_i}$$

con $i \neq j,\ i = 1, \ldots, n;\ j = 1, \ldots, n$.

7.3 Desarrollo de Taylor de segundo orden de funciones escalares

Sea una función escalar, $f(\mathbf{x})$, de $\mathrm{R}^n \to \mathrm{R}$, con derivadas segundas continuas en un punto interior, $\mathbf{x}_0$, del dominio, se puede aproximar en un punto 'próximo', $\mathbf{x}$, separado de $\mathbf{x}_0$ por un vector 'pequeño' $\mathbf{v}$, por su **desarrollo de Taylor de segundo orden**:

$$f(\mathbf{x}_0 + \mathbf{v}) \simeq f(\mathbf{x}_0) + \sum_{i=1}^{n} \frac{\partial f(\mathbf{x}_0)}{\partial x_i} v_i + \sum_{i,j=1}^{n} \frac{\partial^2 f(\mathbf{x}_0)}{\partial x_i \partial x_j} v_i v_j$$

Que podemos reescribir en forma matricial:

$$f(\mathbf{x}_0 + \mathbf{v}) \simeq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)\, \mathbf{v} + \frac{1}{2}\mathbf{v}^T\, \mathbf{H}(\mathbf{x}_0)\, \mathbf{v}$$

donde $\mathbf{H}(\mathbf{x}_0)$ es la **matriz hessiana** valorada en $\mathbf{x}_0$, cuyas componentes son las derivadas segundas,

$$\mathbf{H}_{ij}(\mathbf{x}_0) = \frac{\partial^2 f(\mathbf{x}_0)}{\partial x_i \partial x_j}$$

y donde $\mathbf{v}$, es el vector en forma columna, y $\mathbf{v}^T$ en forma fila.

Cuestión: ¿Es la hessiana una matriz simétrica?

Cuestión: Se escapa del alcance de este curso, pero inténtalo, ¿cómo crees que sería el desarrollo de Taylor de segundo orden de una función vectorial?

Date cuenta: Al menos hay una función vectorial de la que conoces su desarrollo de Taylor de 2º orden. ¡La de la trayectoria de un movil a lo largo del tiempo!

$$\mathbf{r}(t_0 + \Delta t) \simeq \mathbf{r}(t_0) + \mathbf{r}'(t_0)\Delta t + \frac{1}{2}\mathbf{r}''(t_0)(\Delta t)^2$$

Ejemplo: Calcula el desarrollo de Taylor de la función escalar, $f(x, y) = 2x^2 + 4y^2$, en el punto $(0, 0)$.

Los distintos sumandos del desarrollo de Taylor son:

$$f(\mathbf{x}_0) = 0$$

El gradiente es

$$\nabla f(\mathbf{x}) = (4x, 8y)$$

lo que nos da:

$$\nabla f(\mathbf{x}_0) = (0, 0)$$

No hay pendientes en ese punto.

La matriz hessiana es:

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 4 & 0 \\ 0 & 8 \end{pmatrix}$$

que es independiente del punto, $\mathbf{x}$.

Luego el desarrollo de Taylor respecto al punto $\mathbf{x}_0$ es:

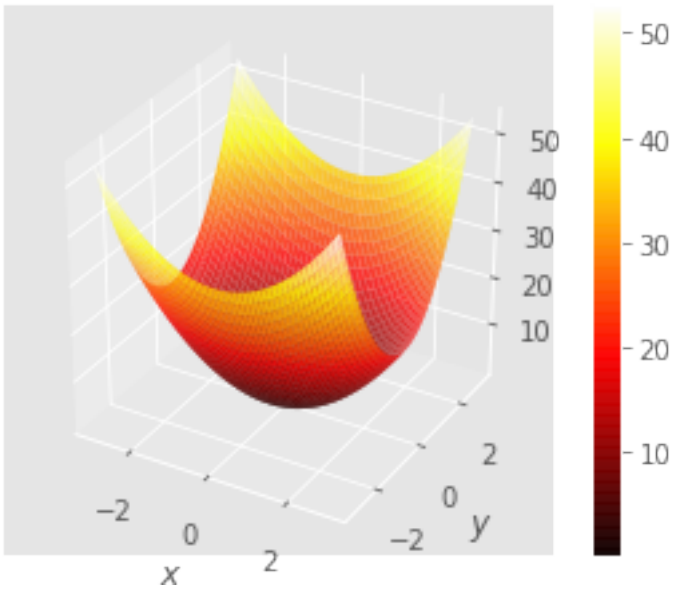
$$f(\mathbf{x}_0 + \mathbf{v}) = \frac{1}{2}(v_x, v_y) \begin{pmatrix} 4 & 0 \\ 0 & 8 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = 2v_x^2 + 4v_y^2$$

Observa la gráfica de la función y sus conjuntos de nivel.

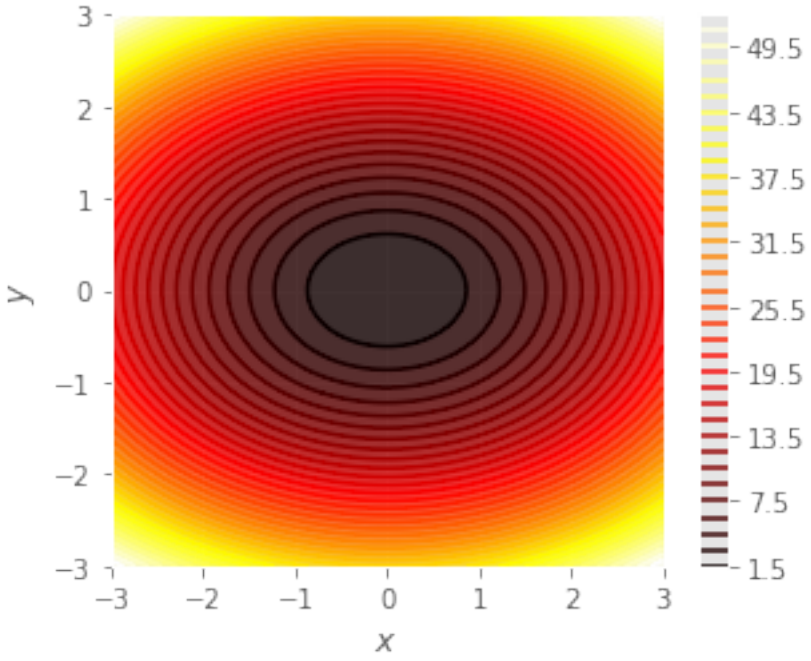
Cuestión: ¿Puedes determinar si hay un mínimo o un máximo local y dónde?

Es obvio, ¿verdad?

```
In [4]: fun = lambda x, y : 2*x*x + 4*y*y
        gf.graph(fun);
```




```
In [5]: gf.contour(fun, contours = 40);
```



7.4 Mínimos y máximos de funciones escalares

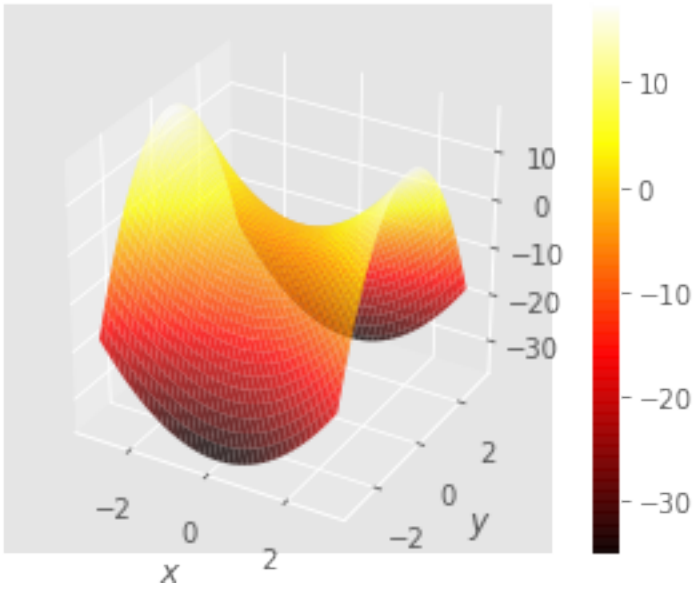
Como en una dimensión, llamaremos un punto **extremo** o **crítico**, $\mathbf{x}_0$, de una función escalar diferenciable, $f(\mathbf{x})$, de $\mathrm{R}^n \to \mathrm{R}$, a aquellos puntos del dominio cuyo gradiente sea nulo, $\nabla f(\mathbf{x}_0) = \mathbf{0}$.

En el ejemplo anterior, la función escalar, $f(x,y) = 2x^2 + 4y^2$ tiene un extremo en $(0,0)$.

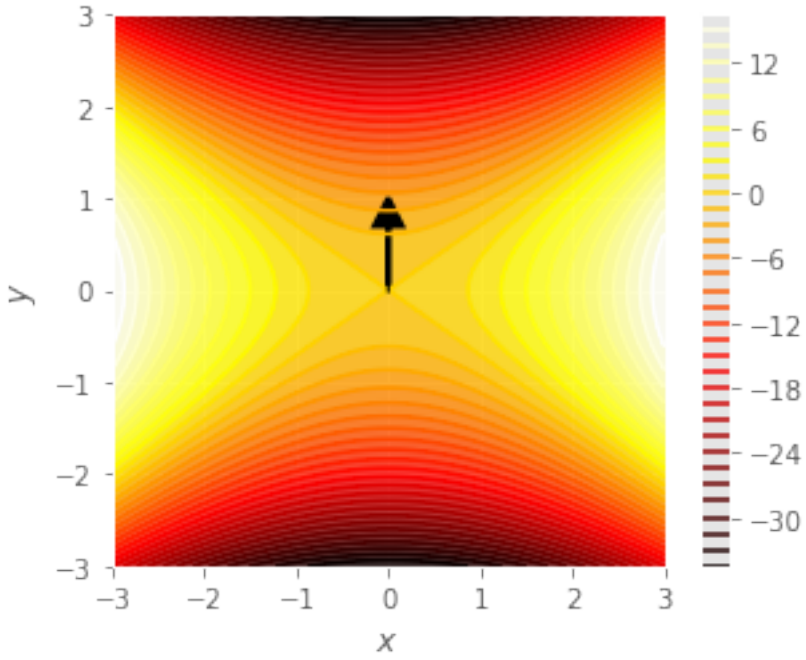
Observa la gráfica y los conjuntos de nivel de la siguiente función, $f(x,y) = 2x^2 - 4y^2$. Puedes comprobar que el punto $(0,0)$ es un extremo, tiene gradiente nulo (el gradiente es $\nabla f(x,y) = (4x, -8y)$), sin embargo no es ni máximo ni mínimo, es un **punto silla**. Definimos como tal, al punto extremo a partir del que la función asciende en una dirección y desciende en otra.

Explora y modifica el vector **v** sobreimpuesto sobre los conjuntos de nivel y observa como dependiendo de la dirección de **v** la función finalmente asciende o desciende.

```
In [6]: fun = lambda x, y: 2*x*x - 4*y*y
        gf.graph(fun);
```




```
In [7]: x0, y0 =  0., 0.
        vx, vy =  0., 1.
        gf.contour(fun, contours = 40);
        gf.arrow(x0, y0, vx, vy);
```



Ejercicio: Da el desarrolo de Taylor de segundo orden de la función $f(x,y) = 2x^2 - 4y^2$ en el punto silla $(0,0)$.

La función en $\mathbf{x}_0$, vale:

$$f(\mathbf{x}_0) = 0$$

El gradiente es nulo:

$$\nabla f(x,y) = (4x, -8y) \Rightarrow \nabla f(\mathbf{x}_0) = (0,0)$$

La matriz hessiana es:

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 4 & 0 \\ 0 & -8 \end{pmatrix}$$

que es independiente del punto, $\mathbf{x}$.

Luego el desarrollo de Taylor respecto al punto $\mathbf{x}_0$ es:

$$f(\mathbf{x}_0 + \mathbf{v}) = \frac{1}{2}(v_x, v_y) \begin{pmatrix} 4 & 0 \\ 0 & -8 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = 2v_x^2 - 4v_y^2$$

Donde vemos que si avanzamos *solo* en la dirección x la función asciende, mientras que si avanzamos *solo* en la dirección y, desciende. Se trata pues de un punto silla.

Clasificación de puntos extremos Los puntos extremos de una función escalar, serán máximos, mínimos o puntos silla, dependiendo de valor del término de la matriz hessiana de su desarrollo de Taylor:

$$\frac{1}{2}\mathbf{v}^T \mathbf{H}(\mathbf{x}_0)\, \mathbf{v}$$

Si para todo vector $\mathbf{v}$ el término es positivo, entonces es un mínimo, la función asciende siempre. Si es negativo, es un máximo, la función desciende siempre. Y si dependiendo del vector, es positivo o negativo, es un punto silla, la función en una dirección asciende y en otra desciende.

Consideremos el caso de una función escalar $f(x, y)$ de $\mathrm{R}^2 \to \mathrm{R}$, en el que la matriz Hessiana es diagonal, y los elementos de la diagonal son λ_1, λ_2 en un punto extremo $\mathbf{x_0}$.

El término del Hessiano en el desarrollo de Taylor sería:

$$\frac{1}{2}(v_x, v_y) \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \frac{1}{2}\left(\lambda_1 v_x^2 + \lambda_2 v_y^2\right)$$

Podemos ver que si los dos valores son positivos, $\lambda_1 > 0, \lambda_2 > 0$, el punto será un mínimo, para cualquier vector (v_x, v_y) la función asciende.

Si los dos son negativos, $\lambda_1 < 0, \lambda_2 < 0$, es un máximo, la función desciende siempre.

Si uno es negativo y otro positivo, dependera del vector, (v_x, v_y), en determinadas direcciones la función asciende y otras denciende, luego será un punto silla.

Ejercicio: Calcula la matriz hessiana para el punto extremo $(0, 0)$ de la función, $f(x, y) = 3x^2 + 3y^2 - 2xy$.

Observa la gráfica de esta función y sus conjuntos de nivel.

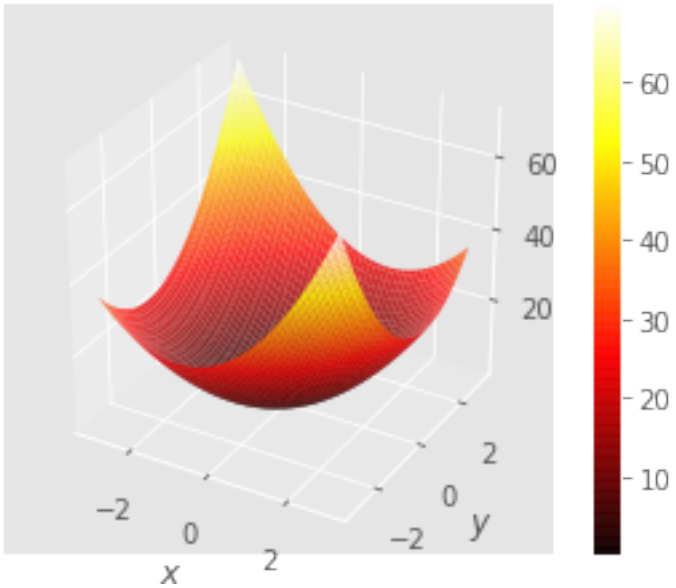
El gradiente es:

$$\nabla f(x, y) = (6x - 2y, 6y - 2x)$$

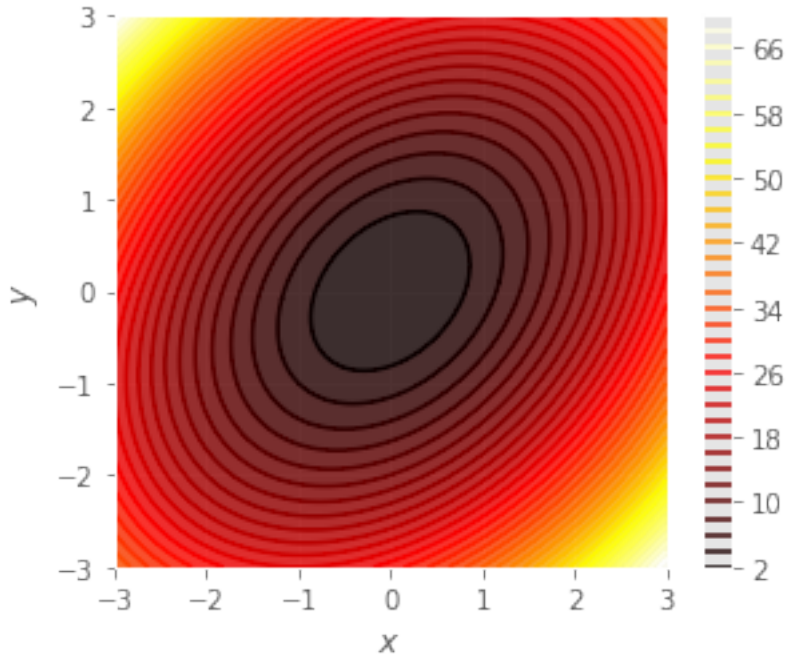
Y la matriz Hessiana:

$$\mathbf{H}(x, y) = \begin{pmatrix} 6 & -2 \\ -2 & 6 \end{pmatrix}$$

```
In [8]: fun = lambda x, y: 3*x*x + 3*y*y -2*x*y
        gf.graph(fun);
```



```
In [9]: gf.contour(fun, contours=40);
```



¿Y si la hesiana en el punto extremo, $\mathbf{x}_0$, no es diagonal?

Sabemos que la matriz hessiana, al ser las derivadas segundas continuas, es simétrica, y también sabemos que las matrices simétricas son diagonalizables.

Por lo tanto, existe una base ortonormal, $\{\mathbf{e}'_1, \mathbf{e}'_2\}$, en la que la matriz hessiana es diagonal, $\mathbf{H}'(\mathbf{x}_0)$ con auto-valores λ'_1, λ'_2. La relación entre las dos viene dada por la matriz ortonormal de cambio de base $\mathbf{U}$.

$$\mathbf{H}(\mathbf{x}_0) = \mathbf{U}\,\mathbf{H}'(\mathbf{x}_0)\,\mathbf{U}^T$$

donde:

$$\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$$

El vector $\mathbf{v}$ lo podemos expresar en la base de los autovectores (lo denotamos como $\mathbf{v}'$), mediante el cambio:

$$\mathbf{v} = \mathbf{U}\,\mathbf{v}'$$

Podemos entonces reescribir el término de la hessiana del desarrollo de Taylor en la base de los autovectores:

$$\frac{1}{2}\mathbf{v}^T\,\mathbf{H}(\mathbf{x}_0)\,\mathbf{v} = \frac{1}{2}\mathbf{v}'^T\mathbf{U}^T\,\mathbf{U}\mathbf{H}'(\mathbf{x}_0)\mathbf{U}^T\,\mathbf{U}\mathbf{v}' = \frac{1}{2}\mathbf{v}'\,\mathbf{H}'(\mathbf{x}_0)\,\mathbf{v}'$$

Esto es, el término de la hessiana del desarrollo de Taylor no depende de la base en la que lo calculemos.

Si expresamos el vector, (v_x, v_y), en la base de los autovectores, esto es, (v'_x, v'_y), tenemos

$$\frac{1}{2}\mathbf{v}'\,\mathbf{H}'(\mathbf{x}_0)\,\mathbf{v}'^T = \frac{1}{2}(v'_x, v'_y)\begin{pmatrix} \lambda'_1 & 0 \\ 0 & \lambda'_2 \end{pmatrix}\begin{pmatrix} v'_x \\ v'_y \end{pmatrix} = \frac{1}{2}\left(\lambda'_1 v'^2_x + \lambda'_2 v'^2_y\right)$$

Y por lo tanto, si los dos autovectores de la Hessiana son positivos, $\lambda'_1 > 0, \lambda'_2 > 0$, el punto extremo $\mathbf{x}_0$ es mínimo; si los dos son negativos $\lambda'_1 < 0, \lambda'_2 < 0$, es un máximo, y si uno es positivo y otro negativo, es un punto silla.

En el ejemplo anterior, la Hessiana era:

$$\mathbf{H}(x, y) = \begin{pmatrix} 6 & -2 \\ -2 & 6 \end{pmatrix}$$

Calculamos sus auto-valores:

$$\begin{vmatrix} (6-\lambda) & -2 \\ -2 & (6-\lambda) \end{vmatrix} = (6-\lambda)^2 - 4 = 0$$

Esto es:

$$(6-\lambda) = \pm 2 \;\Rightarrow\; \lambda = 6 \pm 2 \;\Rightarrow\; \lambda'_1 = 4,\; \lambda'_2 = 8$$

El punto extremo es mínimo porque los dos autovalores son positivos.

Calculemos sus autovectores $\mathbf{e}'_1 = (u_x, u_y),\ \mathbf{e}'_2 = (v_x, v_y)$, que recordemos deben ser ortonormales:

$$\begin{pmatrix} 6 & -2 \\ -2 & 6 \end{pmatrix}\begin{pmatrix} u_x \\ u_y \end{pmatrix} = 4\begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

Esto es:

$$6u_x - 2u_y = 4u_x \;\Rightarrow 2u_x = 2u_y \;\Rightarrow u_x = u_y$$

Dado que el vector tiene normal unidad:

$$\mathbf{e}'_1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

Mientras que para el segudo autovector:

$$\begin{pmatrix} 6 & -2 \\ -2 & 6 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \end{pmatrix} = 8\begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

Esto es:

$$6v_x - 2v_y = 8v_x \;\Rightarrow 2v_y = -2v_y \;\Rightarrow\; v_x = -v_y$$

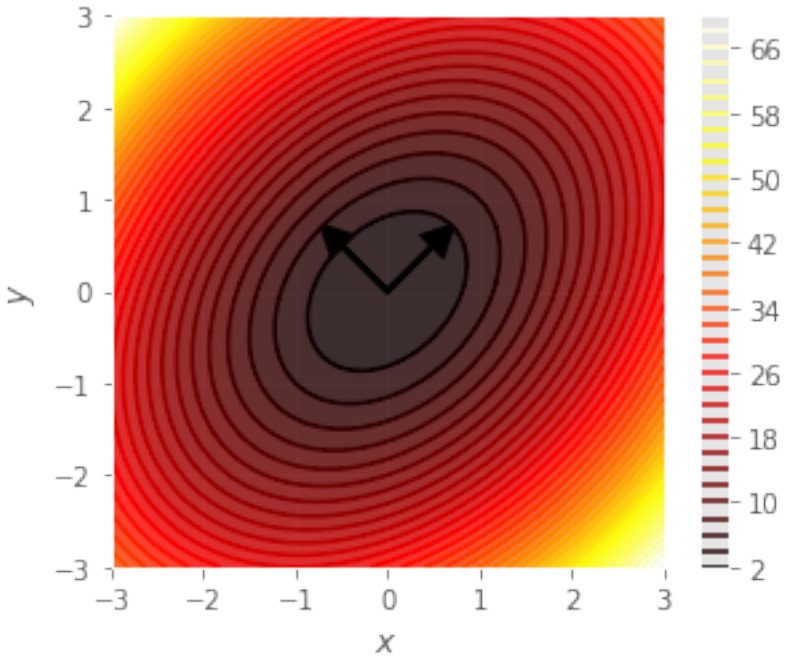
Como tienen que ser ortonormales:

$$\mathbf{e}'_2 = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$$

En la siguiente celda dibujamos los conjuntos de nivel de la función y sobre el punto extremo, dibujamos los autovectores.

Cuestión: ¿Puedes dar la relación que guardan los autovectores, los autovalores y los conjuntos de nivel del desarrollo de Taylor en torno a un extremo?

```
In [10]: x0, y0 =              0.,             0.
         ux, uy =  1./np.sqrt(2.), 1./np.sqrt(2.)
         vx, vy = -1./np.sqrt(2.), 1./np.sqrt(2.)
         gf.contour(fun, contours = 40)
         gf.arrow(x0, y0, ux, uy)
         gf.arrow(x0, y0, vx, vy);
```



El desarrollo de Taylor de segundo orden en torno a un punto extremo *siempre* tiene ejes de simetría, que vienen dados por los autovectores.

La ecuación de los conjuntos de nivel de valor c en el desarrollo de Taylor de segundo orden en un punto extremo, (x'_0, y'_0), en la base de autovectores, viene dada por:

$$f(x'_0 + v'_x,\, y'_0 + v'_y) \simeq f(x'_0, y'_0) + \frac{1}{2}\left(\lambda'_1 v'^2_x + \lambda'_2 v'^2_y\right) = c$$

Los conjuntos de nivel son *elipses* si es un mínimo o un máximo, o *hypérbolas* si es un punto silla, respecto sus ejes de simetría.

Los autovalores nos cuantifican la curvatura en cada uno de los ejes de simetría. La curvatura es más acuciada conforme más grande es el autovalor.

O también podemos asociarlos a los ejes de las elipses de los conjuntos de nivel. El eje de la elipse es mayor para autovalores menores.

Generalicemos ahora para funciones escalares de varias dimensiones

Sea la función escalar, $f(\mathbf{x})$, de $\mathrm{R}^n \to \mathrm{R}$, un punto $\mathbf{x}_0$ es **extremo** o **crítico** si su gradiente es nulo, $\nabla f(\mathbf{x}_0) = \mathbf{0}$.

Decimos que un punto extremo es **mínimo local** si todos los puntos dentro de una bola de tamaño δ, centrada en $\mathbf{x}_0$ tienen un valor de la función menor o igual al de $\mathbf{x}_0$

$$\mathbf{x} \text{ t.q. } \| \mathbf{x} - \mathbf{x_0} \| < \mathbf{ffi} \Rightarrow \mathbf{f}(\mathbf{x}) \geq \mathbf{f}(\mathbf{x_0})$$

Y de forma similar, serrá un **máximo local** si:

$$\mathbf{x} \text{ t.q. } \| \mathbf{x} - \mathbf{x_0} \| < \mathbf{ffi} \Rightarrow \mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x_0})$$

Finalmente, es **punto silla** si en toda bola centrada en $\mathbf{x}_0$, hay puntos dentro de la bola cuyo valor de la función es mayor y otros menor que el valor en el punto crítico.

El punto extremo es un mínimo, si todos los **autovalores**, $\{\lambda_i, i = 1, \ldots, n\}$, de su **matriz hessiana**, $\mathbf{H}(\mathbf{x_0})$, son positivos; es un máximo si todos son negativos; y es un punto silla si alguno es negativo y alguno positivo.

Cuestión: Considera una función escalar $f(x, y)$ y el punto extremo, (x_0, y_0), el determinante de la matriz hessiana es negativo. ¿Puedes clasificar el extremo, es mínimo, máximo, punto silla? ¿Y si el determinante es positivo?

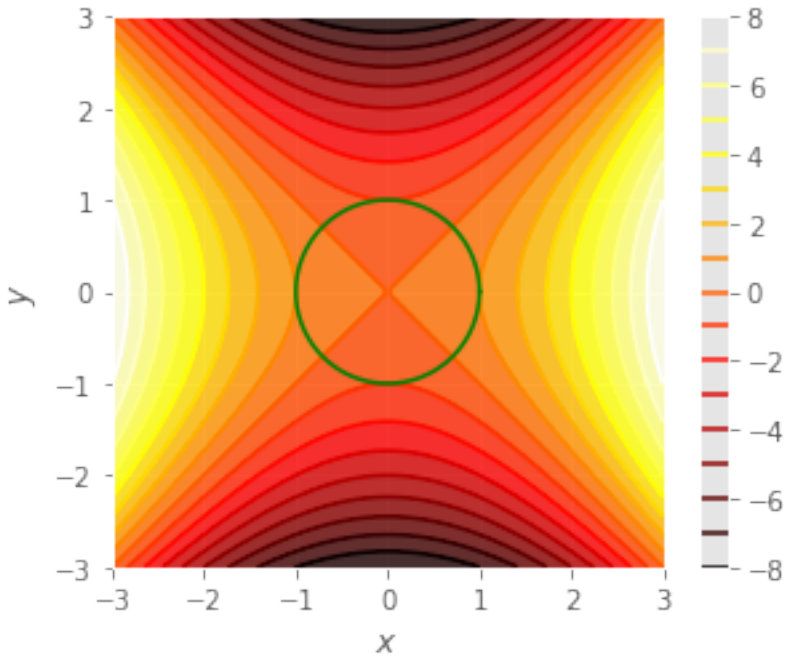
Mínimos y máximos condicionados Considera el caso de que queremos calcular los extremos de una función escalar, $f(\mathbf{x})$, pero no en todo su dominio, si no en una parte del mismo. En este caso hablamos de **extremos condicionados**.

Por ejemplo, sea la función: $f(x, y) = x^2 - y^2$, y queremos calcular sus extremos en la circunferencia de radio unidad, esto es, en los puntos que cumplen $x^2 + y^2 = 1$.

La siguiente celda te dibuja los conjuntos de nivel de la función y la circunferencia. Como puedes ver hemos parametrizado la circunferencia en función de t que toma los valores en $[0, 2\pi]$.

Cuestión: ¿Puedes identificar los puntos extremos de la función a lo largo de la circunferencia?

```
In [11]: f  = lambda x, y : x*x - y*y
         xt  = lambda t    : np.cos(t)
         yt  = lambda t    : np.sin(t)
         gf.contour(f);
         gf.line2d(xt, yt, trange=(0., 2*np.pi, 100), color = 'green',newfig = False);
```



Vamos a dibujar sobre un determinado punto el gradiente de la función, que como sabemos nos indica la dirección en la que la función sufre el cambio máximo.

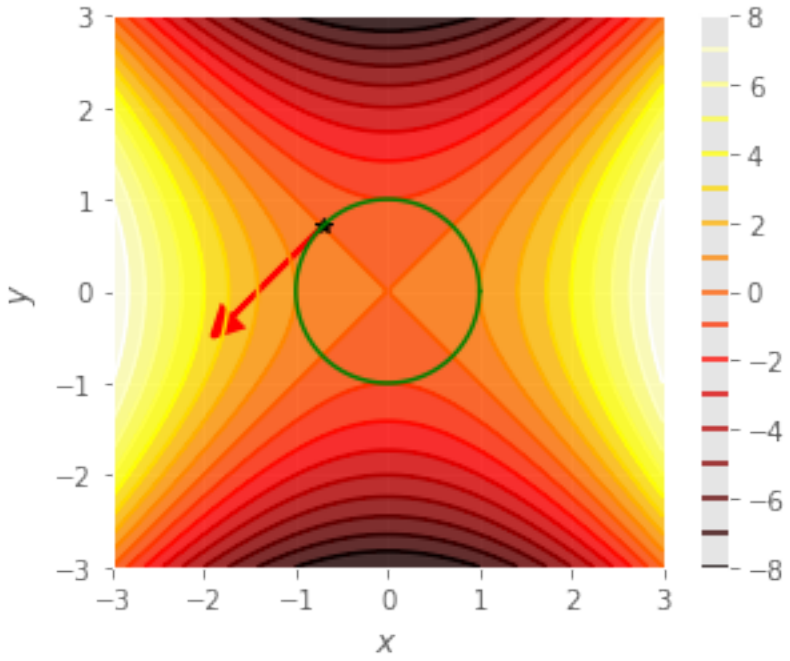
$$f(x,y) = x^2 - y^2 \Rightarrow \nabla f(x,y) = (2x, 2y)$$

Considera primero el valor $t = 3\pi/4$, que nos da el punto $\left(-1/\sqrt{2}, 1/\sqrt{2}\right)$.

Si quisieramos dirigirnos en una dirección en la que la función aumentase sin salir de la circunferencia, solo podríamos movernos hacia la izquierda, (como si el gradiente tirase de un movil que solo pudiera desplazarse a lo largo de la circunferencia), pero ¿hasta cuándo?

Considera ahora el valor $t = \pi$, esto es el punto $(-1, 0)$. El gradiente ahí es radial, si queremos permanecer en la circunferencia, entonces ya no hay ninguna dirección que nos permita hacerlo y que a su vez nos permita aumentar el valor de la función. Hemos alcazado el máximo condicionado de la función.

```
In [12]: t0  = 3.*np.pi/4
         fpx = lambda x, y :  2*x
         fpy = lambda x, y : -2*y
         x0, y0 = xt(t0), yt(t0)
         gf.contour(f);
         gf.dot(x0, y0)
         gf.line2d(xt, yt, trange=(0., 2*np.pi, 100), color = 'green',newfig = False);
         gf.arrow(x0, y0, fpx(x0, y0), fpy(x0, y0), color = 'red'); # nabla f
```



Si te das cuenta el extremo de la función coincide en el punto cuyo gradiente sea *ortogonal* a la circunferencia.

La circunferencia podemos darla como el conjunto de nivel de una función, (con $c = 1$), simplemente:

$$g(x, y) = x^2 + y^2 = c$$

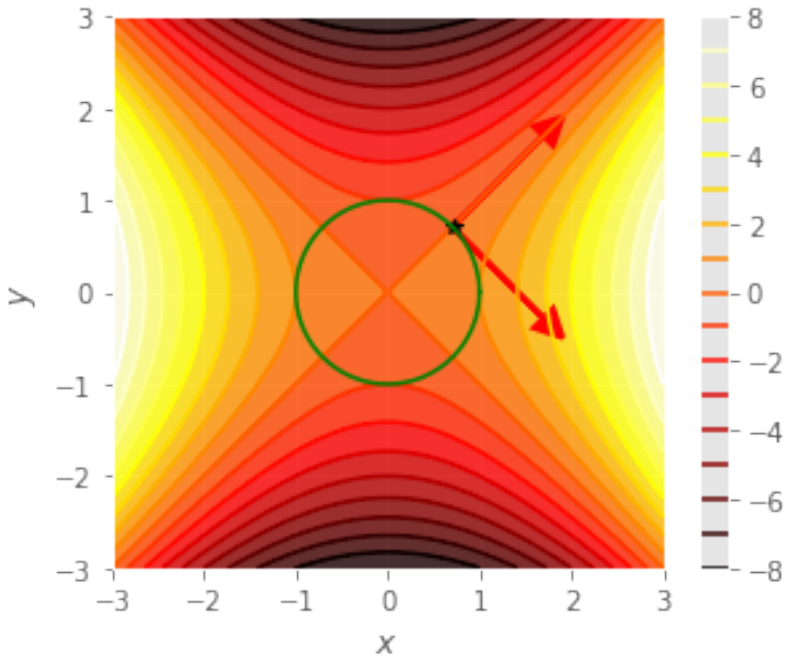
y sabemos que el gradiente de esta función siempre sera *ortogonal* al conjunto de nivel.

Luego el punto extremo, (x_0, y_0) los dos gradientes, el de la función y el de la condición de conjunto de nivel, son proporcionales

$$\nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0)$$

donde λ es un valor de proporcionalidad.

```
In [13]: t0  = 0.5*(np.pi/2.)
         x0, y0 = xt(t0), yt(t0)
         g   = lambda x, y : x*x + y*y
         gpx = lambda x, y : 2*x
         gpy = lambda x, y : 2*y
         gf.contour(f);
         gf.dot(x0, y0)
         gf.line2d(xt, yt, trange=(0., 2*np.pi, 100), color = 'green',newfig = False);
         gf.arrow(x0, y0, fpx(x0, y0), fpy(x0, y0), color = 'red'); # nabla f
         gf.arrow(x0, y0, gpx(x0, y0), gpy(x0, y0), color = 'red'); # nabla g
```



Multiplicadores de Lagrange Sea una función escalar, $f(\mathbf{x})$, los extremos, $\mathbf{x_0}$ de esa función, donde $\mathbf{x}_0$ pertenece a los conjuntos de nivel dados por $g(\mathbf{x}) = c$, cumplen que:

$$\nabla f(\mathbf{x}_0) = \lambda \nabla g(\mathbf{x}_0)$$

donde λ es un factor de proporcionalidad, llamado **multiplicador de Lagrange**.

Ejercicio: Calcula los extremos condicionados de la función $f(x, y) = x^2 - y^2$ en la circunferencia de radio unidad

La condición podemos darla por:

$$g(x, y) = x^2 + y^2$$

La condición de Lagrange:

$$\nabla f(x_0, y_0) = \lambda \nabla g(x_0, y_0) \;\Rightarrow\; (2x_0, -2y_0) = \lambda(2x_0, 2y_0)$$

De la primera condición $2x_0 = \lambda 2x_0$, obtenemos $\lambda = 1$; pero para que se cumpla se segunda $-2y_0 = \lambda 2y_0$, si $\lambda = 1$, entonces $y_0 = 0$. Solo hay dos puntos en la circunferencia con $y_0 = 0$, son $(-1, 0)$ y $(1, 0)$. En ambos se cumple la condición de Lagrange. Son por lo tanto extremos condicionados.

De la segunda condición $2y_0 = -\lambda 2y_0$, obtenemos $\lambda = -1$; pero para que se cumpla en este caso la primera, $2x_0 = 2\lambda x_0$ con $\lambda = -1$, entonces $x_0 = 0$. Solo hay dos puntos en la circunferencia con $x_0 = 0$ que son: $(0, 1)$ y $(0, -1)$. Son también extremos condicionados

La función en $(-1, 0)$ y $(1, 0)$ vale 1. Mientras que en $(0, 1)$ y $(0, -1)$ vale -1. Los dos primeros son máximos condicionados y los dos últimos mínimos condicionados.

Verifica: que es correcto en la celda anterior, colocando $t = 0, \pi/2, \pi, 3\pi/2$, y observa como los gradientes se alinean y coinciden con los valores máximos y mínimos que puede tomar la función en la circunferencia.

¡Aún hay más!

Joseph Louis Lagrange

Fue uno de los grandes matématicos y físicos del siglo XVIII y principios del XIX. Un hombre de su tiempo, italiano del norte que recorrió la Europa de su época, y que fue profesor en la gran escuela "École Polytechnique", recién fundada por Napoleón. A él le debemos muchas contribuciones, entre ellas las reformulación de la mecánica Newtoniana. Aquí tienes su entrada en la wikipedia: https://en.wikipedia.org/wiki/Joseph-Louis_Lagrange

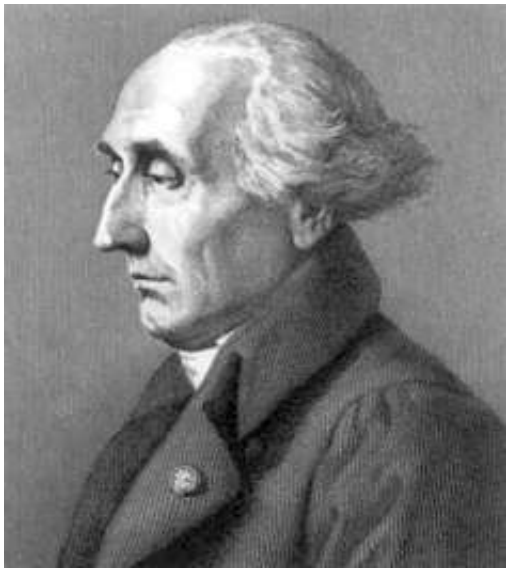
Principio de mínima acción

Los máximos y mínimos juegan un papel fundamental en Física. De hecho, la mecánica clásica (y también la moderna) pueden formularse en función del principio de mínima acción, que nos dice que la dinámica de un sistema físico está regida por la búsqueda de la mínima acción, que matemáticamente se define como la integral de un término que se denomina, **Lagrangiano**. Aquí en la wikipedia: https://en.wikipedia.org/wiki/Principle_of_least_action

Este principio se enunció por primera vez por el físico francés Maupertius en el siglo XVIII, y en su versión morerna, en el siglo XX por el gran físico y profesor americano Richard **Feymann**. Aquí tienes un enlace a una de sus famosas clases : http://www.feynmanlectures.caltech.edu/II_19.html y su entrada en la wikipedia: https://en.wikipedia.org/wiki/Richard_Feynman

El método de mínimos cuadrados

Los mínimos y máximos juegan un papel fundamental también en la estimación de parámetros y observables.



Joseph Louis Lagrange

Richard Feymann

Seguramente has utilizado en técnicas experimental de laboratorio el método de los mínimos cuadrados para hacer el ajuste de unos datos, por ejemplo, para ajustarlos a una recta.

Como sabes la Naturaleza tiene un comportamiento probabilístico, y nuestras medidas estás sujetas *siempre* a las incertidumbres de medida, físicas o teóricas.

En el método de mínimos cuadrados, se calcula la distancia (al cuadrado) de los valores medidos frente al valor de una función de hipótesis, por ejemplo la recta, que depende de unos parámetros a estimar, (en el caso de la recta son la pendiente, a, y el umbral en el origen, b). Estimamos sus parámetros y sus incertidumbres calculando el mínimo de la suma de las distancias al cuadrado, a esta función la llamamos, χ^2, que en el caso de ajuste a una recta, es una función escalar de dos variables, a, b, los parámetros de la recta.

En las siguientes celdas hemos preparado un experimento de ajuste a una recta.

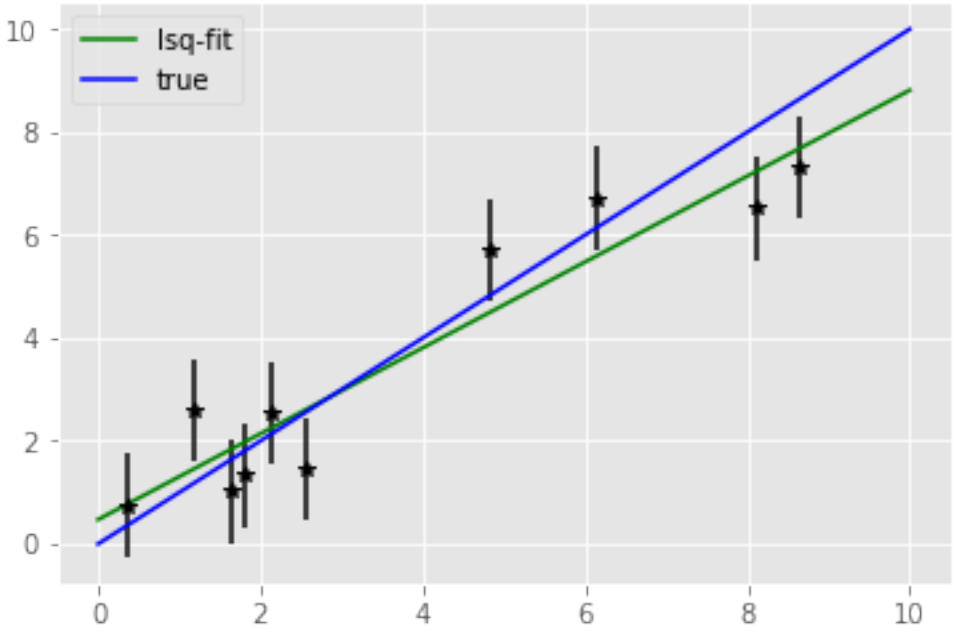
Primero el experimento genera n puntos aleatorios (x_i, y_i) a lo largo de una recta, cada uno con la misma incertidumbre σ. Luego estimamos la pendiente y el umbral mediante el método de mínimos cuadrados.

En la figura se dibujan los puntos con su error, la recta verdadera y la recta estimada.

In [14]:
```
import scipy.stats    as stats
import scipy.optimize as optimize

def experiment(a, b, sigma, npoints, xrange = (0, 10)):
    x0, xf = xrange
    xs = x0 + (xf - x0)*stats.uniform.rvs(size = npoints)
    ys = a * xs + b + stats.norm.rvs(scale = sigma, size = npoints)
    return (xs, ys)

def lsq_fit(xs, ys, sigma, aguess =1, bguess = 0.):
    fres = lambda pars: (ys - pars[0]*xs - pars[1])/sigma
    x, _ = optimize.leastsq(fres, (aguess, bguess))
    return x

def chisq(a, b):
    chi2 = 0.
    for i in range(len(ys)):
        dx = (ys[i] - a*xs[i] -b)/sigma
        chi2 += dx*dx
    return chi2/(len(ys)-2)
```

In [15]:
```
atrue, btrue, sigma, npoints = 1., 0., 1., 10
xrange = (0, 10)
xs, ys     = experiment(atrue, btrue, sigma, npoints, xrange = xrange)
ahat, bhat = lsq_fit(xs, ys, sigma)
plt.errorbar(xs, ys, yerr=sigma, fmt='*', color = 'black')
ts = np.linspace(*xrange, 100)
plt.plot(ts, bhat  + ahat *ts, color = 'green', label = 'lsq-fit')
plt.plot(ts, btrue + atrue*ts, color = 'blue' , label = 'true')
print('a = ', ahat, 'b = ', bhat)
plt.legend();
```

```
a =  0.8339852650033879 b =  0.4731658503105632
```



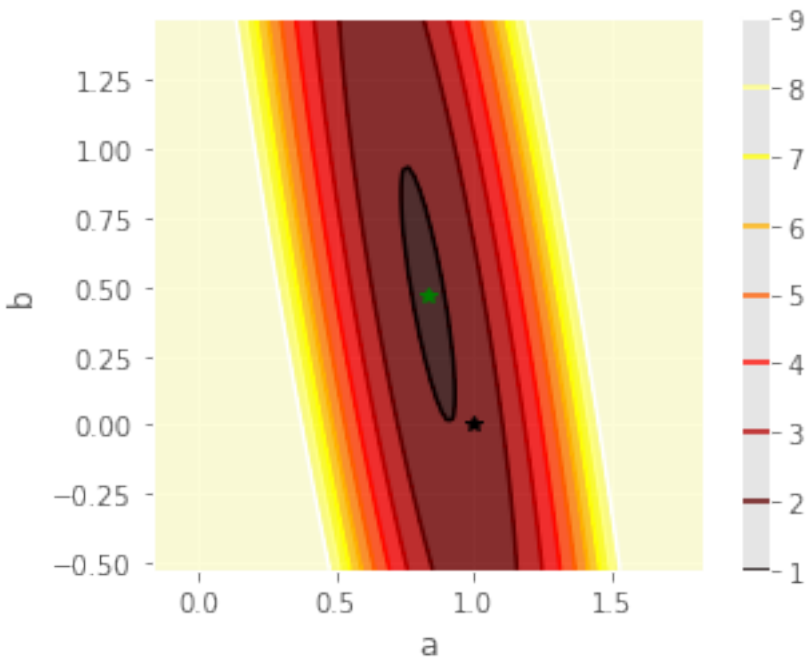
En la siguiente celda, dibujamos la función $\chi^2(a,b)$ para los datos del experimento. Esto es:

$$\chi^2(a,b) = \frac{1}{n-2}\sum_{i=1}^{n}\frac{(y_i - ax_i - b_i)^2}{\sigma^2}$$

El divisor $n-2$ nos da cuenta de los grados de libertad del sistema, el número de medidas, n menos el número de parámetros a estimar que son dos: a y b.

Puedes ver para este experimento que los valores estimados nos dan el mínimo de χ^2 y puedes ver también los conjuntos de nivel a distinto valores, que al ser próximos al mínimo deben ser elipses, y que nos dan una estimación de la incertidumbre de los parámetros a, b. El valor verdadero es el punto en negro, mientras que el estimado es el punto en rojo.

```
In [16]: xchi =  chisq(ahat, bhat)
         print('chisq ', xchi)
         gf.contour(chisq, contours = 10, xrange = (ahat - 1, ahat + 1, 100), zlim = (0, 10),
                    yrange = (bhat - 1, bhat +1, 100))
         gf.dot(ahat, bhat, color = 'green'); gf.dot(atrue, btrue, color = 'black');
         ax = plt.gca(); ax.set_xlabel('a'), ax.set_ylabel('b');

chisq  0.9020493954439668
```



Las redes neuronales

Las redes neuronales son algoritmos matemáticos que pueden 'aprender' cómo identificar patrones o clasificar muestras. Por ejemplo, Google desarrolló una potente red neuronal que puede distingir imágenes de gatos y perros. Se llaman así porque simulan el corportamento de las redes neuronales del sistema nervioso.

Las redes están compuestas de neuronas, que se disponen en capas. En la figura se muestra un esquema de una red neuronal. Cada neurona tienen un conjunto de entradas y una sola salida. Matemáticamente cada neurona es una función escalar. La última capa de la red suele ser una sola neurona y su salida la respuesta final. Por ejemplo, su respuesta es un valor entre [0, 1], donde el 0 indica que la imagen era de un gato y 1 de un perro.

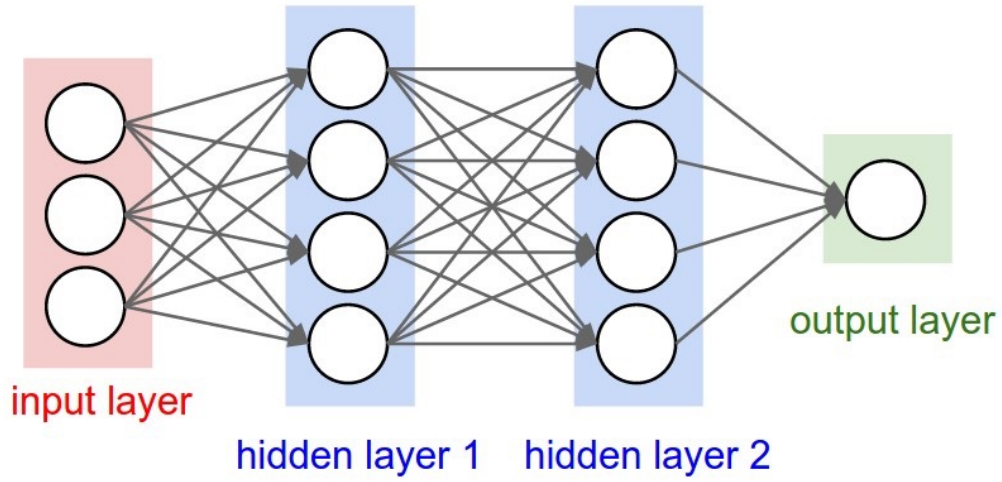
Cada una de las neuronas recibe la entrada de las neuronas de la capa anterior. La neurona lm es la neurona m-ésima de la capa l-ésima. Ella recibe como entrada las respuestas de las neuronas de la capa anterior, x_i, donde i recorre todas la neuronas de la capa $l-1$, supongamos que hay N neuronas en todas las capas. Cada neurona en sí es una función escalar, que a partir del vector de entradas $\mathbf{x}$, y mediante un vector interno $\mathbf{a}$, y una posible constante b, da respuesta a partir de la cantidad escalar $\mathbf{ax}+b$, mediante una función real:

$$u(\mathbf{ax}+b)$$

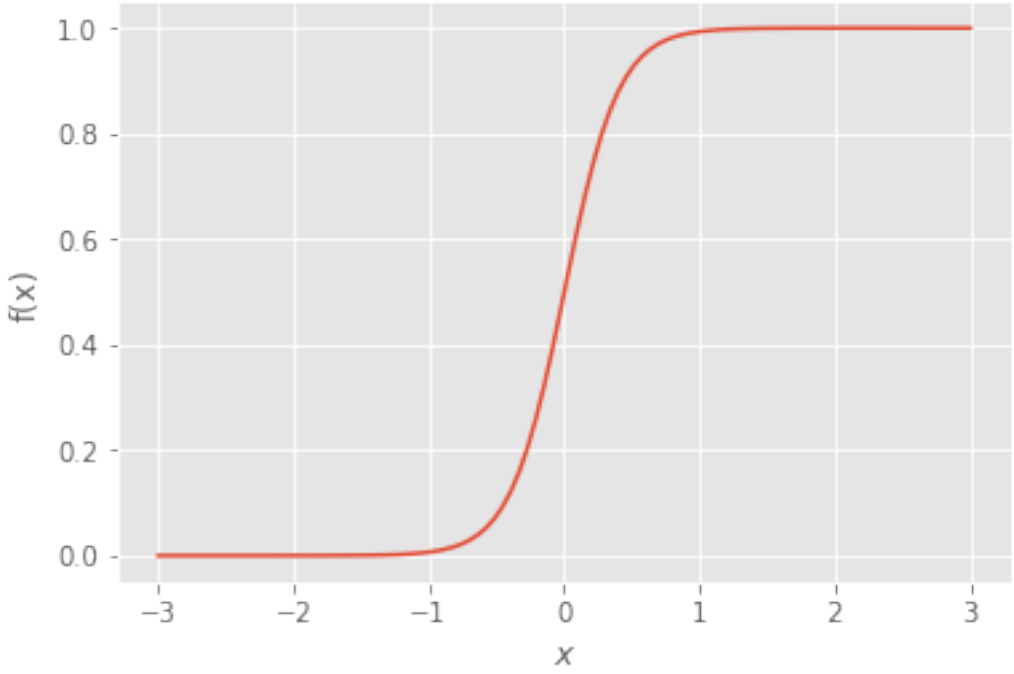
donde $\mathbf{a}, b$ son los parámetros de cada neurona. La función más común es un sigmoide, que tiene su salida comprendida entre $[0, 1.]$. En la siguiente celda se muestra la fución sigmoide.

$$u(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{ax}-b}}$$

```
In [17]: a, b = 5., 0
         f = lambda x: 1/(1+np.exp(-a*x - b))
         gf.fun1d(f);
```



Esquema de una red neuronal



Pero, ¿Cómo funcionan?

Mediante entrenamiento, imaginate que le muestras una imagen de un gato a las neuronas iniciales (cada neurona de la primera capa 've' un 'pixel' de la imagen) y esperas que de como respuesta un 0, si le enseñas la imagen de un perro debe dar un 1.

La red neuronal es pues en sí misma una función escalar que a partir de la imagen inicial $\mathbf{x}$ (los pixels de la imagen son x_{ij}) nos da una sola respuesta $v(\mathbf{x})$. Entrenamos la red mediante la construcción de siguiente función error, que compara la respuesta de la red, $v(\mathbf{x}_i)$, con el valor

verdadero, y_i, donde 0 corresponde a la imagen de un gato y 1 de un perro perro, con un número M, $(i = 1, \ldots, M)$, grande, de imágenes.

$$E(\mathbf{a}^{lm}, b^{lm}) = \sum_{i=1}^{M} (v(\mathbf{x_i}) - y_i)^2$$

Los parámetros de la función error, $E(\mathbf{a}^{lm}, b^{lm})$, son muchísimos, todos los parámetros internos, $(\mathbf{a}, b)$, de las neuronas, del orden de $LN(N+1)$, donde L es el número de capas y N el número de neuronas por capa.

Si te das cuenta, la función error será *mínima* cuando la red 'acierte' la imágenes, esto es su salida, $v(\mathbf{x}_i)$ coincida con el valor verdadero y_i, 0 para gato y 1 para perro.

¿Cómo calculamos pues los parámetros internos de las neuronas? ¡Minimizamos la función error! Algo parecido a lo que hicimos con el ajuste a una recta por mínimos cuadrados con la función χ^2. Claro que como hay tantos parámetros no es trivial. Ahí es donde entra Google y sus empleados matemáticos, para desarrollar algorizmos que minimicen rápidamente. ¡Sus algorizmos estan basados en calcular grandientes!

Una vez la red está "entrenada" o "ha aprendido" (esto es, la hemos minimizado), le podemos mostrar otras fotos de gatos y perros y para cada una de ellas, nos dará una respuesta.

¿Y ésto tiene que ver con la Física?

Más de lo que te imaginas. En la actualidad pocos son los análisis en Física de Partículas, por ejemplo, donde no se utilice una red neuronal o un método similar.

Se utilizan para identificar si una partícula es un electrón o un muón (su hermano más pesado). O si en una interacción hay un bosón de Higgs o no. Para decidir si una interacción es relevante o es del montón.

Ya te puedes imaginar que las redes neuronales están detrás de algunas aplicaciones como Facebook, Instagram, Spotify y de todos los genios matématicos que online intentan predecir tus gustos.

Si quieres apreder algo más sobre la redes neuronales puedes leer este libro online: http://neuralnetworksanddeeplearning.com/index.html y puedes experimental con el modulo de Python https://scikit-learn.org/stable/

¡Esto es todo por ahora!

7.5 Apendices

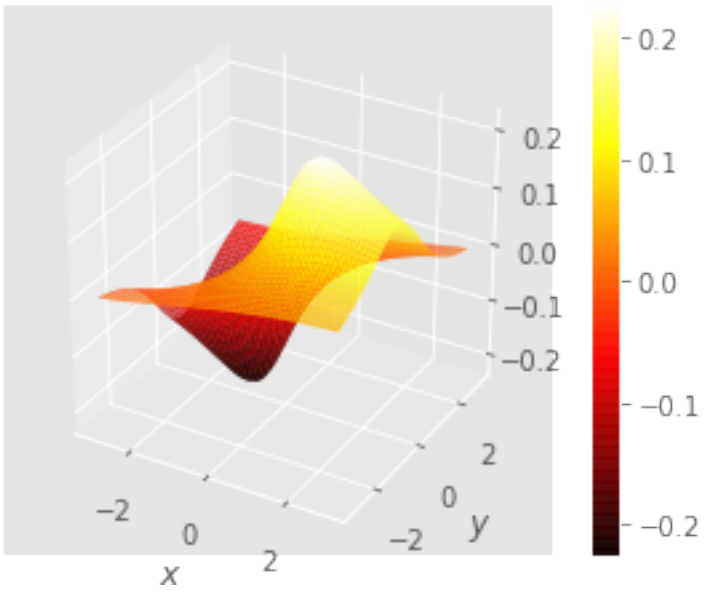
¡Algo más complicado!

Ejercicio: Utiliza Python y las métodos del modulo scipy para calcular los extremos de la función:

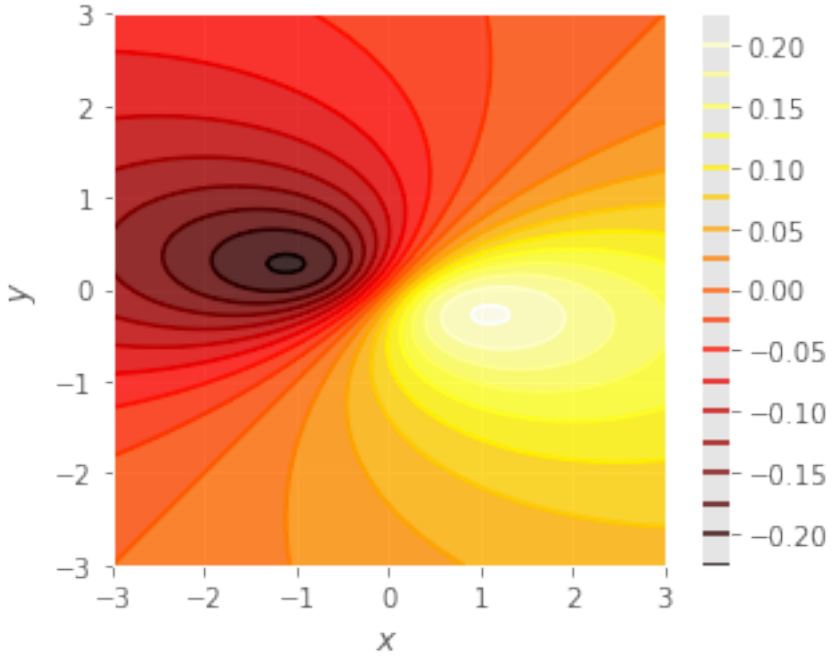
$$f(x, y) = \frac{x - y}{2x^2 + 8y^2 + 3}$$

¡Empieza por dibujar la función y sus conjuntos de nivel!

```
In [18]: fun = lambda x, y: (x-y)/(2*x*x + 8*y*y +3)
         gf.graph(fun);
```



In [19]: `gf.contour(fun);`



De los conjuntos de nivel observamos que hay un mínimo en torno al $(-1,0)$ y un máximo cerca del $(1,0)$, y que los ejes de simetría en los extremos coinciden con los ejes coordenadas.

En las siguientes celdas, primero calculamos numéricamente el punto mínimo y máximo.

Luego usaremos cálculo simbólico para calcular las derivadas parciales y segundas, para después evaluar la matriz hessiana en los puntos extremos.

Finalmente calcularemos numéricamente los autovalores y autovectores de la matriz hessiana, y comprobaremos que se trata de un máximo y un mínimo y como los autovectores coinciden con los vectores de coordenadas.

```
In [20]: import scipy.optimize as optimize
         import scipy.linalg   as linalg
         fun = lambda x : (x[0]-x[1])/(2*x[0]*x[0] + 8*x[1]*x[1] + 3)
         res1 = optimize.minimize(fun, (-1, 0))
         x0, y0 = res1.x
         print('punto mínimo ', x0, y0)
         fun = lambda x : -(x[0]-x[1])/(2*x[0]*x[0] + 8*x[1]*x[1] + 3)
         res2 = optimize.minimize(fun, (1, 0))
         x1, y1 = res2.x
         print('punto máximo ', x1, y1)

punto mínimo  -1.0954920902489502 0.2738621986560991
punto máximo  1.0954920606202636 -0.27386221675122946


In [21]: import sympy
         x, y = sympy.symbols('x y')
         fx = sympy.diff((x-y)/(2*x*x+8*y*y+3), x)
         fy = sympy.diff((x-y)/(2*x*x+8*y*y+3), y)
         print('fx ', fx)
         print('fy ', fy)

fx  -4*x*(x - y)/(2*x**2 + 8*y**2 + 3)**2 + 1/(2*x**2 + 8*y**2 + 3)
fy  -16*y*(x - y)/(2*x**2 + 8*y**2 + 3)**2 - 1/(2*x**2 + 8*y**2 + 3)


In [22]: fxx = sympy.diff(fx, x)
         fxy = sympy.diff(fx, y)
         fyy = sympy.diff(fy, y)
         fyx = sympy.diff(fy, x)
         print('fxx ', fxx)
         print('fxy ', fxy)
         print('fyx ', fyx)
         print('fyy ', fyy)

fxx  32*x**2*(x - y)/(2*x**2 + 8*y**2 + 3)**3 - 8*x/(2*x**2 + 8*y**2 + 3)**2 - 4*(x - y)/(2*x**2
fxy  128*x*y*(x - y)/(2*x**2 + 8*y**2 + 3)**3 + 4*x/(2*x**2 + 8*y**2 + 3)**2 - 16*y/(2*x**2 + 8*
fyx  128*x*y*(x - y)/(2*x**2 + 8*y**2 + 3)**3 + 4*x/(2*x**2 + 8*y**2 + 3)**2 - 16*y/(2*x**2 + 8*
fyy  512*y**2*(x - y)/(2*x**2 + 8*y**2 + 3)**3 + 32*y/(2*x**2 + 8*y**2 + 3)**2 - 16*(x - y)/(2*x


In [23]: print('punto ', x0, y0)
         hxx = fxx.subs([(x, x0), (y, y0)])
```

```
        hxy = fxy.subs([(x, x0), (y, y0)])
        hyx = fyx.subs([(x, x0), (y, y0)])
        hyy = fyy.subs([(x, x0), (y, y0)])
        h   = np.array([[float(hxx), float(hxy)], [float(hyx), float(hyy)]])
        print('hessiana ', h)
        eigvals, eigvecs = linalg.eig(h)
        print('autovalores', eigvals)
        print('autovector 1 :', eigvecs[:, 0])
        print('autovector 2 :', eigvecs[:, 1])
```

```
punto  -1.0954920902489502 0.2738621986560991
hessiana  [[1.52129395e-01 5.62777899e-06]
 [5.62777899e-06 6.08558515e-01]]
autovalores [0.15212939+0.j 0.60855852+0.j]
autovector 1 : [-1.00000000e+00  1.23300174e-05]
autovector 2 : [-1.23300174e-05 -1.00000000e+00]
```

```
In [24]: print('punto ', x1, y1)
         hxx = fxx.subs([(x, x1), (y, y1)])
         hxy = fxy.subs([(x, x1), (y, y1)])
         hyx = fyx.subs([(x, x1), (y, y1)])
         hyy = fyy.subs([(x, x1), (y, y1)])
         h   = np.array([[float(hxx), float(hxy)], [float(hyx), float(hyy)]])
         print('hessiana ', h)
         eigvals, eigvecs = linalg.eig(h)
         print('autovalores', eigvals)
         print('autovector 1 :', eigvecs[:, 0])
         print('autovector 2 :', eigvecs[:, 1])
```

```
punto  1.0954920606202636 -0.27386221675122946
hessiana  [[-1.52129403e-01 -5.63252983e-06]
 [-5.63252983e-06 -6.08558504e-01]]
autovalores [-0.1521294+0.j -0.6085585+0.j]
autovector 1 : [ 1.00000000e+00 -1.23404266e-05]
autovector 2 : [1.23404266e-05 1.00000000e+00]
```