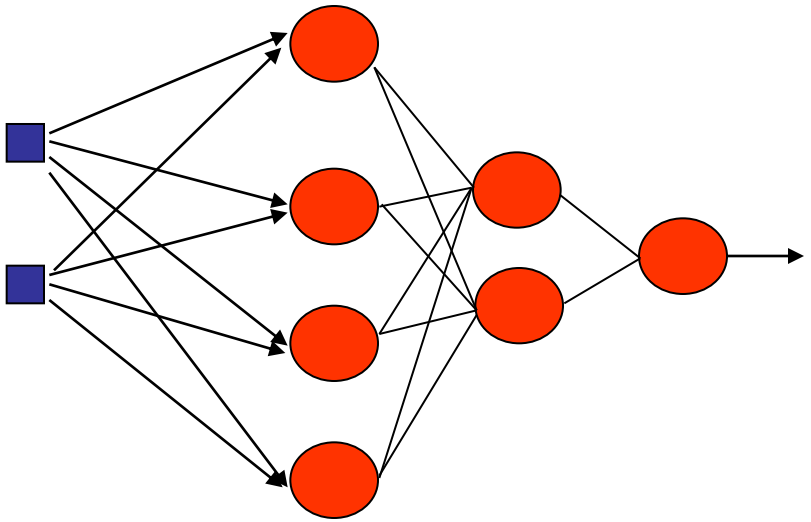
MLP
Multiple Layer Perceptron
(PERCEPTRONES MULTICAPAS)



El algoritmo de descenso de gradiente es intuitivo y sencillo. Si reconocemos que el algortimo del BP no es otra cosa que la minimización del error del entrenamiento sobre el conjunto de variables (parámetros libres), caemos en el terreno (fértil, extenso y en expansión) de optimización no-lineal.

Existe una amplia variedad de ideas, motivaciones y métodos. Nos vamos a pasear por algunos de los más conocidos y usados por la comunidad RNA:

- **Gradiente conjugado**
- **Método de newton y Quasi-newton**
- **Levenberg-Marquadt**

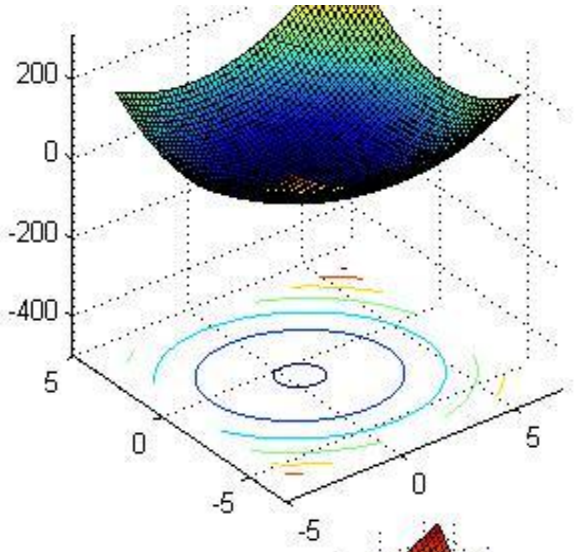
Una forma cuadrática es una función de la forma:

$$f(x) = \frac{1}{2} x^t H x - b^t x + E$$

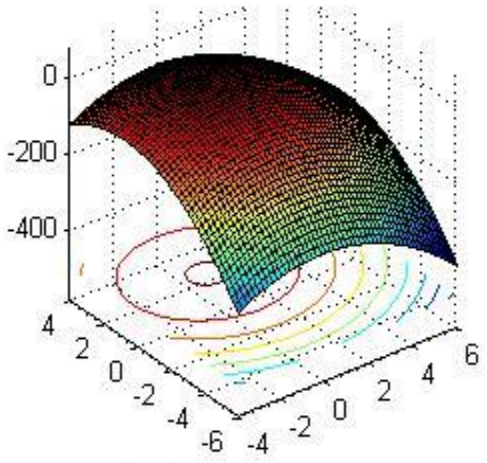
Si H es simétrica y positiva definida (usando cálculo vectorial) el mínimo de esta función se encuentra en la solución del sistema

$$Hx = b$$

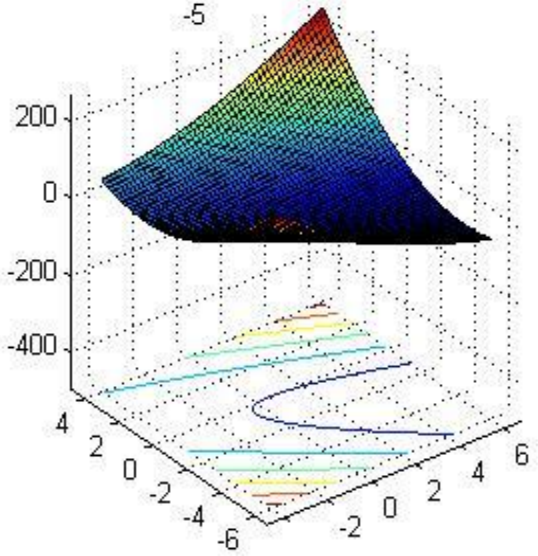
PD



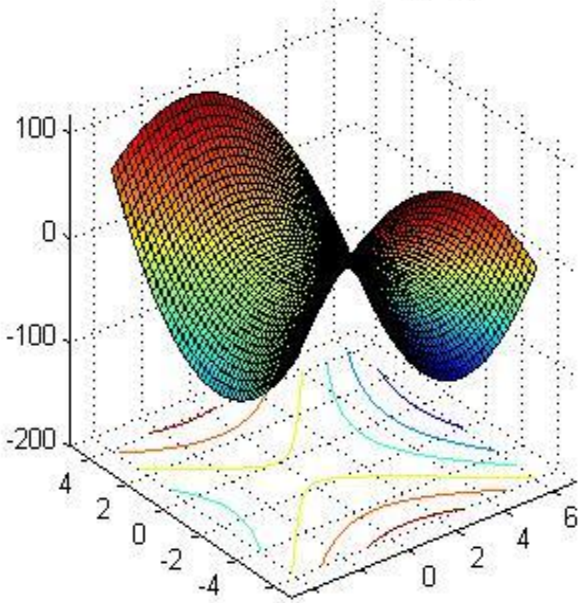
ND



Sing + PD

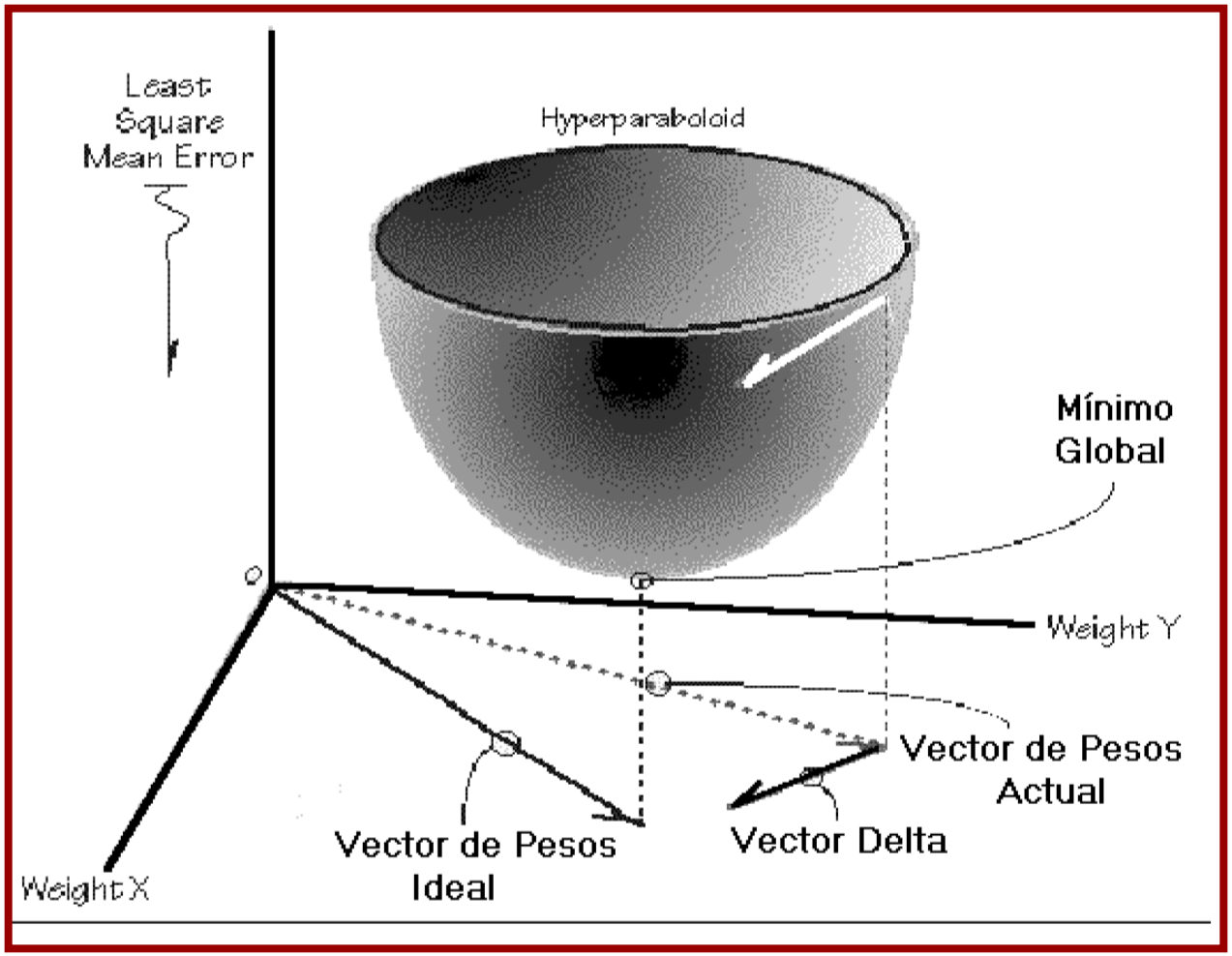


Sing

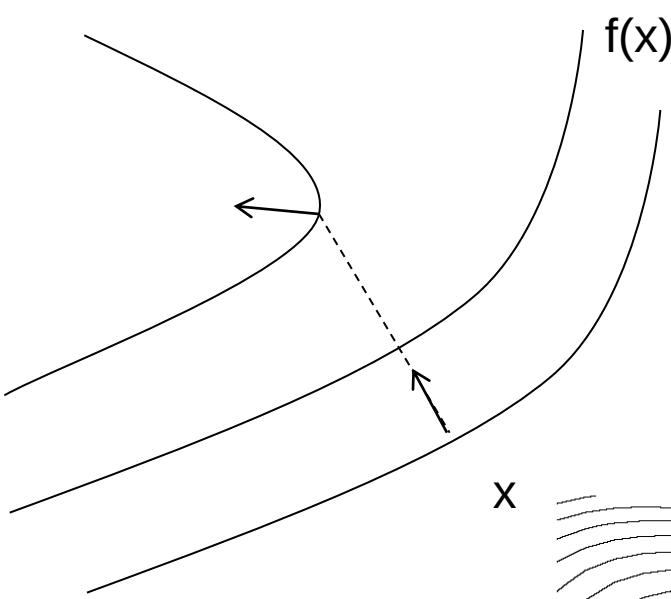


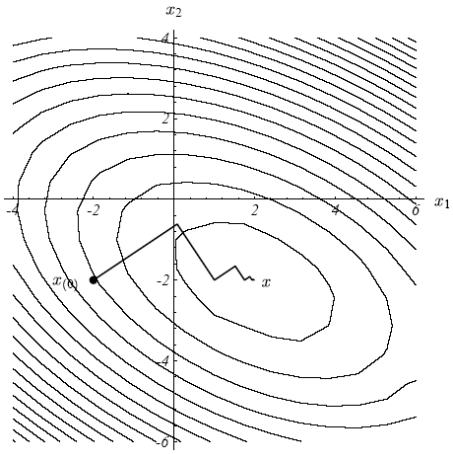
Qué hay de malo con descenso de gradiente?

Recordar cómo funciona el descenso del gradiente?



En cada paso las soluciones se mueven a lo largo de la dirección de descenso, esto trae como consecuencia que cada paso sea perpendicular al paso anterior.





Suponga que acaba de minimizar en dirección a u, ∇f es perpendicular a u. Porqué?

Porque si no habríamos podido continuar minimizando en esa dirección!

Se produce un zig-zag, que puede hacer que la convergencia sea lenta, sobre todo cuando la iteración entra en un "valle estrecho."

El algortimo general es:

Para m=1,2,3,3,……repetir hasta alcanzar la condición de parada:

$$u = \nabla f(x^{(m)})$$

$$\text{si } u = 0, \text{ parar. En caso contrario } t^* = \operatorname{argmin}\ g(t) = f(x^{(m)} + tu)$$

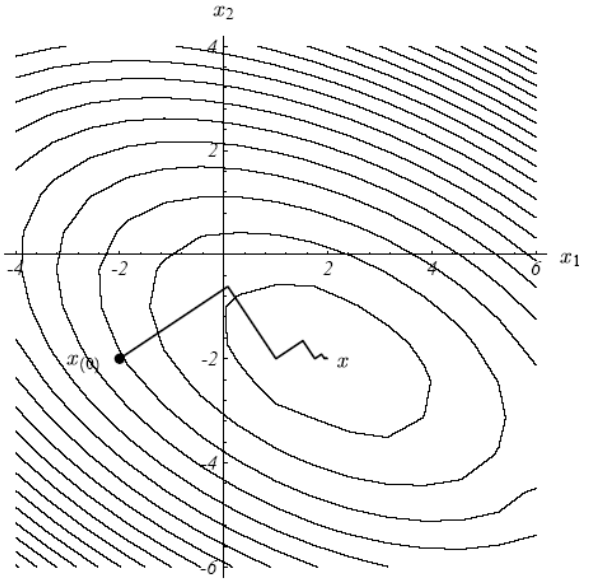
$$\text{hacer } x^{(m+1)} = x^{(m)} + t^* u$$

En el algortimo del LMS, el tamaño del paso era fijo, sin embargo en la versión más general se puede caminar en la dirección de descenso tanto como se pueda!!

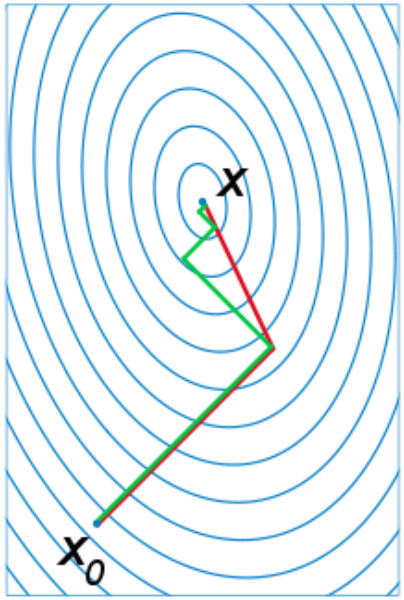
Se puede demostrar que este método converge globalmente a un mínimo local, con velocidad de convergencia lineal.

Muchas veces descenso de gradiente elige direcciones que ya ha usado antes.

Sería mucho mejor tomar esa dirección una sola vez!



Idea: Elegir un conjunto de direcciones de búsqueda ortogonales, $d_1, d_2, \ldots, d_w$ que tomemos una sola vez de modo que en w pasos lleguemos al objetivo!!



Queremos encontrar una dirección v de tal manera que que "un paso no destruya lo que hizo el anterior".

En descenso de gradiente tomamos como siguiente dirección a $-\nabla f$, en el método de gradiente conjugado perturbamos $-\nabla f$ (v= $-\nabla f$ + "algo"). Y queremos elegirlo de modo tal de no deshacer.

Idea: Solicitar que ∇f sea perpendicular antes y después de movernos a lo largo de v. Es decir que la condición local sea que el cambio en ∇f sea perpendicular a u.

Un cambio pequeño en x, δx producirá un cambio en ∇f, $\delta(\nabla f)$ y la conexión viene por el Hessiano.

$$\delta(\nabla f) = \left[\nabla^2 f\right]\delta x$$

Esta idea induce a que la condición de "no-interferencia" conlleva a :

$$u^t \delta\left(\nabla^2 \mathrm{f}\right) = 0$$
$$u^t \left[\nabla^2 \mathrm{f}\right] v = 0$$

Dada una matriz H $w \times w$ simétrica y positiva definida (PD) decimos que el conjunto de vectores : $\{d(0), d(1), \ldots\ldots, d(w-1)\}$ es *A*- conjugado si

$$d^t(n) H d(j) = 0 \; \forall n \text{ y } j \mid n \neq j.$$

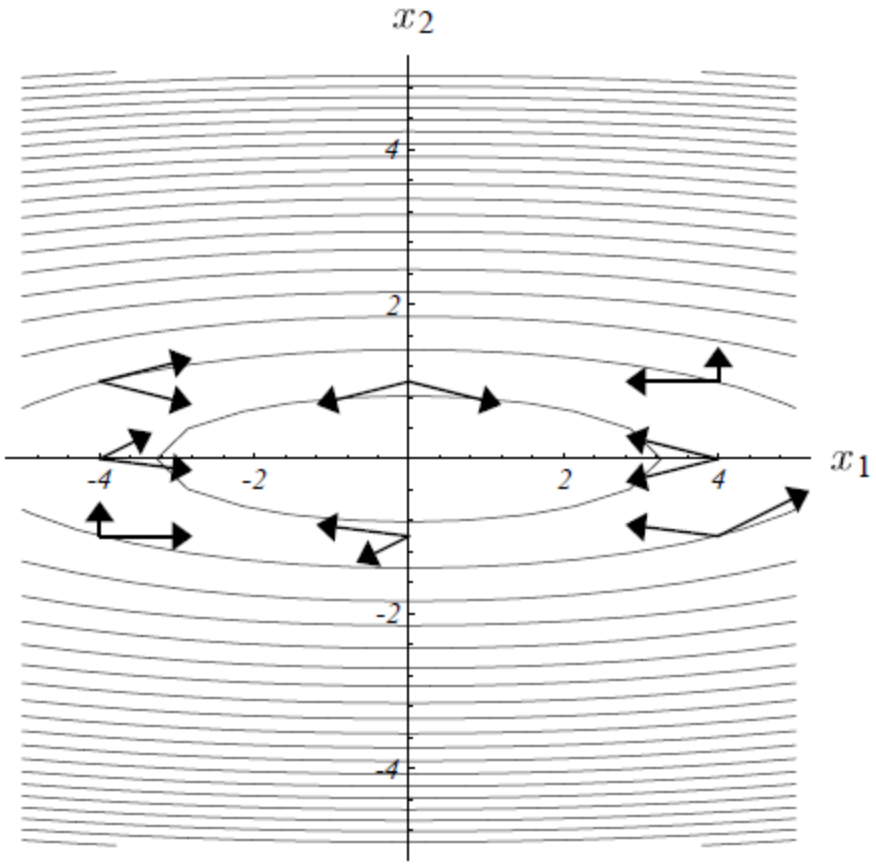
Nota: Si $H = I$ conjugado implica ortogonalidad.

Si $\mathrm{d}(0) = \sum_{j=1}^{w-1} \alpha_j d(j),$ ⇒ $\mathrm{d}^{\mathrm{t}}(0) H d(0) = \sum_{j=1}^{w-1} \alpha_j d^t(0) H d(j) = 0.$

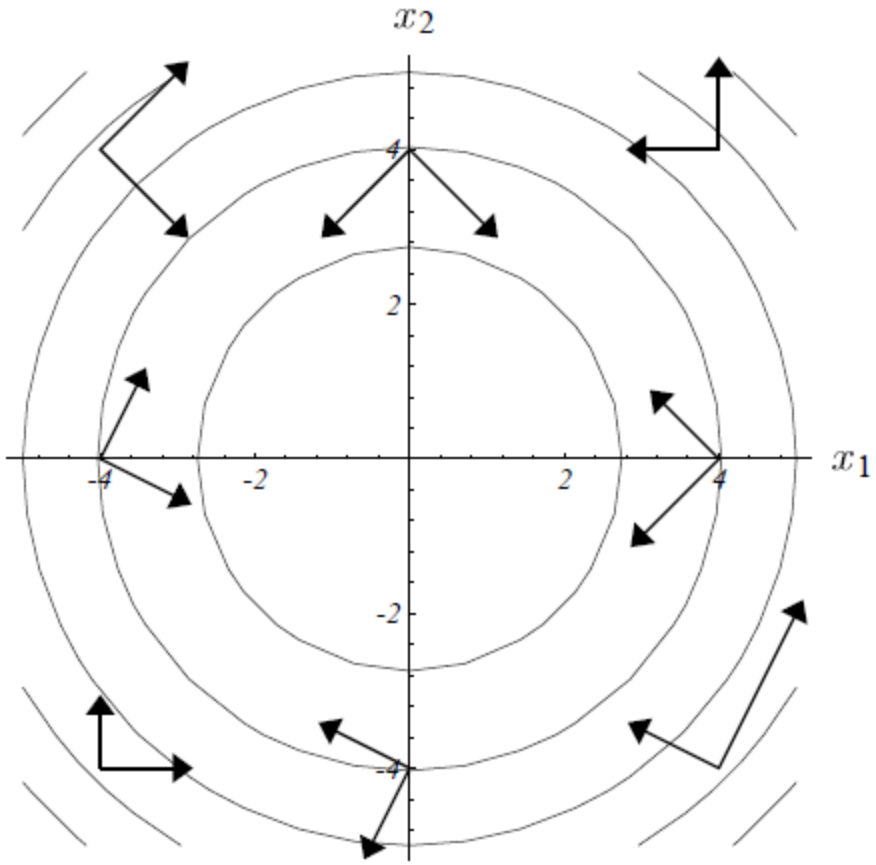
Resultado: El conjunto de vectores H-conjugado es L.I.

H es PD y $\mathrm{d}(0) \neq 0$

Cómo podemos usar esto para hacer una minimización?



(a)



(b)

Suponga que se quiere minimizar una función de error

$$f(x) = \frac{1}{2} x^t H x - b^t x + E$$

y sea x^* el mínimo de esa función. Si estamos en x_1, quisiera recorrer hasta x^* y por independencia lineal:

$$x^* - x_1 = \sum_{j=1}^{w} \alpha_j d(j) \qquad (1)$$

Definamos:

$$x_j = x_1 + \sum_{i=1}^{j-1} \alpha_i d(i)$$

de (1) los podemos definir de forma iterativa como: $x_{j+1} = x_j + \alpha_j d(j)$ (2)

Para hallar los α_j……

$$x^* - x_1 = \sum_{j=1}^{w} \alpha_j d(j)$$

$$d^t(j)Hx^* - d^t(j)Hx_1 = \sum_{i=1}^{w} \alpha_i d^t(j)Hd(i)$$

$$-d^t b - d^t(j)Hx_1 = \alpha_j d^t(j)Hd(j)$$

$f(x) = \frac{1}{2}x^t Hx - b^t x + E$ el mínimo se alcanza en $0 = Hx - b \equiv g(x)$

Agrupando

$$-d^t(j)(b + Hx_1) = \alpha_j d^t(j)Hd(j)$$

$$\alpha_j = -\frac{d_j^t\,(b + Hx_1)}{d_j^t Hd_j}$$

$$\alpha_j = -\frac{d_j^t(b + Hx_1)}{d_j^t H d_j}$$

Recordemos que :

$$\boldsymbol{x_j} = \boldsymbol{x_1} + \sum_{i=1}^{j-1} \boldsymbol{\alpha_i d_i}$$

$$\boldsymbol{d_j H x_j} = \boldsymbol{d_j H x_1} + \mathbf{0}$$

$$d_j^t(b + Hx_1) = d_j^t\left(b + Hx_j\right) = d_j^t g_j$$

$$\alpha_j = -\frac{d_j^t g_j}{d_j^t H d_j}$$

Equivalente a nuestra tasa de aprendizaje

Hecho: Cada nueva dirección es ortogonal a los gradientes en las iteraciones anteriores….

$$g_{j+1} - g_j = H(x_{j+1} - x_j) = \alpha_j H d_j \quad \text{(usando : } \quad \boldsymbol{x_{j+1} = x_j + \alpha_j d(j)} \text{ (2))}$$

$$d_j^t g_{j+1} - d_j^t = d_j^t \left[\frac{-d_j^t g_j}{d_j^t H d_j}\right] H d_j \quad \Rightarrow \quad d_j^t g_{j+1} = 0$$

Similarmente $\quad d_k^t(g_{j+1} - g_j) = \alpha_j d_k^t H d_j = 0, \ \forall k < j \leq w.$

Aplicando inducción:

$$d_k^t g_j = 0, \forall k < j \leq w.$$

Para completar el proceso nos falta construir las direcciones que sean H- conjugadas….

- **Elegimos a $d_1 = -g_1$**
- **Si $d_{j+1} = -g_{j+1} + \beta_j d_j$ (3)**

donde $\beta_j = \frac{g_{j+1}^t H d_j}{d_j^t H d_j}$ (multiplicando por Hd_j la ecuación de actualización anterior), los vectores generados son H-conjugados (y tendríamos w de ellos):

$$d_{j+1}^t H d_j = -g_{j+1}^t H d_j + \left(\frac{d_j^t H d_j}{d_j^t H d_j}\right) g_{j+1}^t H d_j = 0$$

Recapitulamos: **Para una función cuadrática** $f(x) = \frac{1}{2}x^t Hx - b^t x + E$ **queremos caminar por las direcciones** H**-conjugadas de la siguiente manera:**

$$x_{j+1} = x_j + \alpha_j d(j)$$

Donde los tamaños de paso (tasa de aprendizaje) α_j **están definidos como:**

$$\alpha_j = -\frac{d_j^t g_j}{d_j^t H d_j}$$

Y las direcciones d_j **definidas según la regla iterativa:**

$$d_{j+1} = -g_{j+1} + \beta_j d_j$$

Con $d_1 = -g_1$ **y**

$$\beta_j = \frac{g_{j+1}^t H d_j}{d_j^t H d_j}$$

¿Cómo podemos usar este algoritmo para entrenar a nuestro perceptrón?

$$\varepsilon_{av}\big(w(n)+\Delta w(n)\big)=\varepsilon_{av}\big(w(n)\big)+g^{t}(n)\Delta w(n)+\frac{1}{2}\Delta w^{t}(n)H(n)\Delta w(n)+\ \ orden\ sup.$$

$$g=\frac{\partial\varepsilon_{av}(w)}{\partial w}\qquad\qquad H=\frac{\partial^{2}\varepsilon_{av}(w)}{\partial w^{2}}$$

Peeeero. Pequeño detallazo: El cálculo de los α y β necesitan del hessiano (H) evaluado en cada iteración!! ($$$)

Caso β

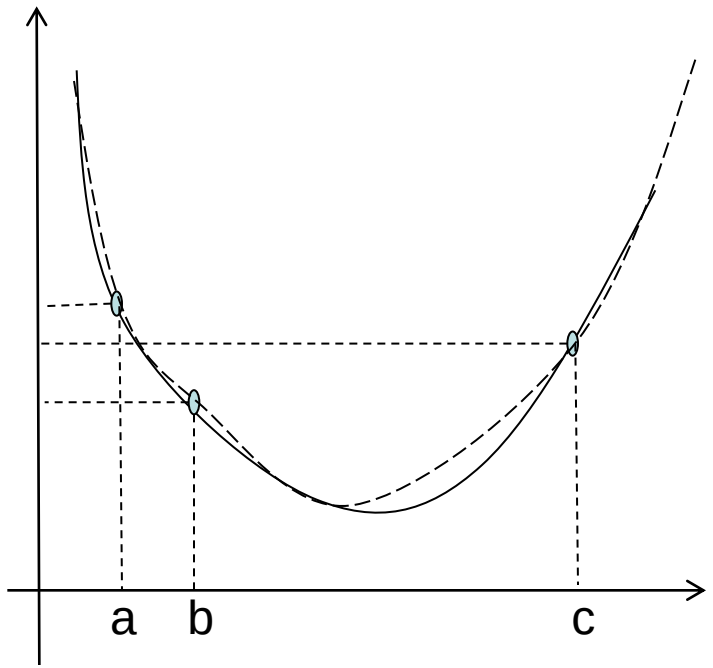
- Polak –Ribiére

$$\beta(n) = \frac{g_n^t(g_n - g_{n-1})}{g_{n-1}^t g_{n-1}}$$

- Fletcher-Reeves

$$\beta(n) = \frac{g_n^t g_n}{g_{n-1}^t g_{n-1}}$$

Caso α



Algoritmo:

1. Escoger peso inicial w_1
2. Evaluar g_1 y tomar $d_1 = -g_1$
3. $\alpha_j = argmin\, \varepsilon(w_j + \alpha\, d_j)$
4. Actualizo: $w_{j+1} = w_j + \alpha_j d_j$
5. Probar condición de parada
6. Evaluar nuevo gradiente g_{j+1}
7. Busco β_j para luego encontrar nueva dirección d_{j+1}
8. $j = j + 1$
9. Ir al paso 3.

$$f(x) = \frac{1}{2} x^t H x - b^t x + E$$

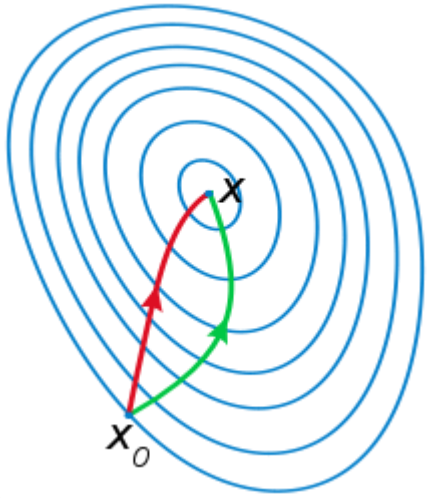
La condición del mínimo es que el gradiente se anule y en el caso cuadrático esto implica

$$w^* = w - H^{-1} g$$

Y que llegamos al óptimo en un solo paso.

En el caso más general, usamos la misma idea y cada paso es tomado en la dirección de $H^{-1}g$

El problema es que el cálculo del Hessiano es muy costoso ($O(nw^2)$) (y mucho más lo es su inversa!! $O(w^3)$), lo cual lo hace un método prohibitivo



Newton vs. descenso de gradiente

Idea: Construir aproximaciones al hessiano en varios pasos, generando una secuencia de matrices tales que $G^{\tau} \rightarrow H^{-1}$

Se arranca con algo PD y se asegura no perder esa condición en cada iteración. La condición quasi-newton es:

$$w^{\tau+1} - w^{\tau} = -H^{-1}(g^{\tau+1} - g^{\tau})$$

Se harán iteraciones de forma:

$$\mathrm{w}^{\tau+1} = \mathrm{w}^{\tau} + \alpha^{\tau} G^{\tau} g^{\tau}$$

$$G^{\tau+1} = G^{\tau} + \frac{pp^t}{p^t v} - \frac{(G^{\tau} v) v^t G^{\tau}}{v^t G^{\tau} v} + (v^t G^{\tau} v) u\, u^t$$

$$p = w^{\tau+1} - w^{\tau}$$
$$v = g^{\tau+1} - g^{\tau}$$
$$u = \frac{p}{p^t v} - \frac{G^{\tau} v}{v^t G^{\tau} v}$$

La primera dirección es de descenso $(-g)$ y $G_0 = I$

Es uno de los favoritos en la comunidad RNA. Resulta de una particularización del método de Newton para resolver específicamente problemas de mínimos cuadrados no-lineales (tal vez por eso sea favorito).

$$E = \frac{1}{2}\sum_n (\varepsilon^n)^2 = \|\varepsilon\|^2$$

Parados en w_{old} queremos caminar hacia w_{new}, asumiendo que la diferencia esntre ellos es pequeña, podemos expandir en serie de Taylor.

$$\varepsilon(w_{new}) = \varepsilon(w_{old}) + Z\,(w_{new} - w_{old}); \quad Z_{ni} = \partial\varepsilon^n/\partial w_i$$

$$E = \frac{1}{2}\|\varepsilon(w_{old}) + Z\,(w_{new} - w_{old})\|^2$$

Minimizando c.r.a w_{new} tenemos:

$$w_{new} = w_{old} - (Z^t Z)^{-1} Z^t \varepsilon(w_{old})$$

El hessiano

$$H_{ik} = \frac{\partial E}{\partial w_i \partial w_k} =$$

$$\sum_n \left\{ \frac{\partial \varepsilon^n}{\partial w_i}\frac{\partial \varepsilon^n}{\partial w_k} + \varepsilon^n \frac{\partial^2 \varepsilon^n}{\partial w_i \partial w_k} \right\} \approx Z^t Z$$

Si en la minimización $\varepsilon(w_{old})$ es muy grande la aproximación ya no es válida y se considera

$$\tilde{E} = \frac{1}{2}\|\varepsilon(w_{old}) + Z(w_{new} - w_{old})\|^2 + \lambda\|w_{new} - w_{old}\|^2$$

El λ gobierna el tamaño del paso y así la iteración queda:

$$w_{new} = w_{old} - (Z^t Z + \lambda I)^{-1} Z^t \varepsilon(w_{old})$$

Hay que elegir al \lambda en la práctica. Tomar $\lambda = O(10^{-1})$.

Si $E\downarrow$ se acepta w_{new} y $\lambda\downarrow$ por 10

Si $E\uparrow$ se descarta w_{new} y $\lambda\uparrow$ por 10 y se vuelve a iterar .

Mientras $\lambda \gg 1$, $\|w_{new} - w_{old}\| \to 0$