

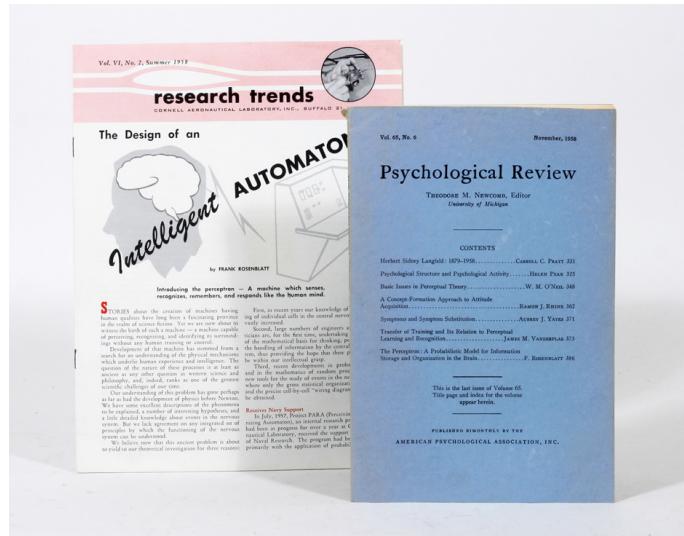
Entrenamiento Básico de Redes Neuronales

Gradientes



Una Larga Espera

- Como hemos dicho, las primeras redes neuronales (McCulloch & Pitts) se "programaban" manualmente para una tarea (prueba & error). Hubo que esperar casi 15 años para que **Frank Rosenblatt** inventara (en 1957) un algoritmo denominado **Perceptrón** que permitía entrenar automáticamente desde ejemplos 1 neurona artificial.

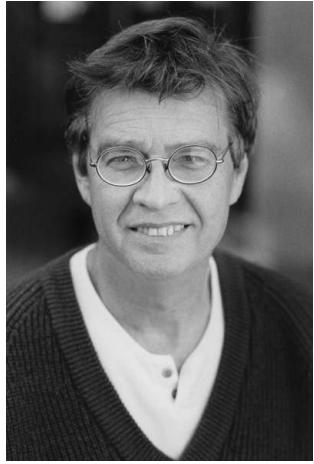


Una Larga Espera

- Pero hubo que esperar mucho más hasta que se inventara (o se afianzara en la comunidad) un método que nos permite entrenar el modelo en el caso general en que se tienen muchas unidades y capas ocultas.



Paul Werbos



David Rumelhart



Geoffrey Hinton

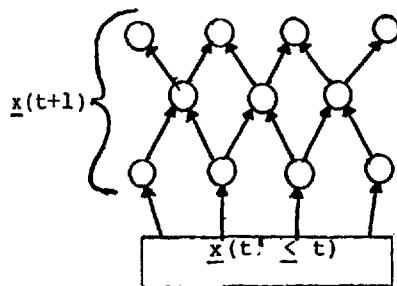
Una Larga Espera

Applications of Advances in Nonlinear Sensitivity Analysis
Paul J. Werbos, U.S. Department of Energy
Forecast Analysis and Evaluation Team

1982

The following paper summarizes the major properties and applications of a collection of algorithms involving differentiation and optimization at minimum cost. The areas of application include the sensitivity analysis of models, new work in statistical or econometric estimation, optimization, artificial intelligence and neuron modelling. The details, references and derivations can be obtained by requesting "Sensitivity Analysis Methods for Nonlinear Systems" from Forecast Analysis and Evaluation Team, Quality Assurance, OSS/EIA, Room 7413, Department of Energy, Washington, DC 20461.

765



"CHAIN RULE" (DYNAMIC FEEDBACK):

$$\frac{\partial^+ x_i}{\partial x_j} = \sum_{k=j+1}^i \frac{\partial^+ x_i}{\partial x_k} \cdot \frac{\partial f_k}{\partial x_j} \quad i > j$$

CONVENTIONAL PERTURBATION:

$$\frac{\partial^+ x_i}{\partial x_j} = \sum_{k=j}^{i-1} \frac{\partial f_i}{\partial x_k} \cdot \frac{\partial^+ x_k}{\partial x_j} \quad i > j$$



Una Larga Espera

CHAPTER 8

Learning Internal Representations by Error Propagation

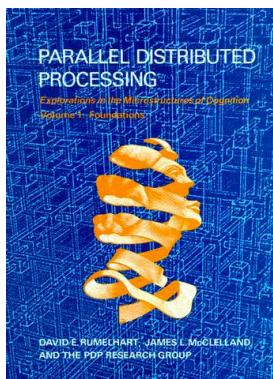
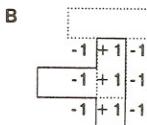
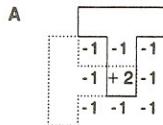
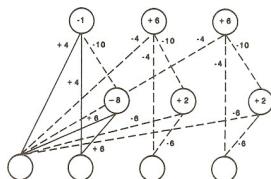
D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

THE PROBLEM

We now have a rather good understanding of simple two-layer associative networks in which a set of input patterns arriving at an input layer are mapped directly to a set of output patterns at an output layer. Such networks have no *hidden* units. They involve only *input* and *output* units. In these cases there is no *internal representation*. The coding provided by the external world must suffice. These networks have proved useful in a wide variety of applications (cf. Chapters 2, 17, and 18). Perhaps the essential character of such networks is that they map similar input patterns to similar output patterns. This is what allows these networks to make reasonable generalizations and perform reasonably on patterns that have never before been presented. The similarity of patterns in a PDP system is determined by their overlap. The overlap in such networks is determined outside the learning system itself—by whatever produces the patterns.

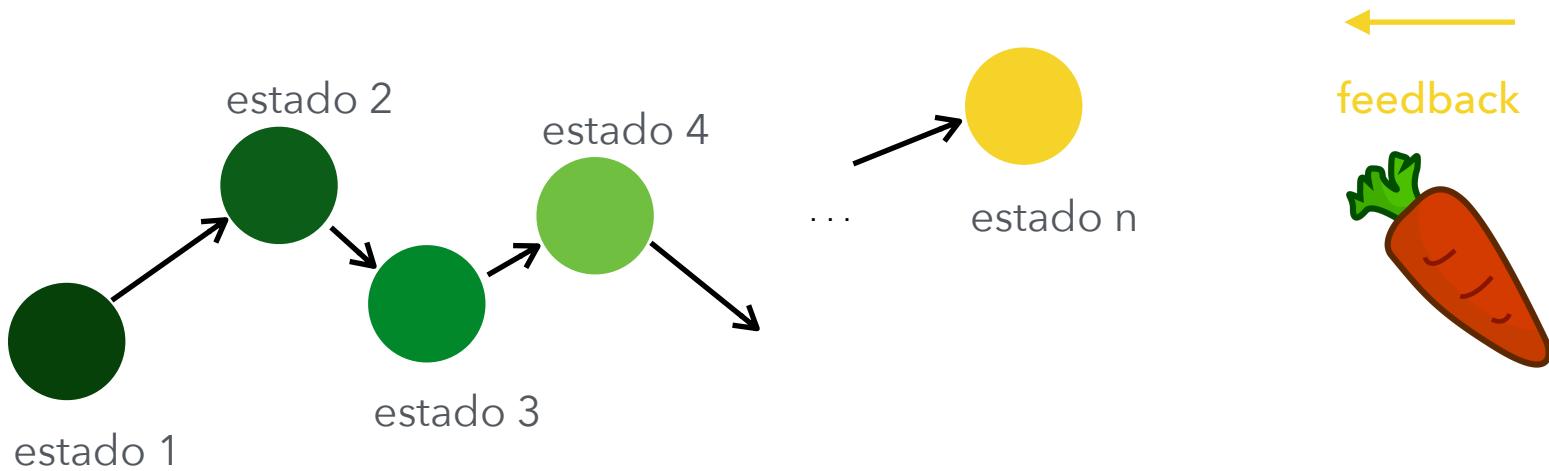
1986

FIGURE 12. The solution discovered for the negation problem. The left-most unit is the negation unit. The problem has been reduced and solved as three exclusive-or problems between the negation unit and each of the other three units.



Credit Assignment

- Dado un grafo de computación con muchos nodos (ocultos) que contribuyen a producir un determinado resultado $f(x)$ cuyo costo Δ podemos medir, ¿cómo determinamos qué unidades del grafo son más o menos responsables de ese error?



Credit Assignment

- ¿Cómo determinamos qué decisiones de una **secuencia de decisiones** son más o menos responsables de un determinado **desenlace** de manera de mejorar la estrategia en el futuro?



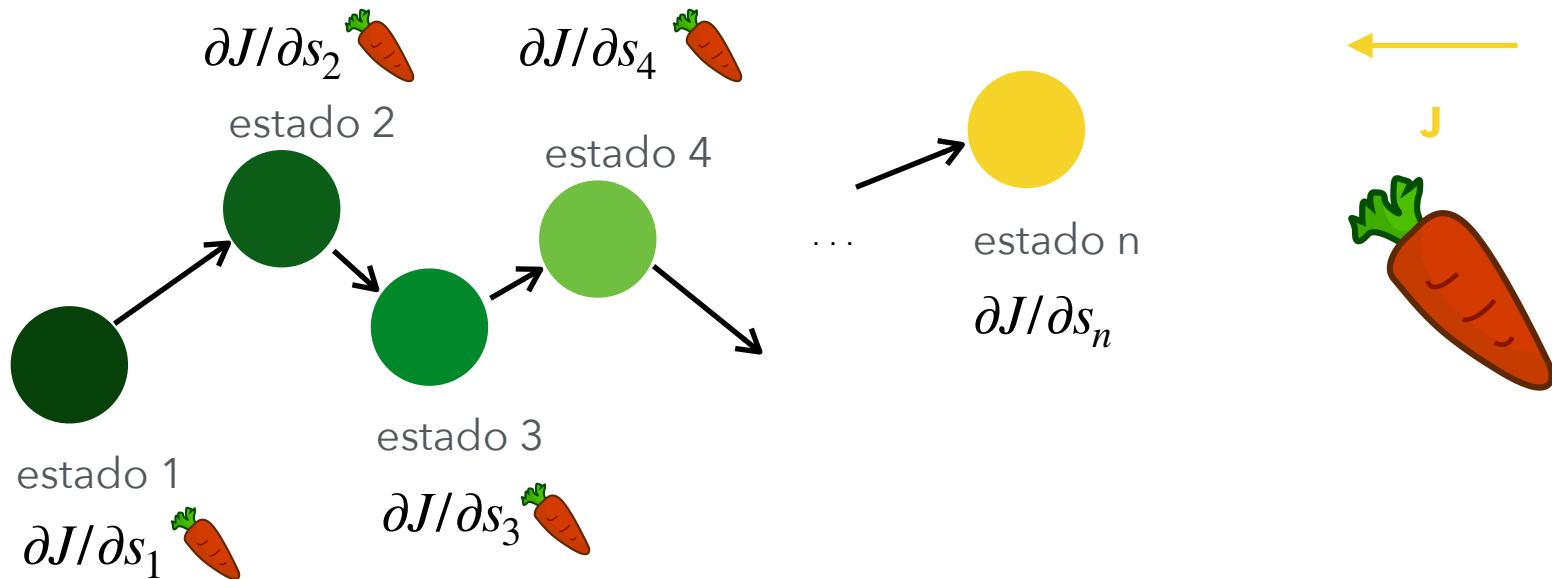
...



mate

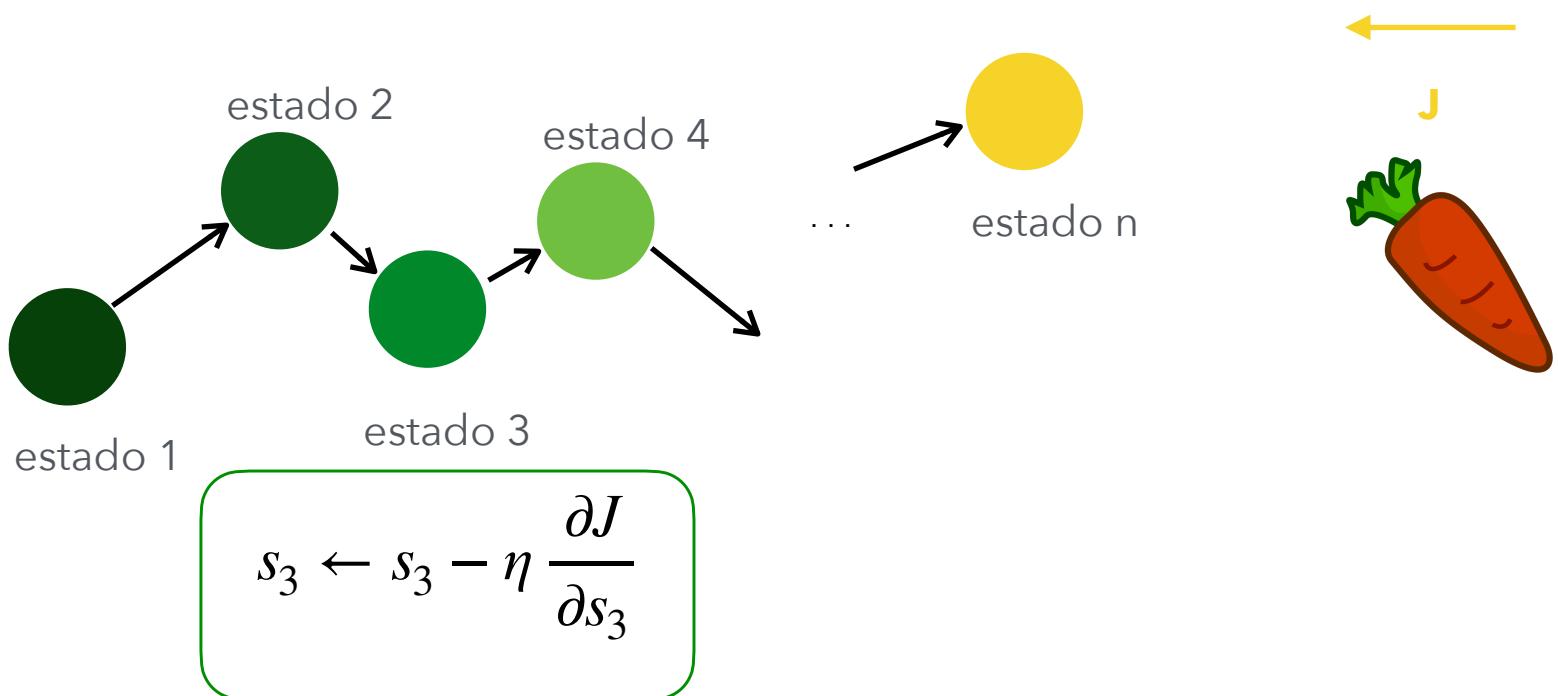
Idea Básica

- Si la función **J** que permite evaluar el resultado de la red es **diferenciable**, podemos usar **derivadas parciales** para transformar feedback global (valor de J) en **feedback local** (crédito de las unidades ocultas)



Idea Básica

- Cada nodo del grafo puede utilizar el feedback recibido para **modificar la regla de cálculo** o decisión que estaba utilizando. En el caso de una neurona artificial esto corresponderá a ajustar sus pesos de conexión y umbral de excitación.



Entrenamiento vía Gradientes

- Sea $f(x; \Theta)$ la función implementada por la red, Θ un vector que contiene todos los parámetros entrenables del modelo (pesos de conexión, umbrales de excitación), y $J(\Theta)$ **la función objetivo** (e.g. error de entrenamiento) con la que deseamos entrenar la red.
- Si $J(\Theta)$ es diferenciable, podemos inicializar Θ y luego entrenar la red iterando hasta convergencia el siguiente algoritmo:

Para $t=1,2, \dots$ (training epochs)

$$\Theta \leftarrow \Theta - \eta_t \nabla J(\Theta)$$

$$\nabla J = \begin{pmatrix} \partial J / \partial \Theta_1 \\ \partial J / \partial \Theta_2 \\ \vdots \\ \partial J / \partial \Theta_m \end{pmatrix}$$

If (convergence) STOP.



Entrenamiento vía Gradientes

- Sea $f(x; \Theta)$ la función implementada por la red, Θ un vector que contiene todos los parámetros entrenables del modelo (pesos de conexión, umbrales de excitación), y $J(\Theta)$ **la función objetivo** (e.g. error de entrenamiento) con la que deseamos entrenar la red.
- Si $J(\Theta)$ es diferenciable, podemos inicializar Θ y luego entrenar la red iterando hasta convergencia el siguiente algoritmo:

Para t=1,2

Para i=1,2 ... m

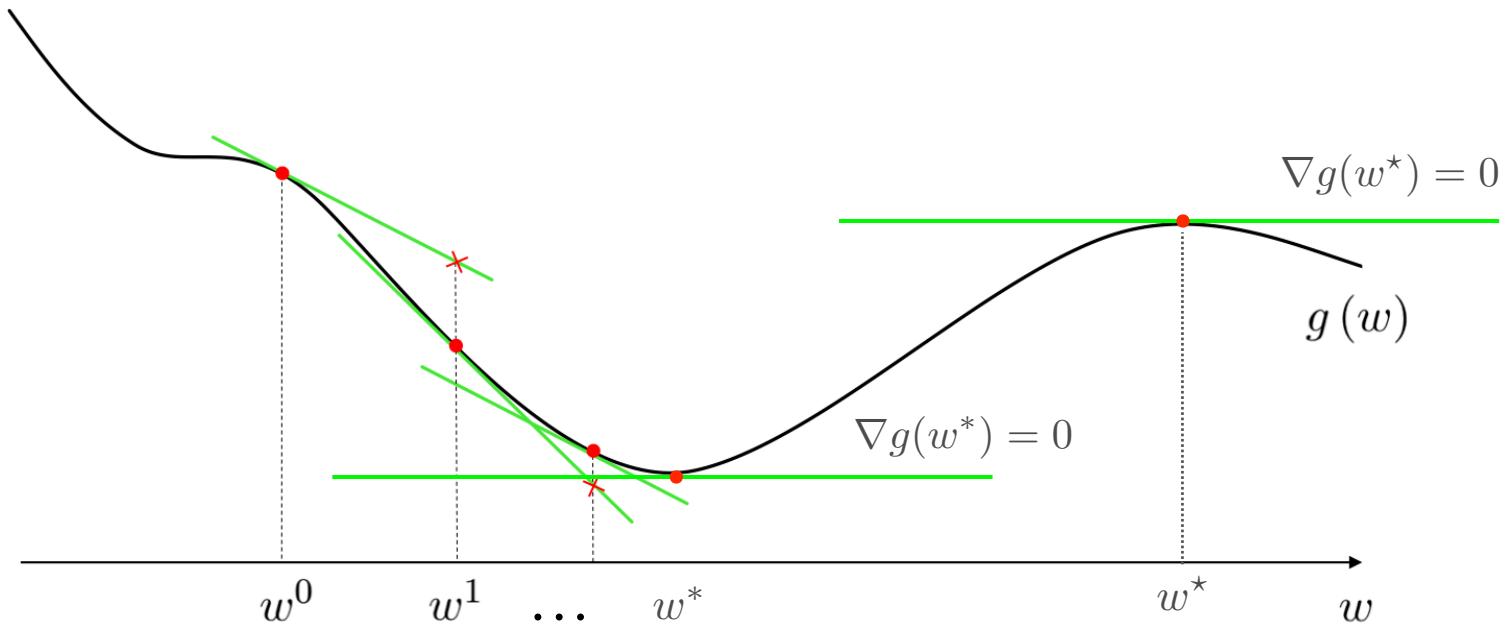
η_t = Learning Rate

$$\Theta_j \leftarrow \Theta_j - \eta_t \frac{\partial J}{\partial \Theta_j}$$

If (convergence) STOP.



Entrenamiento vía Gradientes



Dirección de Descenso

- Supongamos que en cierto momento tenemos una determinada solución para Θ y que consideramos modificar esa solución moviéndonos en la dirección d . Si $J : \mathbb{R}^m \rightarrow \mathbb{R}_0^+$ es diferenciable en Θ , tenemos por el Teorema de Taylor que

$$J(\Theta + \alpha d) = J(\Theta) + \alpha \langle \nabla J(z), d \rangle + o(\alpha)$$

$$\lim_{\alpha \rightarrow 0} \frac{o(\alpha)}{\alpha} = 0$$

- Por lo tanto, si $\nabla J(z) \neq 0$ siempre podremos encontrar un valor $\bar{\alpha}$, tal que $\forall \alpha \in [0, \bar{\alpha}]$
- $$o(\alpha) < \alpha \langle \nabla J(z), d \rangle \Rightarrow J(\Theta + \alpha d) - J(\Theta) \leq 2\alpha \langle \nabla J(z), d \rangle$$
- **Por lo tanto, si $\langle \nabla J(z), d \rangle < 0$, podemos avanzar en la dirección de d para reducir el valor de J !! Se dice que d es una "dirección de descenso".**

Dirección de Máximo Descenso

- Si elegimos $d = -\nabla J$, tenemos que $\forall \alpha \in [0, \bar{\alpha}]$

$$\begin{aligned} J(\Theta + \alpha d) - J(\Theta) &\leq 2\alpha \langle \nabla J(z), d \rangle \\ &\leq -2\alpha \|\nabla J(z)\|^2 \end{aligned}$$

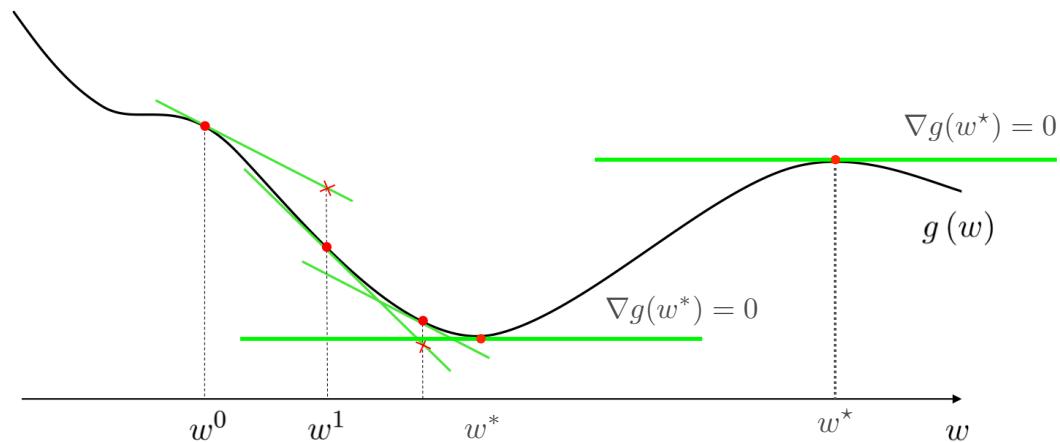
- El valor de $\bar{\alpha}$ podría ser muy pequeño, pero existe un movimiento en la dirección $d = -\nabla J$ que reduce el valor del objetivo.
- Por la desigualdad de Cauchy-Swartz $\langle \nabla J(z), d \rangle \leq \|J(z)\| \|d\|$, la dirección anterior es de hecho la que genera el mayor descenso de la función objetivo.



Descenso por Gradiente (GD)

- La inmensa mayoría de los métodos para entrenar redes neuronales hoy en día son instancias de este método de optimización denominado **gradient descent**. A partir de una solución inicial $\Theta^{(0)}$, el método busca generar una sucesión de soluciones $\Theta^{(1)}, \Theta^{(2)}, \dots, \Theta^{(t)}$ convergente a un mínimo de la función objetivo. La sucesión se genera iterando la regla:

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta_t \nabla J(\Theta^{(t)})$$

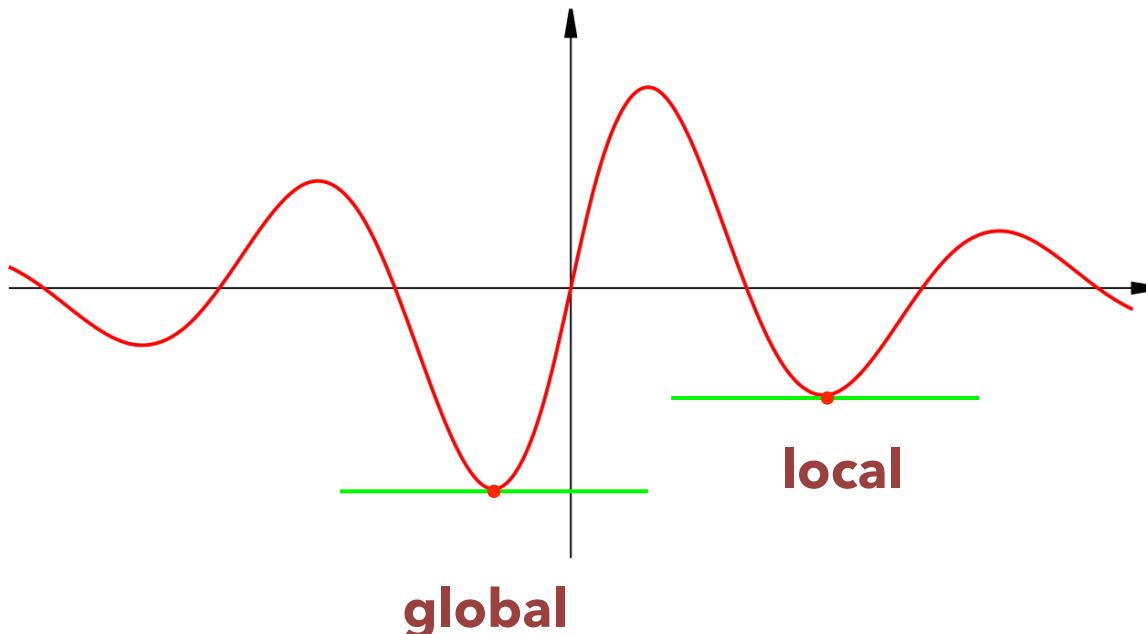


Mínimos Locales y Globales

- El algoritmo podrá, en el mejor de los casos, garantizar convergencia a un mínimo local de la función objetivo, es decir, una solución Θ^* tal que, para algún $\epsilon > 0$

$$J(\Theta^*) \leq J(\Theta) \quad \forall \|\Theta - \Theta^*\|^2 < \epsilon$$

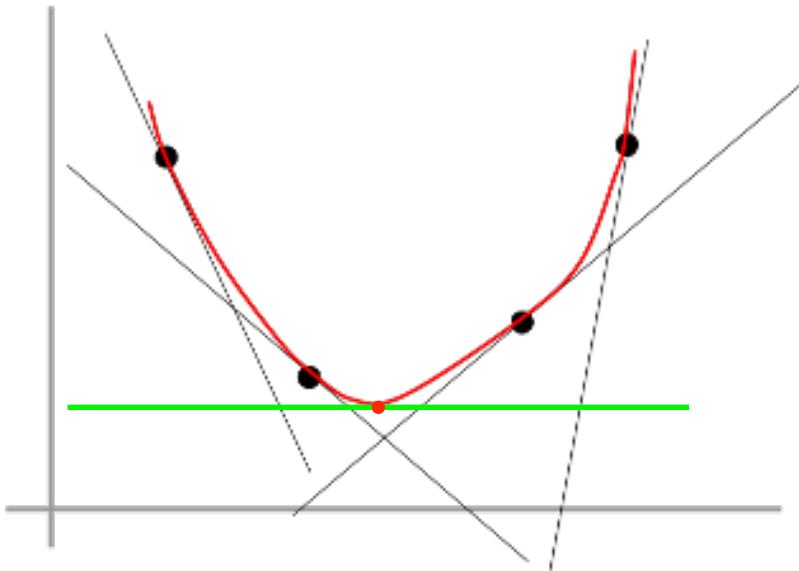
- Más precisamente, demostraremos que: $\|\nabla J(\Theta^{(t)})\|^2 \xrightarrow[t \rightarrow \infty]{} 0$



16

Mínimos Locales y Globales

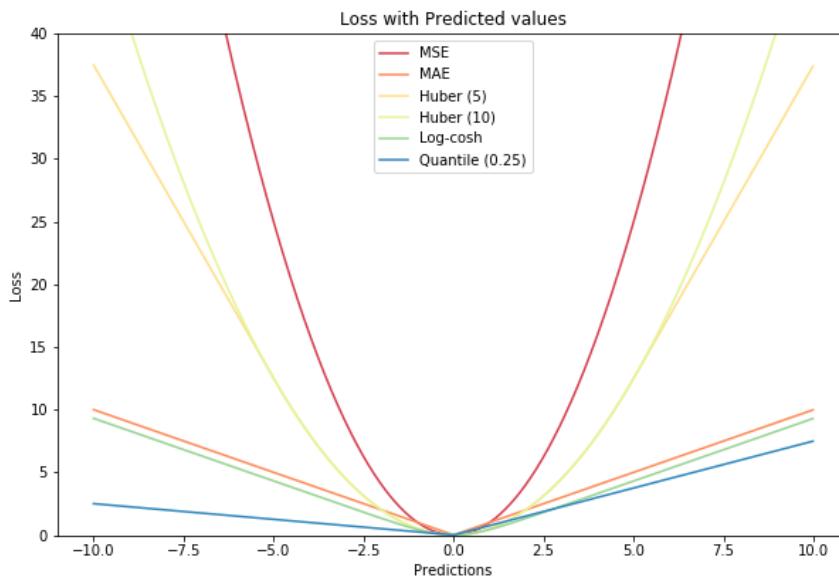
- Si la función objetivo de la red (e.g. error de entrenamiento) fuese **convexa** podríamos garantizar convergencia al mínimo global.



Mínimos Locales y Globales

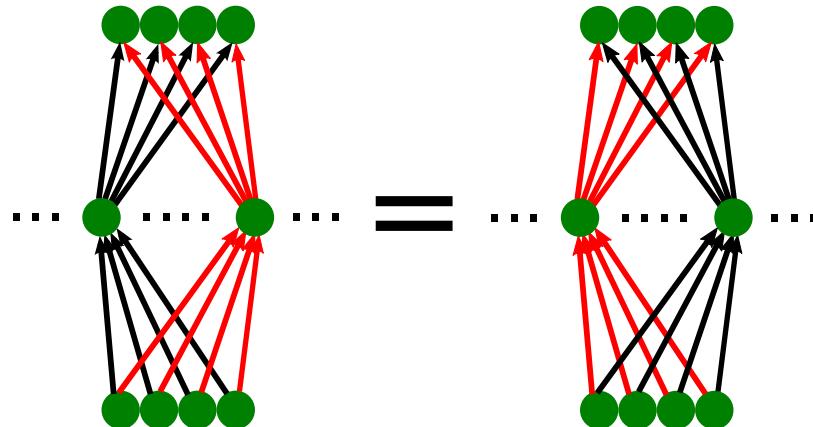
- Sin embargo, en redes con capas ocultas y funciones de activación no-lineales, la función objetivo será **altamente no convexa** como función de los parámetros de la red Θ , aún si la función de costo (loss) es convexa como función de $f(x; \Theta)$. Por ejemplo

$$L(f(x; \Theta), y) = (f(x; \Theta) - y)^2$$

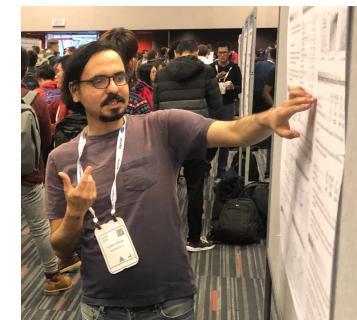


Mínimos Locales y Globales

- En redes muy profundas, la situación se muestra con mayor fuerza: existe un enorme número de puntos críticos (exponencialmente grande en la profundidad). Intuitivamente, esto ocurre porque un modelo profundo es **invariante** a un gran número transformaciones (permutaciones, escalamientos, etc).

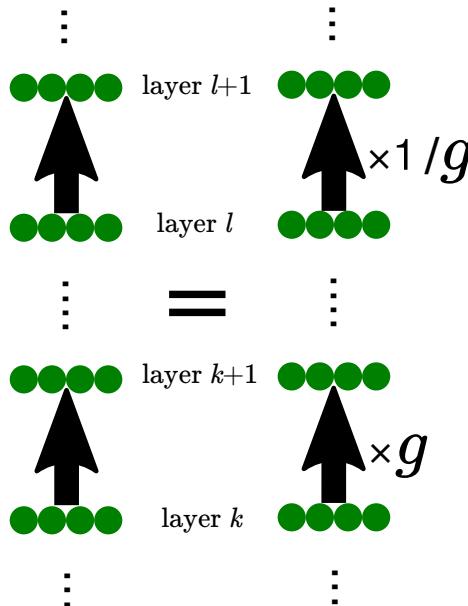


(*) Ver por ejemplo, "Skip Connections as Effective Symmetry-Breaking"
Emin Orhan, 2017.

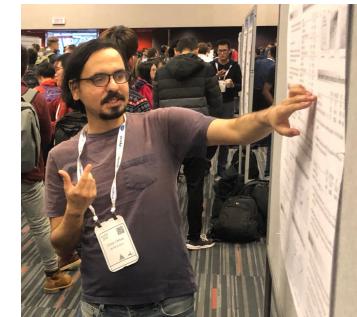


Mínimos Locales y Globales

- En otras palabras, existen múltiples formas de escribir la misma función (el **modelo no es identifiable** midiendo sólo el error de predicción).

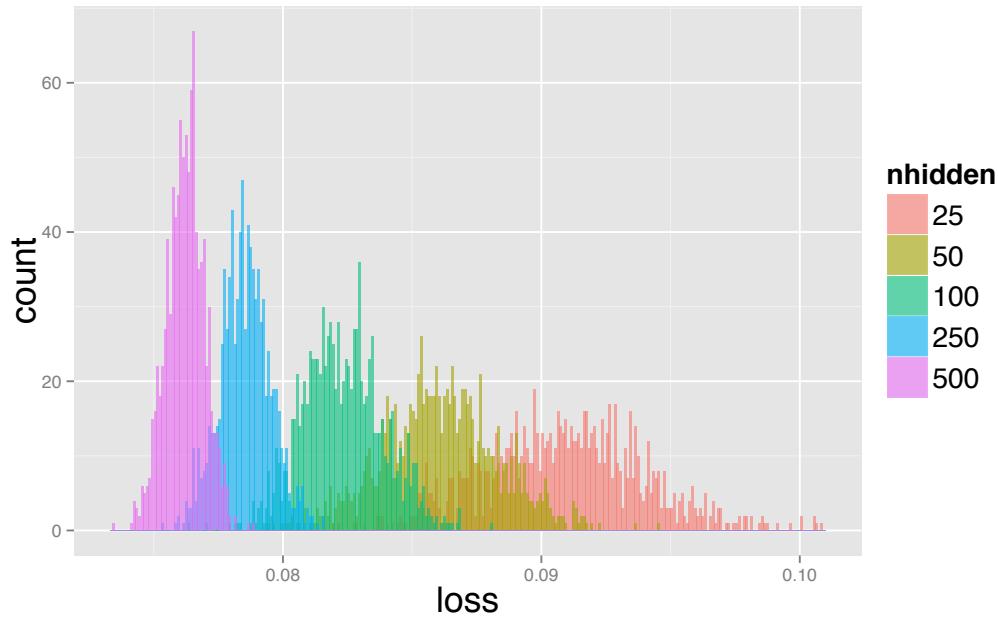


(*) Ver por ejemplo, "Skip Connections as Effective Symmetry-Breaking"
Emin Orhan, 2017.



Mínimos Locales y Globales

- Investigación reciente, demuestra en efecto que, en redes sobre-parametrizadas (con un alto grado de invarianzas, e.g. muy profundas o con muchos parámetros libres), el error de predicción (test) suele ser muy parecido para todos los mínimos locales de error de entrenamiento.



Distribution of the scaled test losses for neural networks with different number of hidden units. “*The Loss Surfaces of Multilayer Networks*”, Choromanska et al. 2015.



Mínimos Locales y Globales

- Es posible demostrar asumiendo condiciones bastante generales (entre las cuales diferenciabilidad y convexidad de las loss como función de $f(x; \Theta)$), que **todos los mínimos locales son globales**, aun si el objetivo de la red no es convexo en los parámetros.

ARTICLE ————— Communicated by Simon Shaolei Du

Every Local Minimum Value Is the Global Minimum Value of Induced Model in Nonconvex Machine Learning

Kenji Kawaguchi

kawaguch@mit.edu

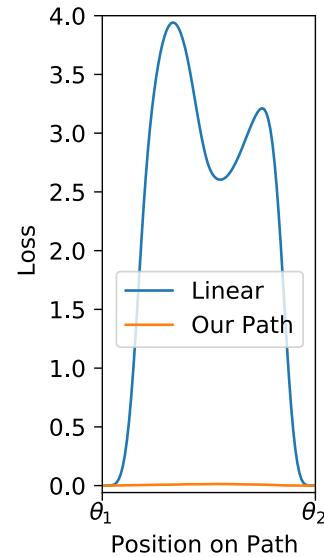
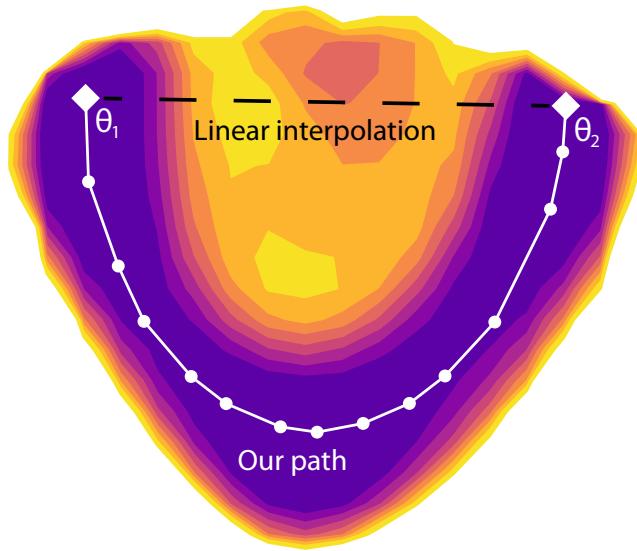
MIT, Cambridge, MA 02139, U.S.A.

Kawaguchi, Kenji, Jiaoyang Huang, and Leslie Pack Kaelbling. "Every Local Minimum Value Is the Global Minimum Value of Induced Model in Nonconvex Machine Learning." *Neural Computation* 31.12 (2019): 2293-2323.



Mínimos Locales y Globales

- Intuitivamente, sucede que los mínimos de la función objetivo no son puntos aislados en el espacio de búsqueda (como solemos verlos cuando miramos el valor del objetivo J), sino que forman un “sub-espacio conexo” de puntos: es posible “viajar” de un punto a otro sin empeorar significativamente el valor de J .



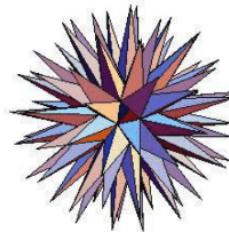
Draxler, Felix, et al. "Essentially No Barriers in Neural Network Energy Landscape." *International Conference on Machine Learning*. 2018.

Mínimos Locales y Globales

- Intuitivamente esto es posible porque en espacios de altísima dimensionalidad (Θ con muchas componentes debido a la sobre-parametrización de la red) el número de posibles caminos de un punto a otro es enorme: resulta fácil encontrar un camino que une los dos mínimos. El valor del objetivo no cambia significativamente en el camino porque debido su extrema versatilidad, la red puede “reparar” una perturbación local de la solución.

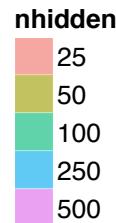
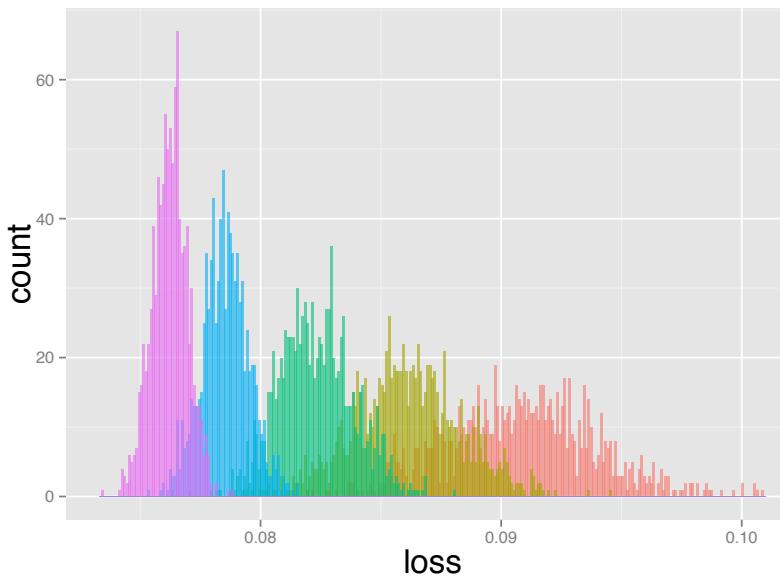
Curse of dimensionality

In R^d , $\exists \exp(d)$ directions
whose pairwise angle is > 60 degrees



Mínimos Locales y Globales

- **En resumen:** en redes modernas, optimizar con un algoritmo perezoso (greedy, primer orden) como GD tiene sentido porque (i) lo que nos interesa es el error de predicción y no la función objetivo de entrenamiento y (ii) porque existen muchas soluciones localmente óptimas que son buenas aproximaciones del óptimo global.

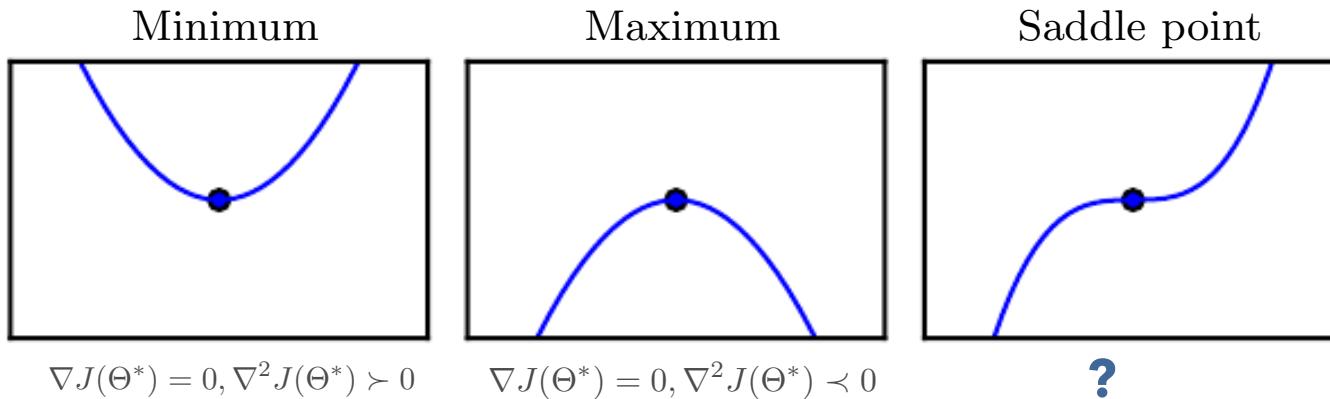


Distribution of the scaled test losses for neural networks with different number of hidden units. "The Loss Surfaces of Multilayer Networks", Choromanska et al. 2015.



Mínimos Locales y Globales

- En la práctica, tiende a ser más relevante la capacidad del algoritmo de entrenamiento de “escapar” de puntos denominados “puntos silla”:



- La probabilidad de encontrarse con uno de estos puntos es mucho más grande que la de encontrarse con uno donde todos los valores de la Hessiana d-dimensional tiene el mismo signo.

Tasa de Aprendizaje

- La capacidad del algoritmo de “escapar” de “puntos silla” dependerá fuertemente de la forma en que éste controle la **tasa**

Para $t=1,2 \dots$

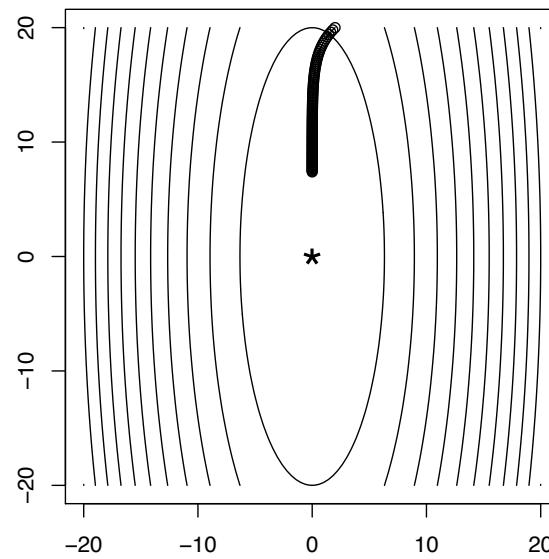
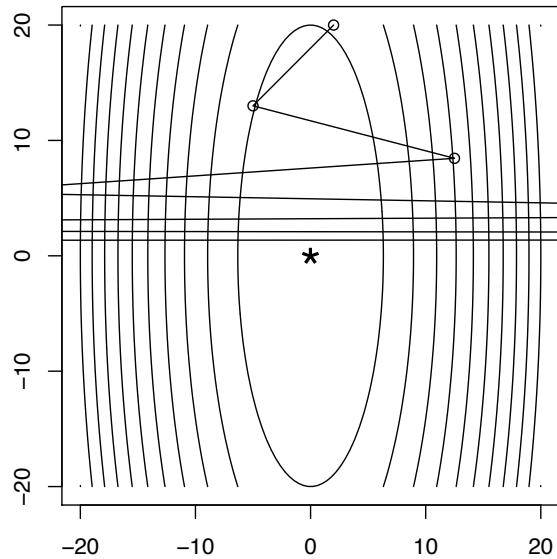
$$\Theta^{(t+1)} = \Theta^{(t)} - \eta_t \nabla J(\Theta^{(t)})$$

If (convergence) STOP.



Tasa de Aprendizaje

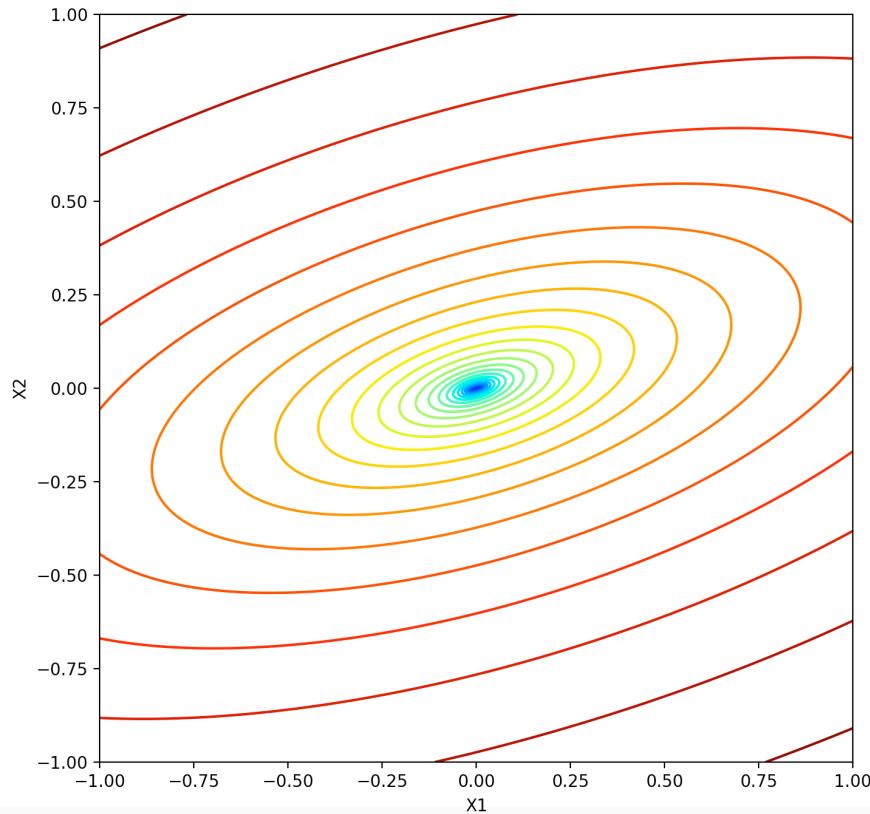
- En el capítulo 4, estudiaremos sofisticados modernos para calibrar la tasa (Adagrad, Adadelta, RMS-Prop, Adam). Por ahora diremos que debemos seleccionar un valor suficientemente pequeño como para no producir divergencia y uno suficientemente grande como para no estancarnos demasiado en una zona poco prometedora.



Ejemplo

- Ruteemos el algoritmo para optimizar (minimizar) la siguiente función de dos variables:

$$g(x, y) = \frac{1}{2}x^2 + 2y^2 - xy$$

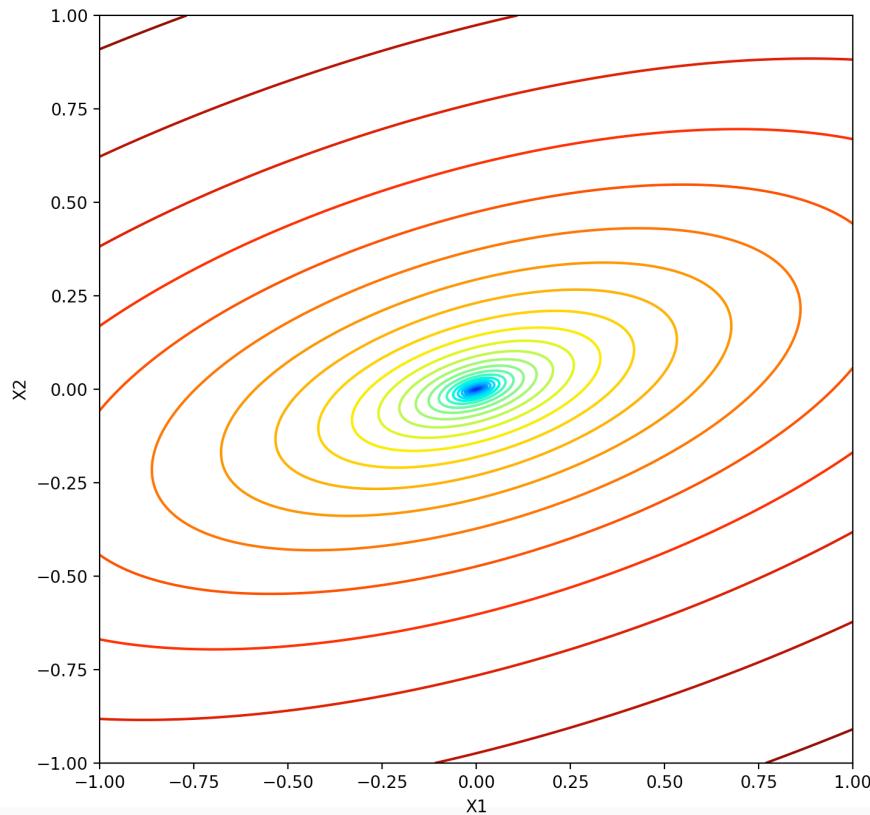


Las curvas representan soluciones (x, y) con el mismo valor del objetivo $g(x, y)$.

Ejemplo

- Ruteemos el algoritmo para optimizar (minimizar) la siguiente función de dos variables:

$$g(x, y) = \frac{1}{2}x^2 + 2y^2 - xy$$



Supongamos que iniciamos del punto $x = 1, y = -1$ y que usamos una tasa fija $\eta = 0.1$

Las derivadas de la función son

$$\partial g / \partial x = x - y$$

$$\partial g / \partial y = 4y - x$$

Ejemplo

- Como $g(x, y) = \frac{1}{2}x^2 + 2y^2 - xy$ y partimos de $x = 1$, $y = -1$, el valor inicial de objetivo es $g(1, -1) = 3.5$.
- Como $\partial g / \partial x = x - y$, $\partial g / \partial y = 4y - x$ y partimos de $x = 1$, $y = -1$, tenemos que el primer movimiento consiste en avanzar en la dirección:

$$-\nabla g(1, -1) = \begin{pmatrix} -2 \\ +5 \end{pmatrix}$$

- La nueva solución es por lo tanto $x = 0.8$, $y = -0.5$ con objetivo $g(0.8, -0.5) = 1.22!$

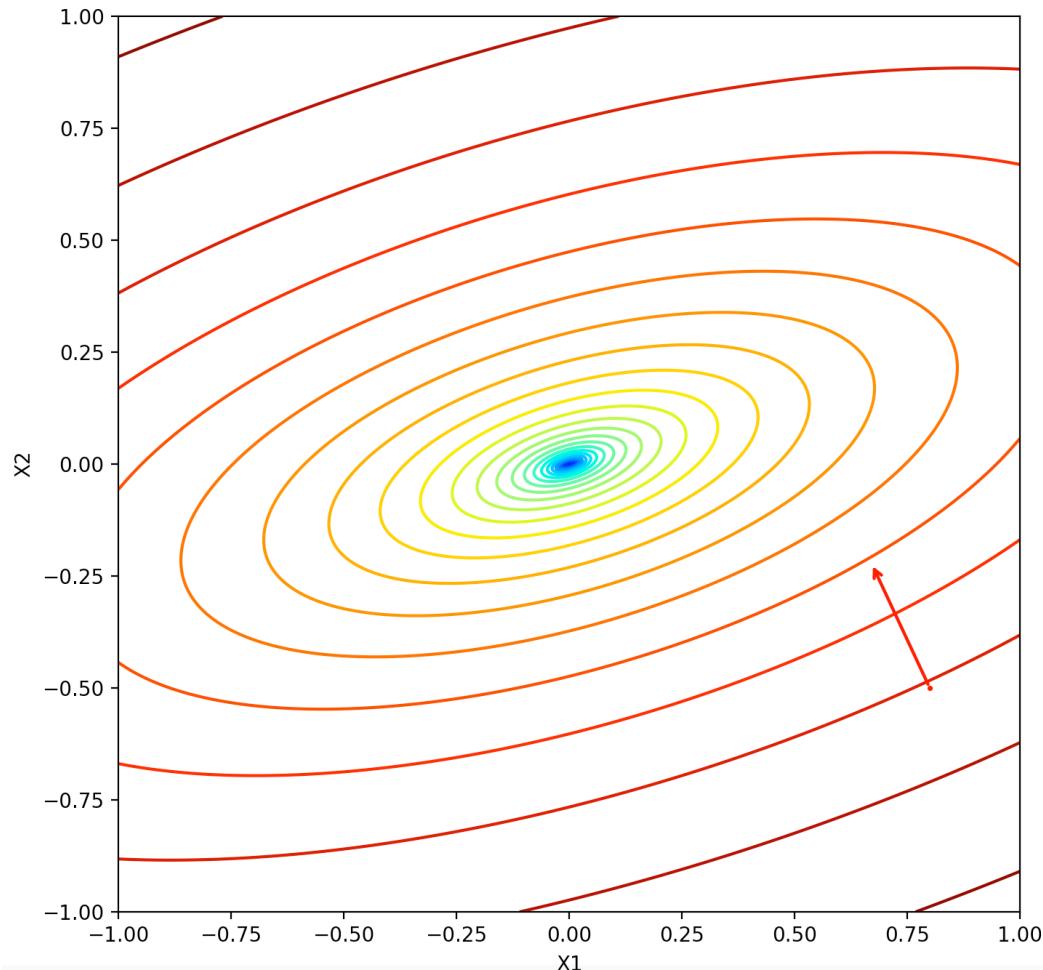
$$x \leftarrow x - \eta \partial g / \partial x = +1 - 0.2 = +0.8$$

$$y \leftarrow y - \eta \partial g / \partial y = -1 + 0.5 = -0.5$$



Ejemplo

- Primera iteración:



Ejemplo

- Como $g(x, y) = \frac{1}{2}x^2 + 2y^2 - xy$ y estamos en $x = 0.8$, $y = -0.5$, el valor de objetivo $g(0.8, -0.5) = 1.22$.
- Como $\partial g / \partial x = x - y$, $\partial g / \partial y = 4y - x$ y estamos en $x = 0.8$, $y = -0.5$, tenemos que el segundo movimiento consiste en avanzar en la dirección:

$$-\nabla g(0.8, -0.5) = \begin{pmatrix} -1.30 \\ +2.80 \end{pmatrix}$$

- La nueva solución es por lo tanto $x = 0.67$, $y = -0.22$ con objetivo $g(0.8, -0.5) = 0.47!$

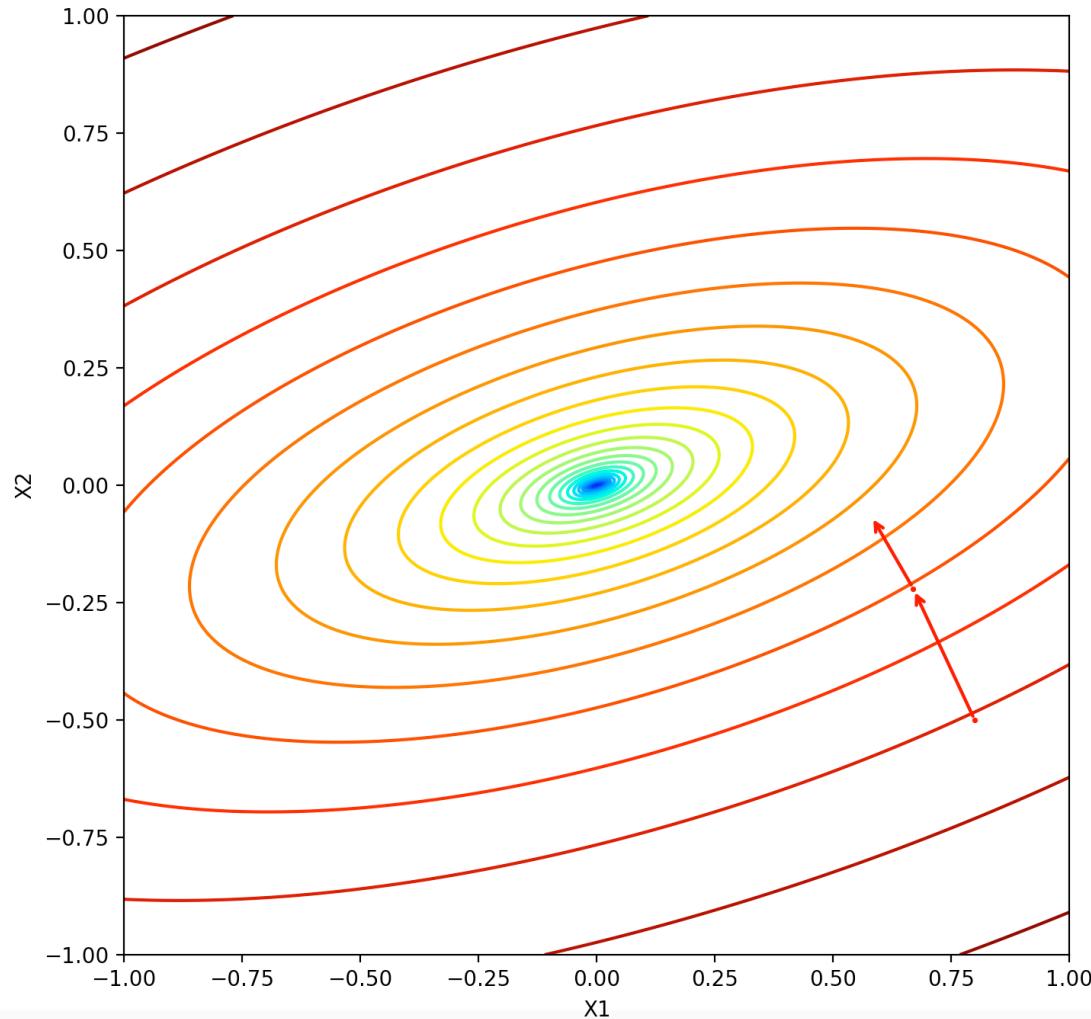
$$x \leftarrow x - \eta \partial g / \partial x = +0.8 - 0.13 = +0.67$$

$$y \leftarrow y - \eta \partial g / \partial y = -0.5 + 0.28 = -0.22$$



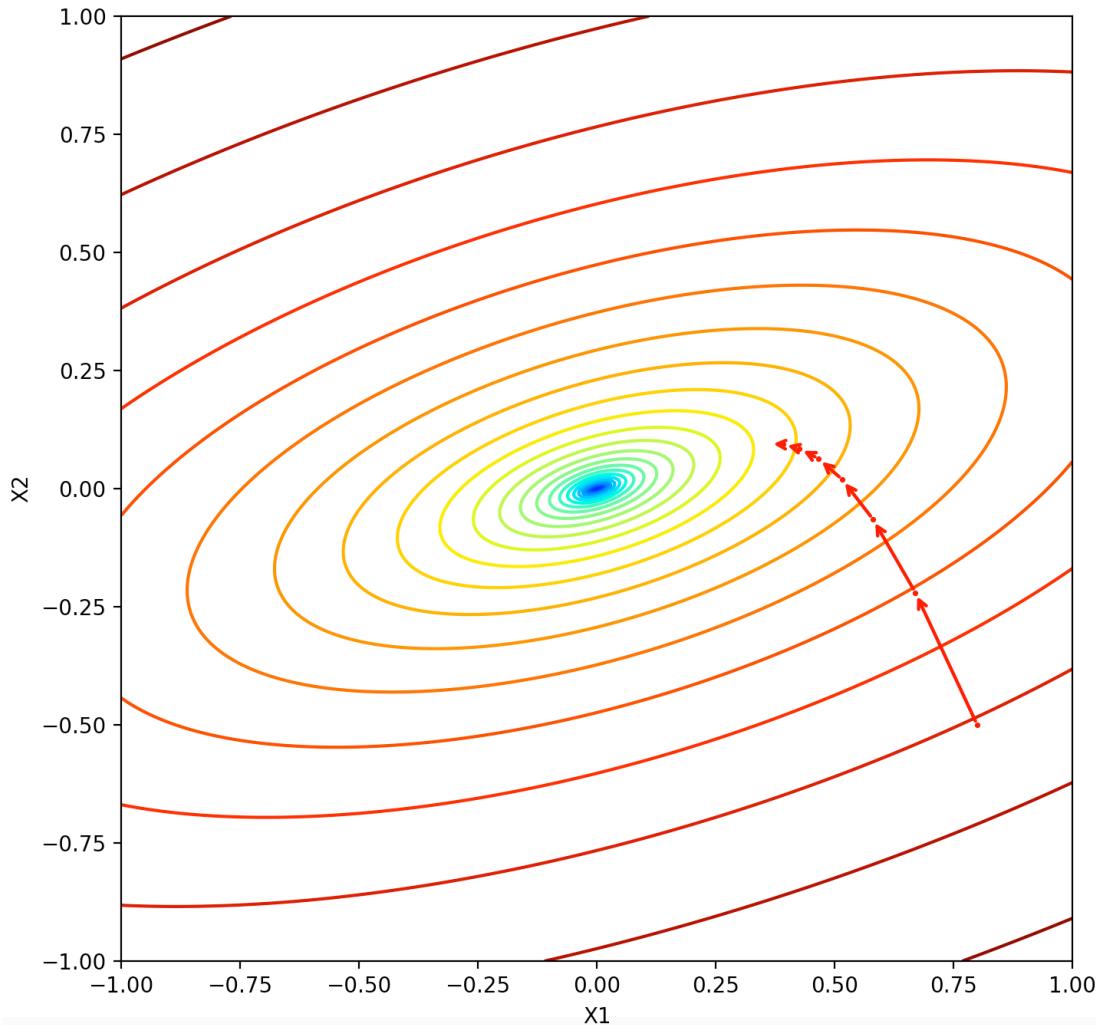
Ejemplo

- Primeras dos iteraciones:



Ejemplo

- Después de 8 iteraciones



$$x = 0.36, y = 0.1$$

$$g(x, y) = 0.2994$$

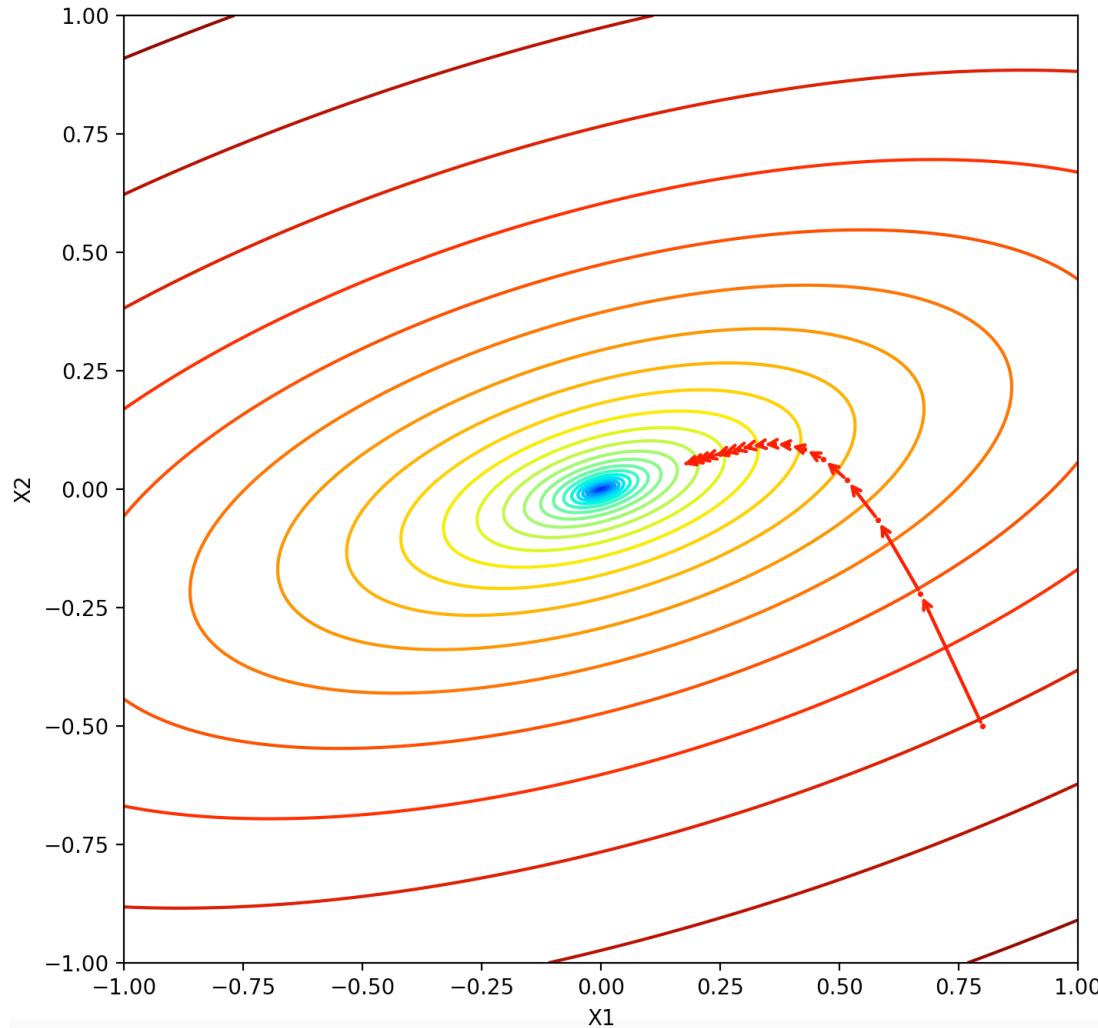
$$\|\nabla g\| = 0.0493$$

valor inicial del objetivo = 3.5



Ejemplo

- Después de 18 iteraciones



$$x = 0.17, y = 0.05$$

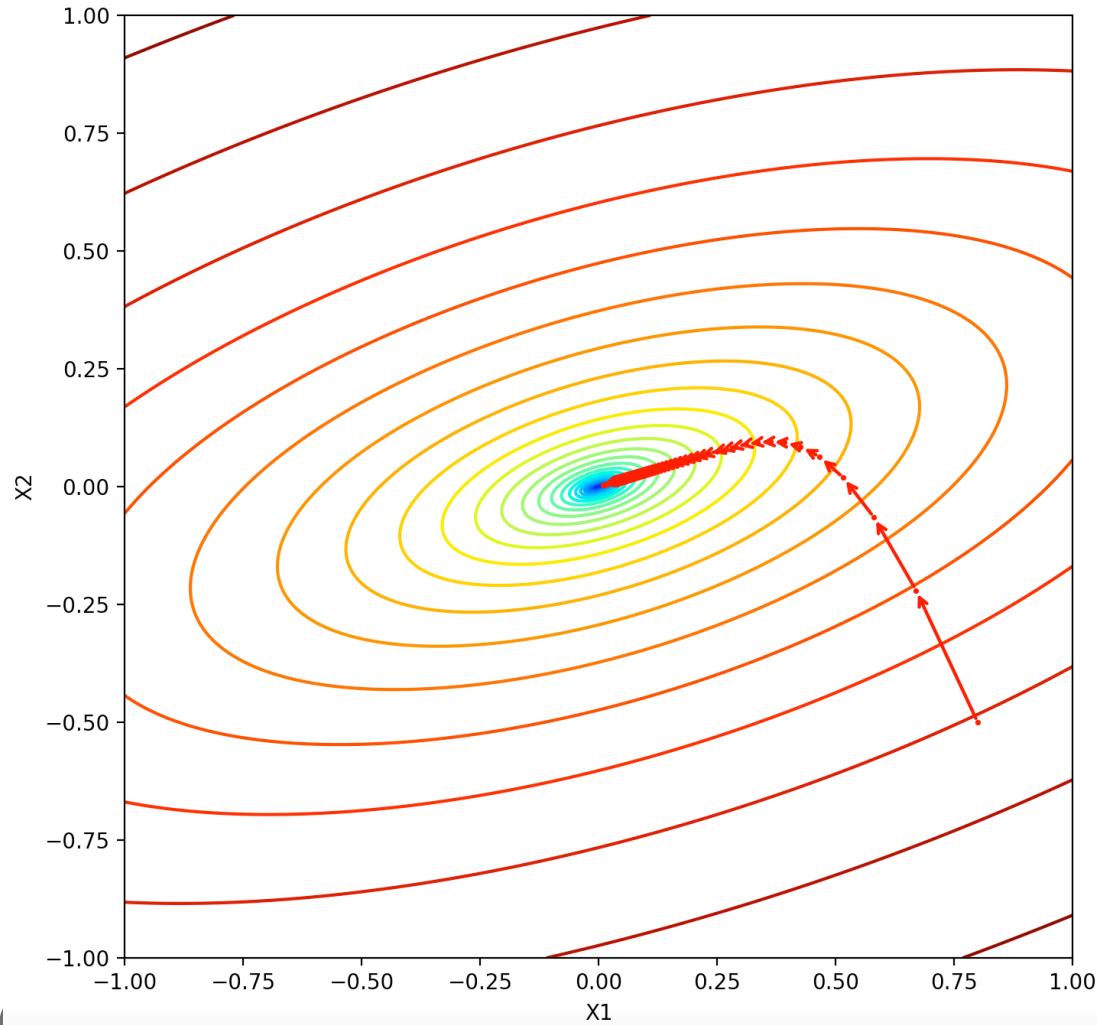
$$g(x, y) = 0.1362$$

$$\|\nabla g\| = 0.0115$$

valor inicial del objetivo = 3.5

Ejemplo

- Después de 48 iteraciones



$$x = 0.04, y = 0.01$$

$$g(x, y) = 0.0321$$

$$\|\nabla g\| = 0.0006$$

valor inicial del objetivo = 3.5

Convergencia

- Como hemos dicho, probaremos que el algoritmo converge en el sentido de que se verifica:

$$\|\nabla J(\Theta^{(t)})\|^2 \xrightarrow{t \rightarrow \infty} 0$$

- Esto sugiere que podemos detenernos monitorizando la magnitud del gradiente. Si la tasa es pequeña y constante (ver argumento sobre la dirección de descenso)

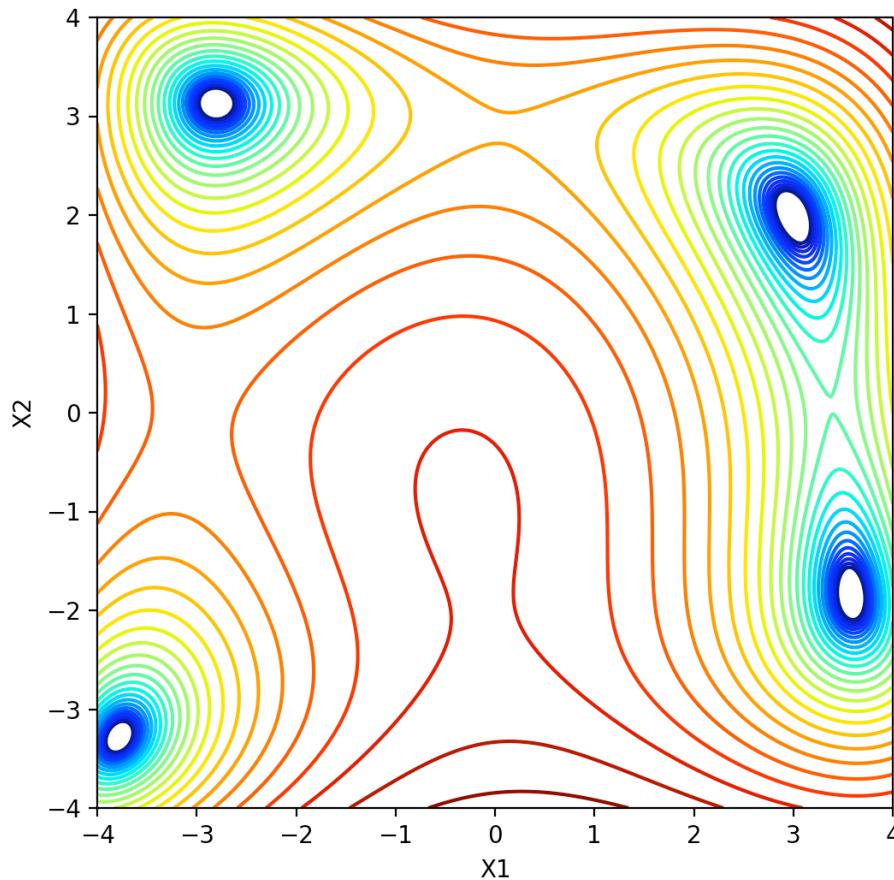
$$J(\Theta^{(t)}) - J(\Theta^{(t+1)}) < 2\eta \|\nabla J(\Theta^{(t+1)})\|^2$$

- En la práctica es común detener el algoritmo después de un número máximo de iteraciones o *epochs*.

Otro ejemplo

- Tarea: replique el experimento anterior usando una función más interesante

$$g(x, y) = ((x^2 + y - 11)^2 + (x + y^2 - 7)^2)$$

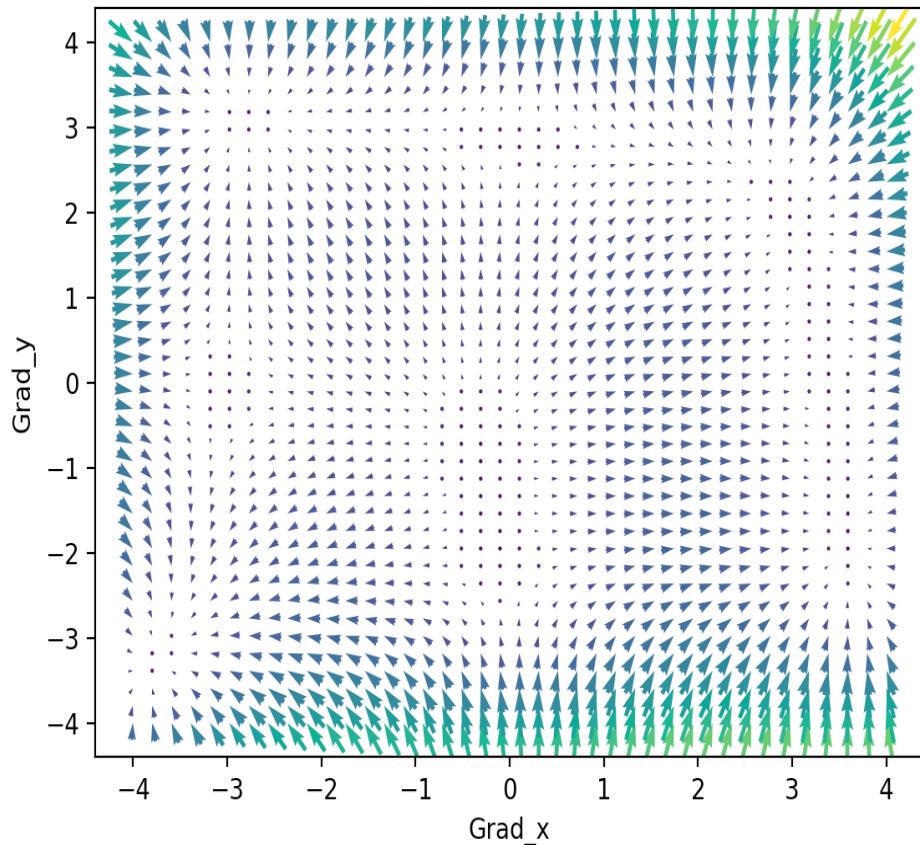


existen 4 mínimos (la función no es convexa) y puntos silla bastante evidentes

Otro ejemplo

- Tarea: replique el experimento anterior usando una función más interesante

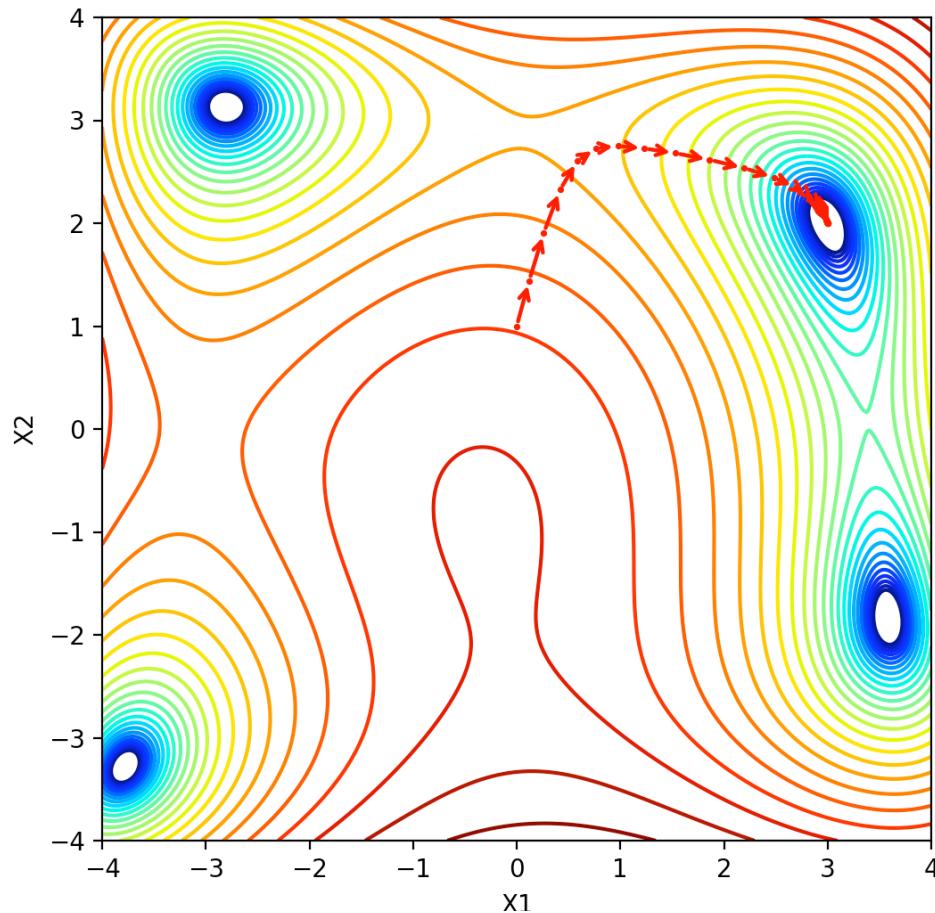
$$g(x, y) = ((x^2 + y - 11)^2 + (x + y^2 - 7)^2)$$



existen 4 mínimos (la función no es convexa) y puntos silla bastante evidentes

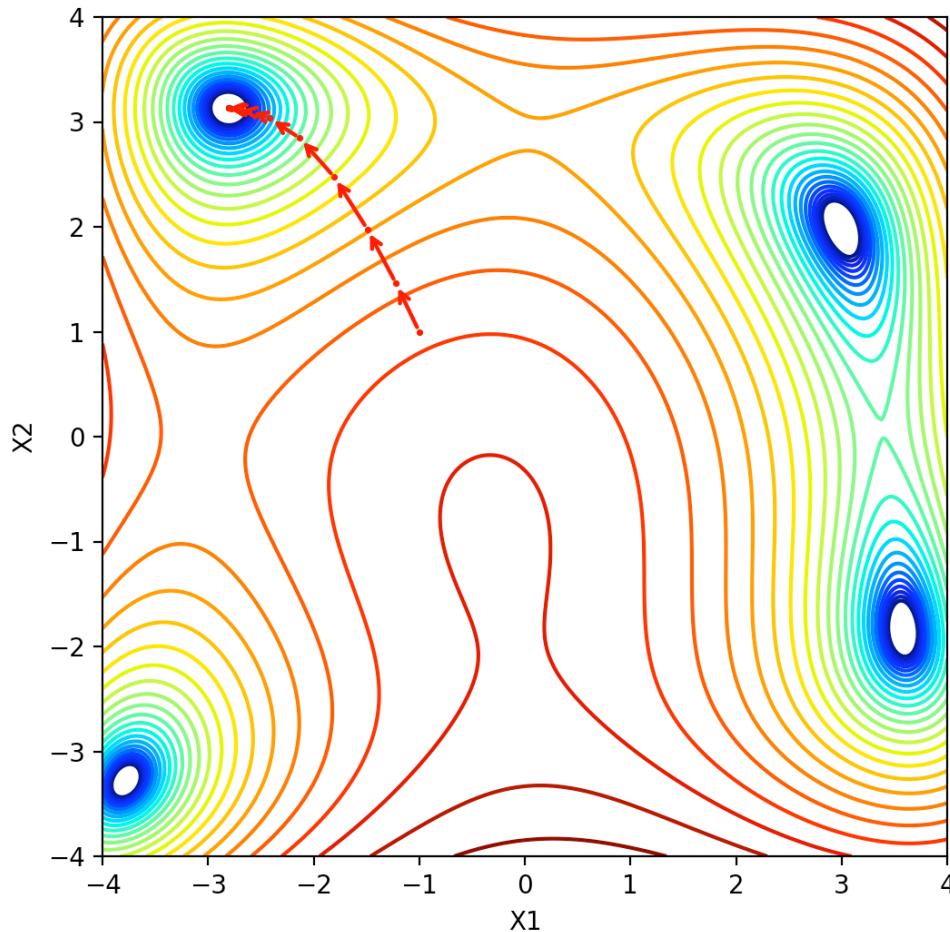
Otro ejemplo

- Resultado después de 100 iteraciones partiendo desde $x = 0, y = 1$



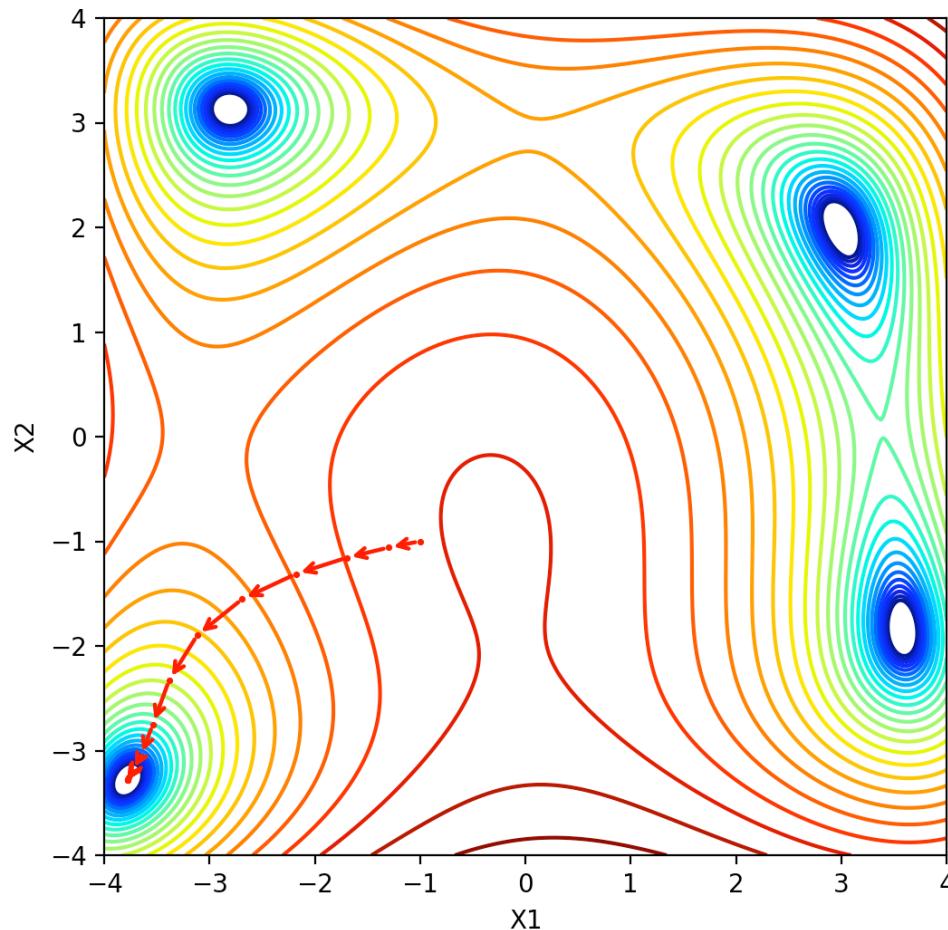
Otro ejemplo

- Resultado después de 100 iteraciones partiendo desde $x = -1, y = 1$



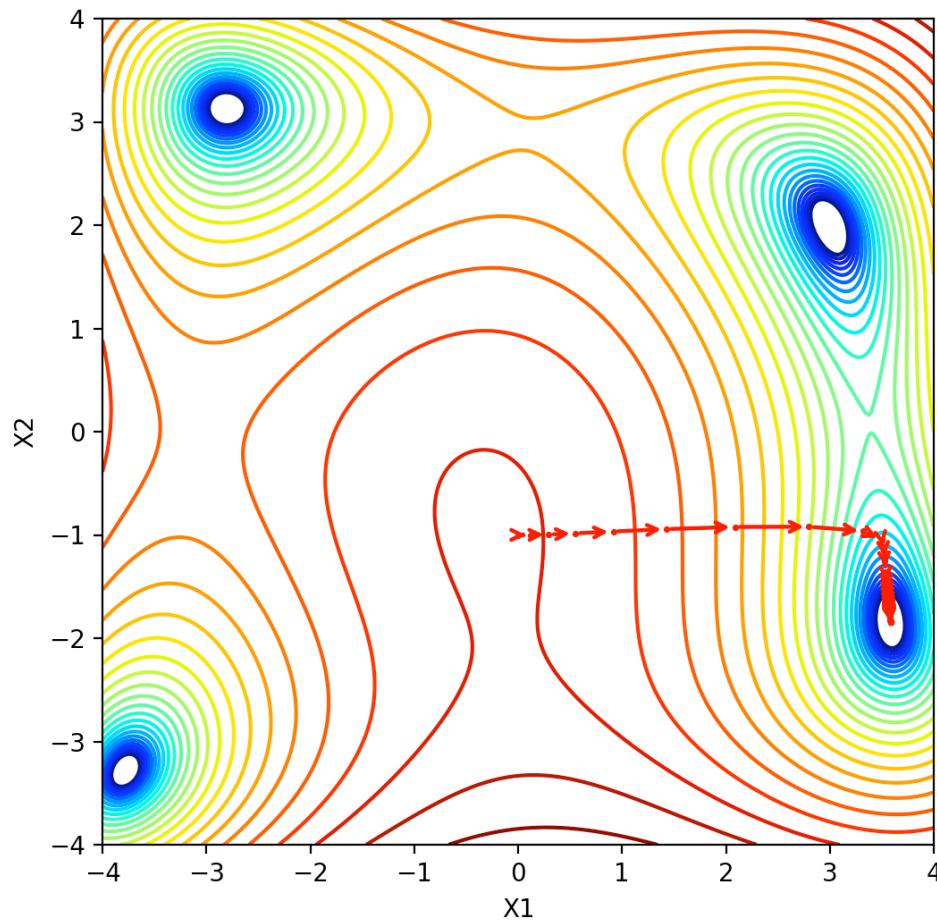
Otro ejemplo

- Resultado después de 100 iteraciones partiendo desde $x = -1, y = -1$



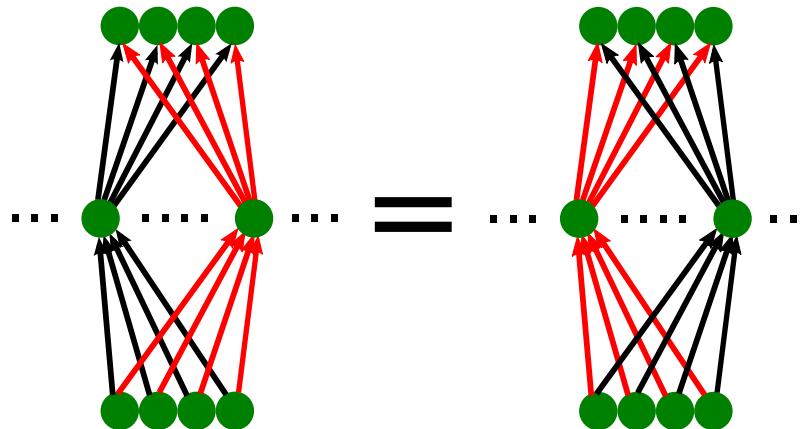
Otro ejemplo

- Resultado después de 100 iteraciones partiendo desde $x = 0, y = -1$



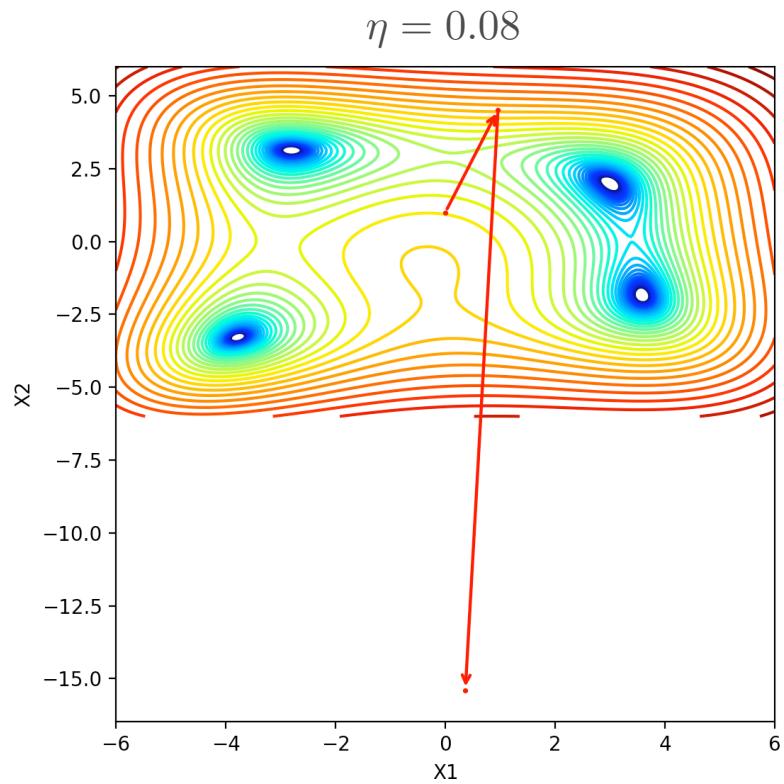
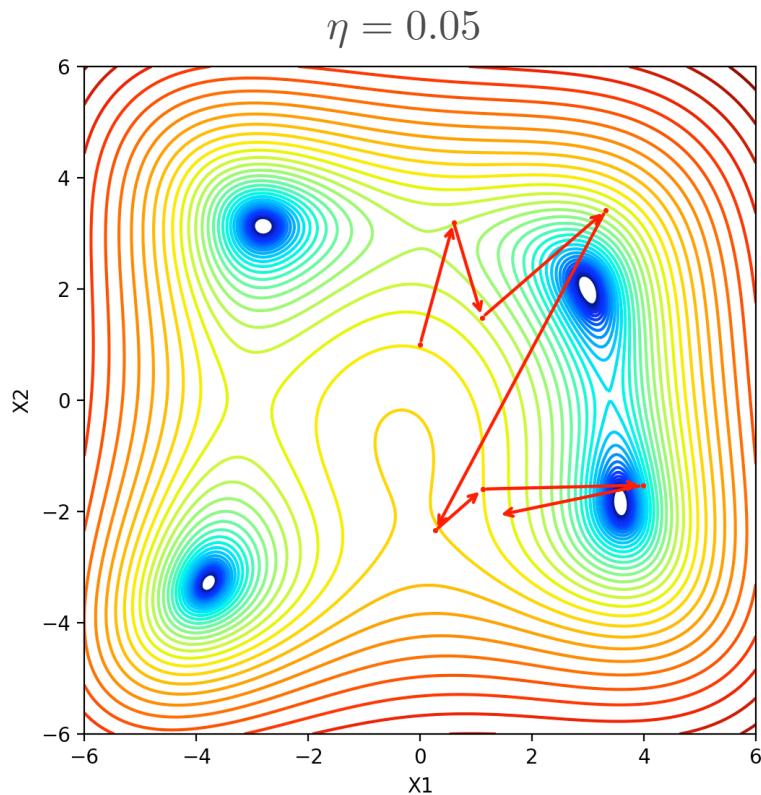
Convergencia

- Como hemos dicho, en el caso de funciones no convexas, el algoritmo converge a un mínimo que depende del punto de partida.
- En la práctica las redes se inicializarán de modo aleatorio para romper posibles simetrías entre las unidades ocultas (symmetry breaking)



Convergencia

- El experimento anterior permite comprobar la necesidad de calibrar apropiadamente la tasa de aprendizaje. Se observa divergencia con mínimos cambios alrededor de la tasa utilizada



Entonces ...

- Es posible entrenar los parámetros de una red usando como señales de error (crédito) las derivadas parciales de la función objetivo. Eso se traduce en iterar la siguiente regla de actualización hasta convergencia

$$\Theta_j \leftarrow \Theta_j - \eta_t \frac{\partial J}{\partial \Theta_j}$$

- La convergencia depende de la elección de la tasa de aprendizaje η_t y en general se producirá a un punto estacionario que puede ser un mínimo local o un punto silla.
- En redes modernas, cualquier mínimo local suele ser una solución tan buena como el mínimo global y es más relevante preocuparse de la capacidad del algoritmo de escapar de puntos silla y/o del objetivo real del aprendizaje: el error de predicción (test).