UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Sesión 1.0

Modelos

OLS vs MLP

TRANSFORMATEC

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Hola

Soy Victor :D

TRANSFORMATEC

Evaluación

Teoría

Martes 6-10pm (A708)

- Quizzes al inicio de las clases (10 minutos)

Laboratorio

Miércoles 6-10pm (L507)

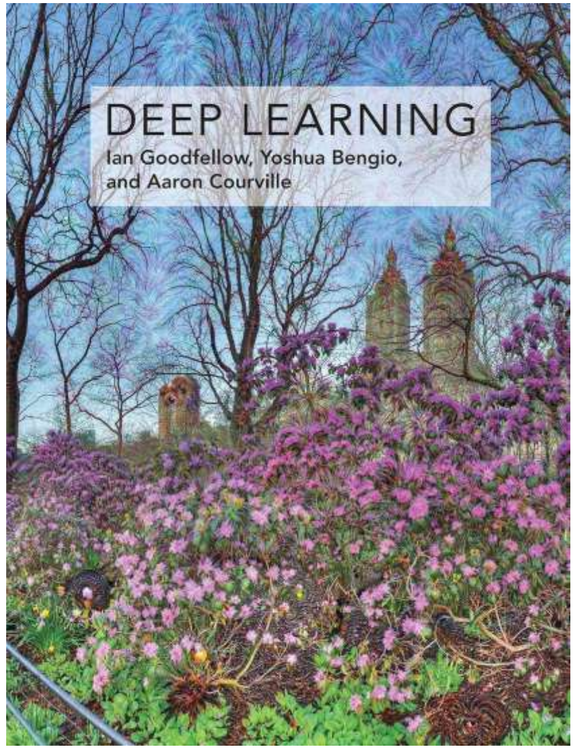
Jueves 6-10pm (L507)

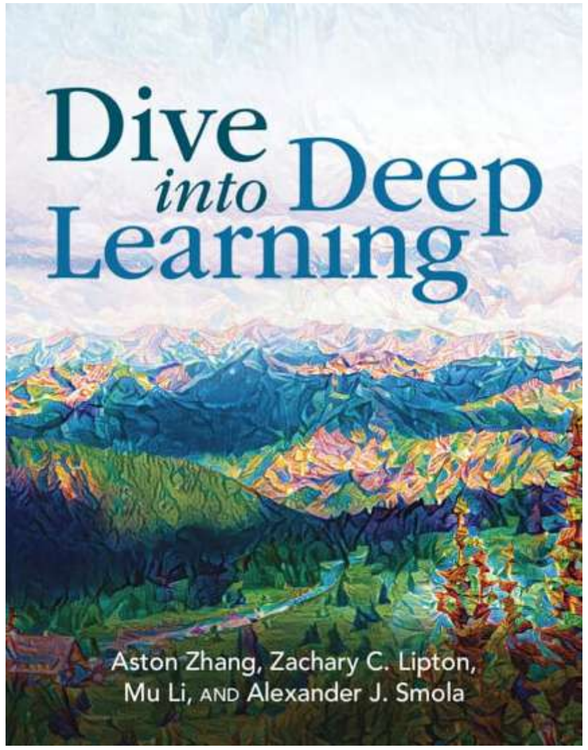
- Laboratorios individuales
- Proyecto grupal

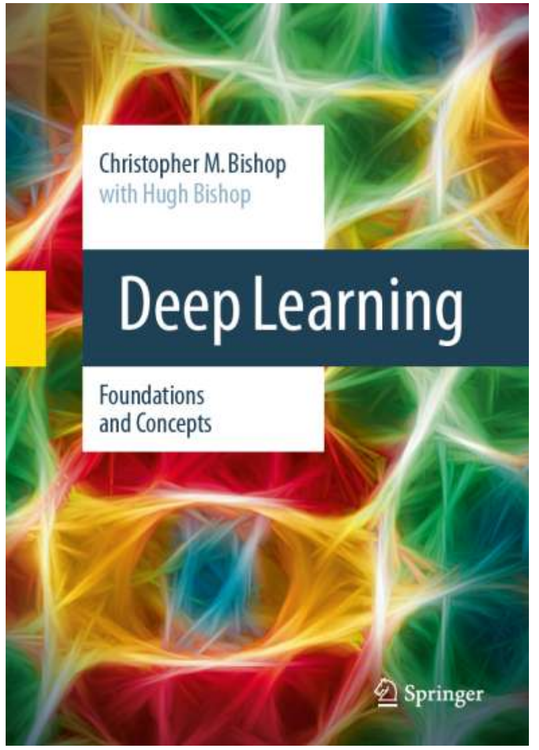
Fechas *importantes*

Quizzes	**10%**	2pts
Examen Parcial	**10%**	2pts
Examen Final	**10%**	2pts
Laboratorios semanales (4)	**40%**	8pts
Proyecto parcial	**15%**	3pts
Proyecto final	**15%**	3pts

Biblio*grafía*







Discord

https://discord.gg/XV8qzEb5

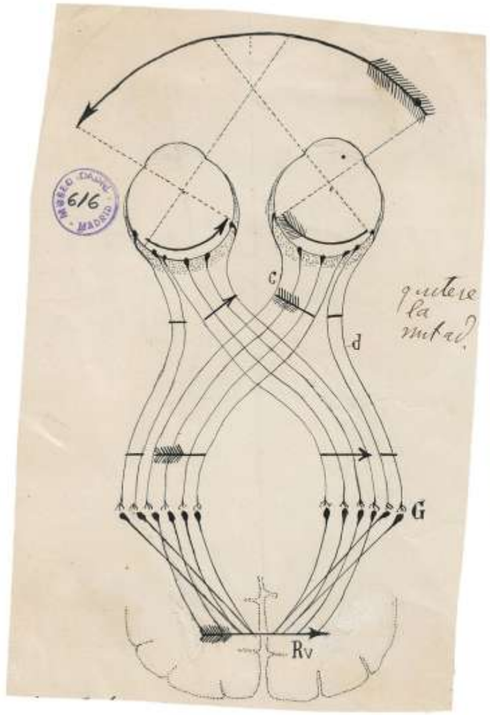
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

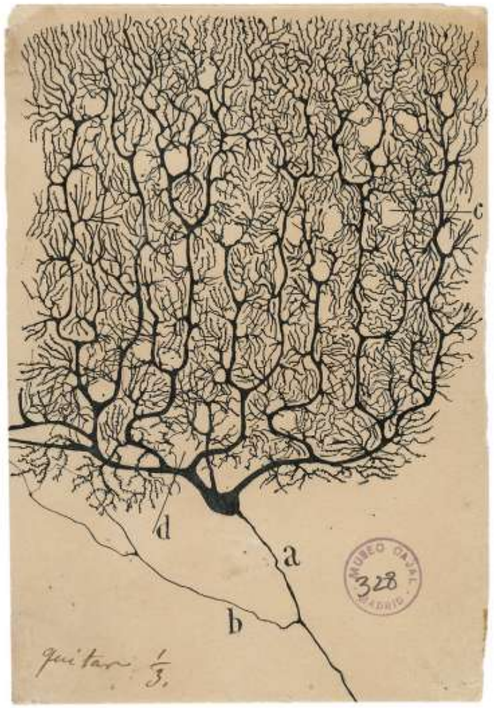
1.

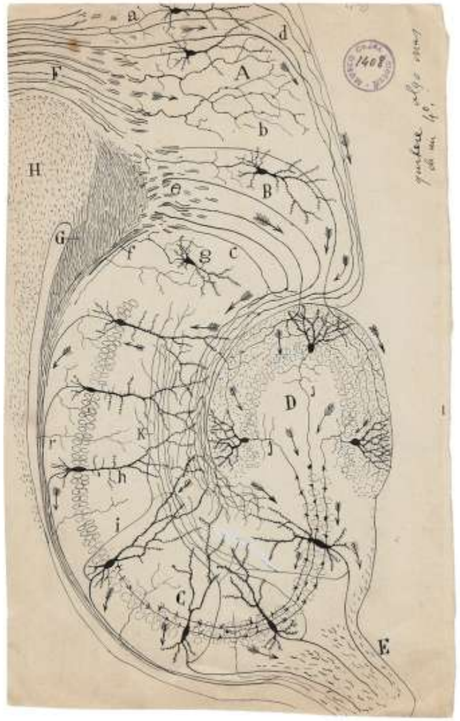
A Brief *history*

TRANSFORMATEC

Neuronas *biológicas*

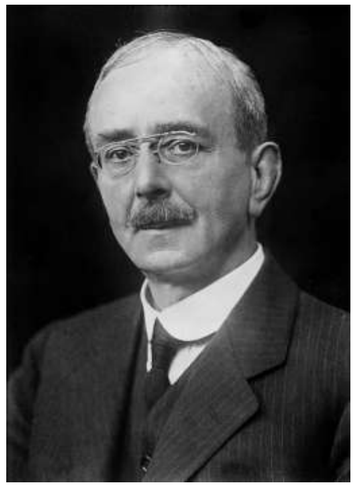




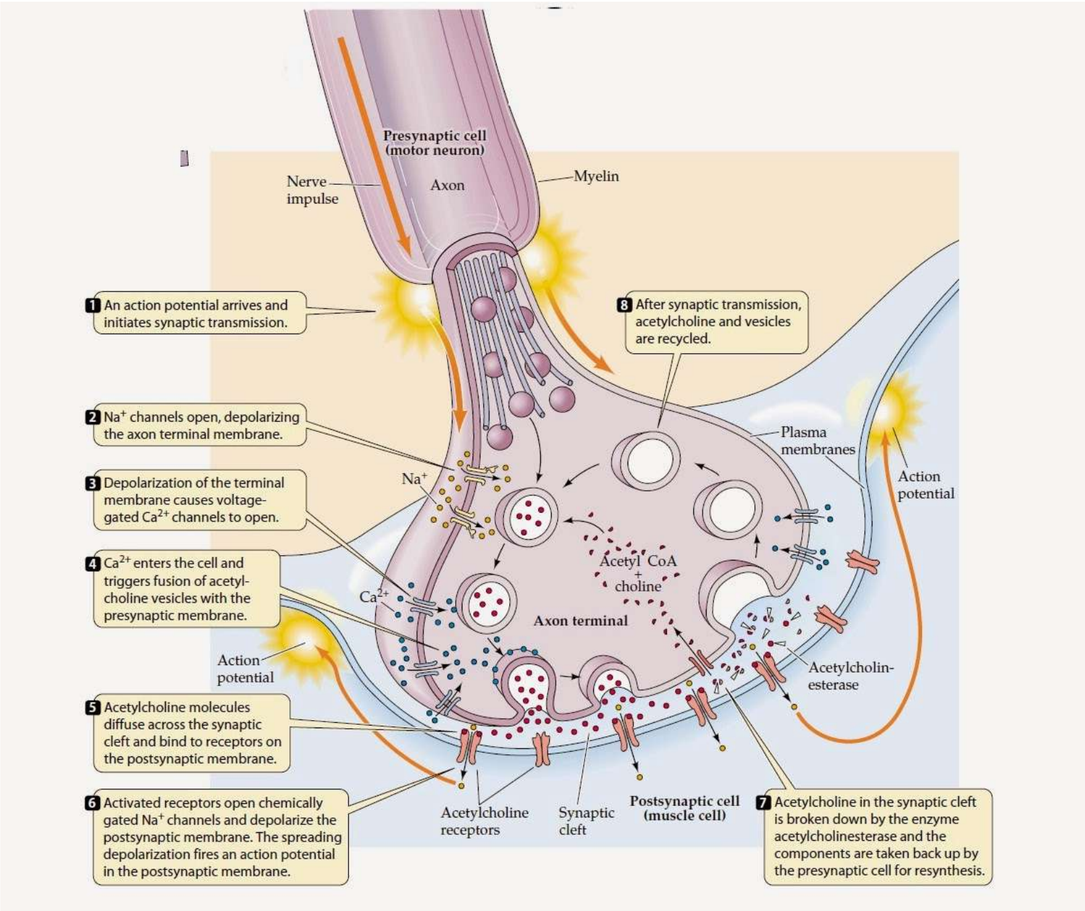


Santiago ramon y cajal (1890s)

Neuronas *biológicas*



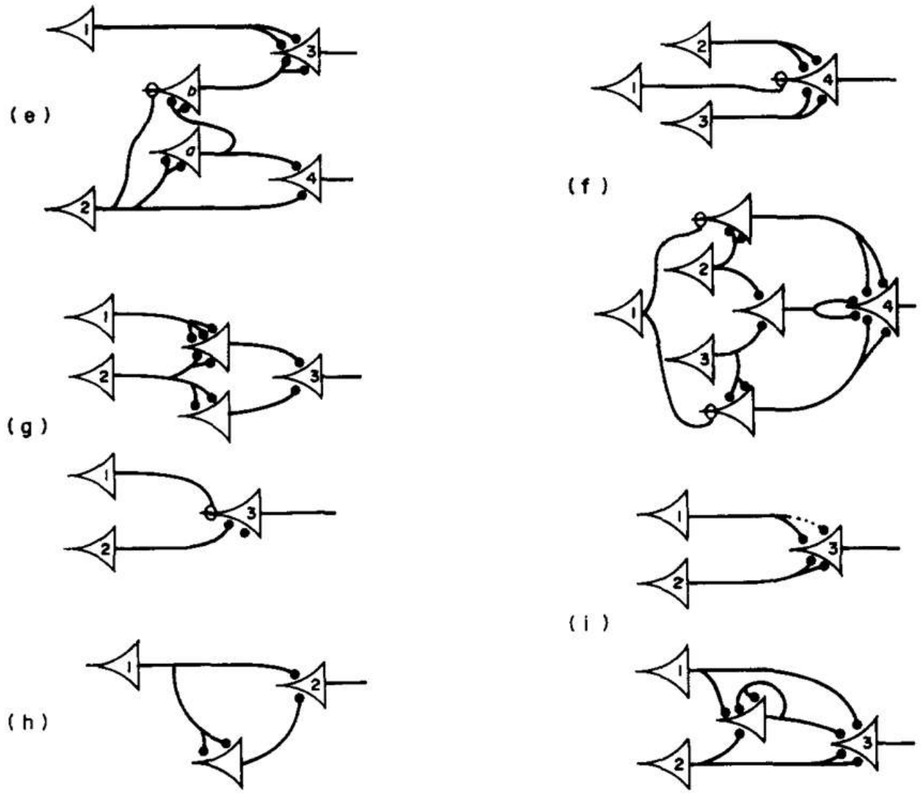
Charles Scott Sherrington (1920s)

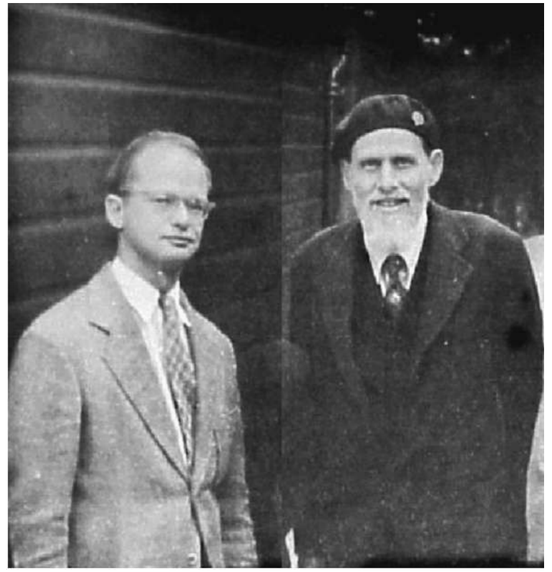


Gordon M. Shepherd and Solomon D. Erulkar (1997) "Centenary of the synapse: from Sherrington to the molecular biology of the synapse and beyond". Trends in neurosciences, 20(9), 385-392.

Neuronas *biológicas*

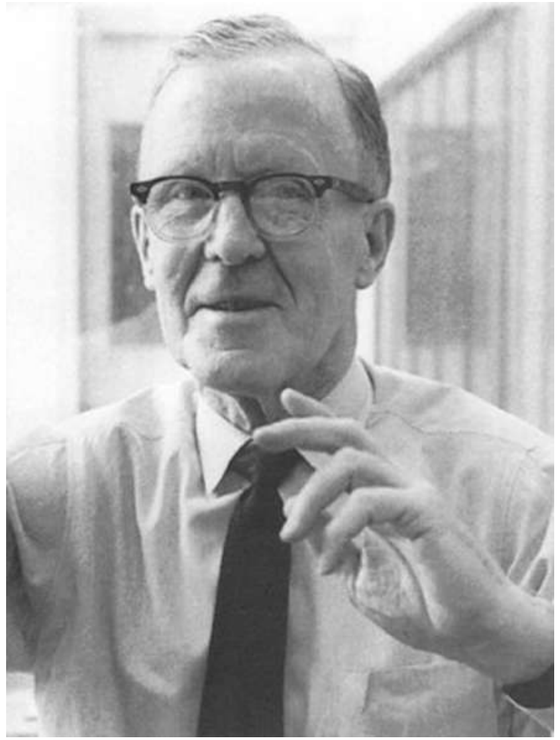
(Threshold Logic Unit (TLU))



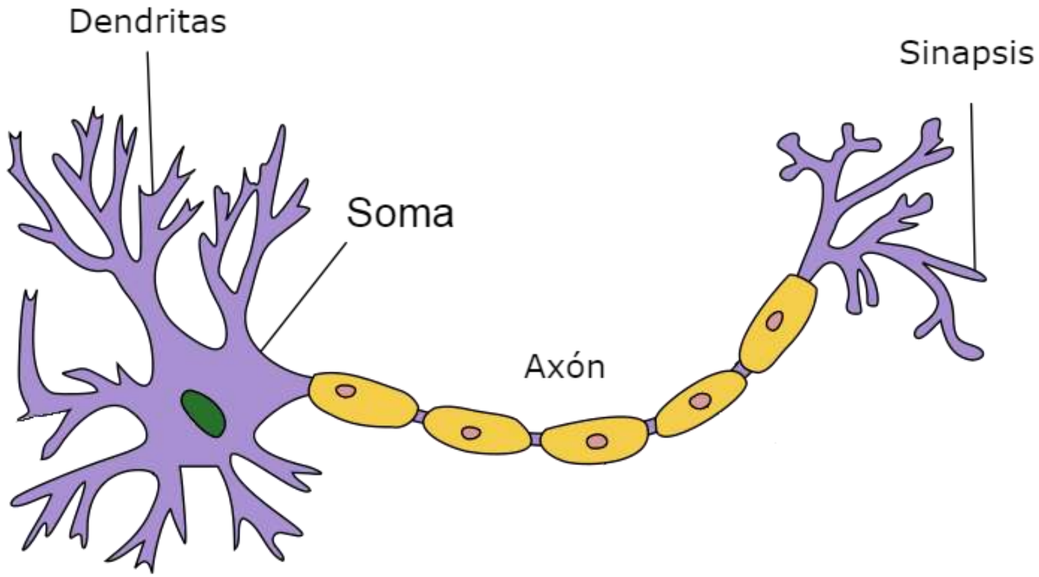


McCulloch & Pitts (1943)

Neuronas *biológicas*



Donald Hebb (1949)

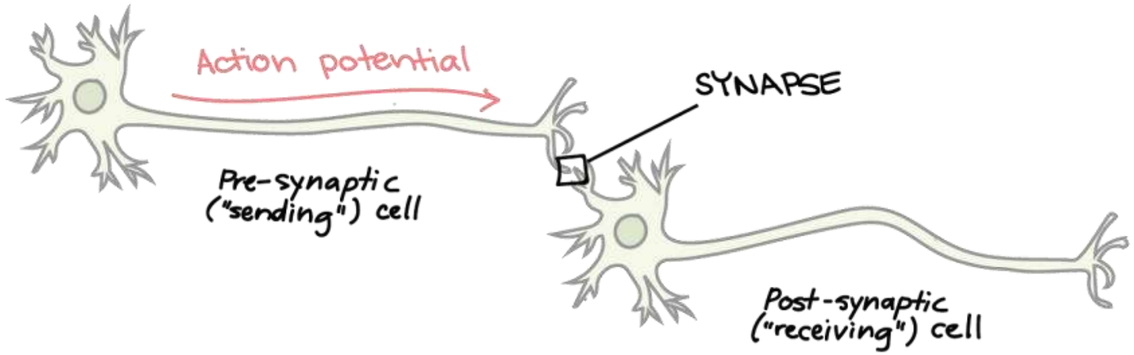


Regla de Hebb

"Las neuronas que se activan juntas tienden a conectarse más fuertemente entre sí."

Regla *de Hebb*

"Cuando una neurona presináptica contribuye repetidamente a la activación de una neurona postsináptica, la conexión sináptica entre ambas se fortalece."



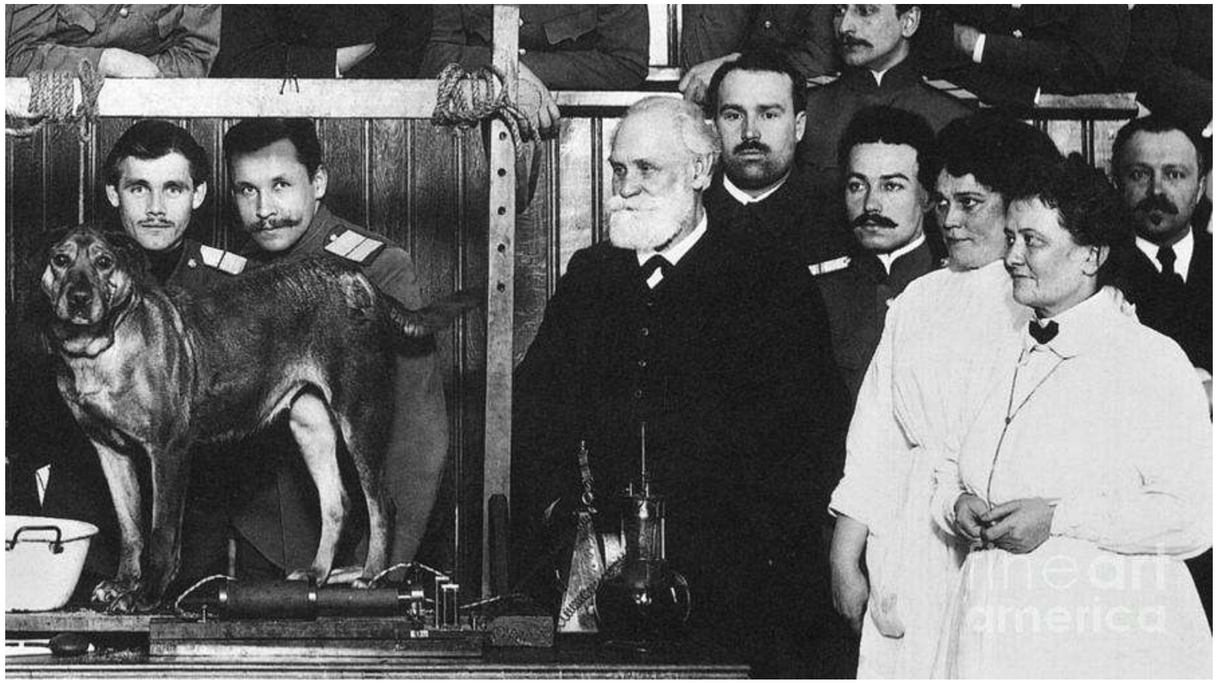
$$\Delta w_{ij} = \eta x_i y_j$$

donde:

- Δw_{ij} es el cambio en el peso sináptico.
- η es la tasa de aprendizaje (un valor positivo que regula la magnitud del cambio).
- x_i es la actividad de la neurona presináptica i.
- y_j es la actividad de la neurona postsináptica j.

Apredizaje

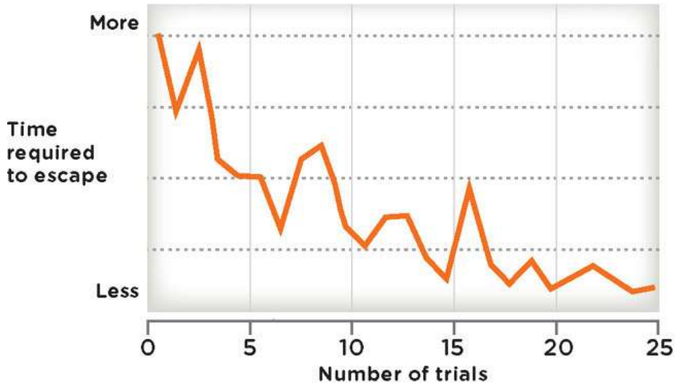
(condicionamiento clásico)



Ivan Pavlov (1897)

Apredizaje

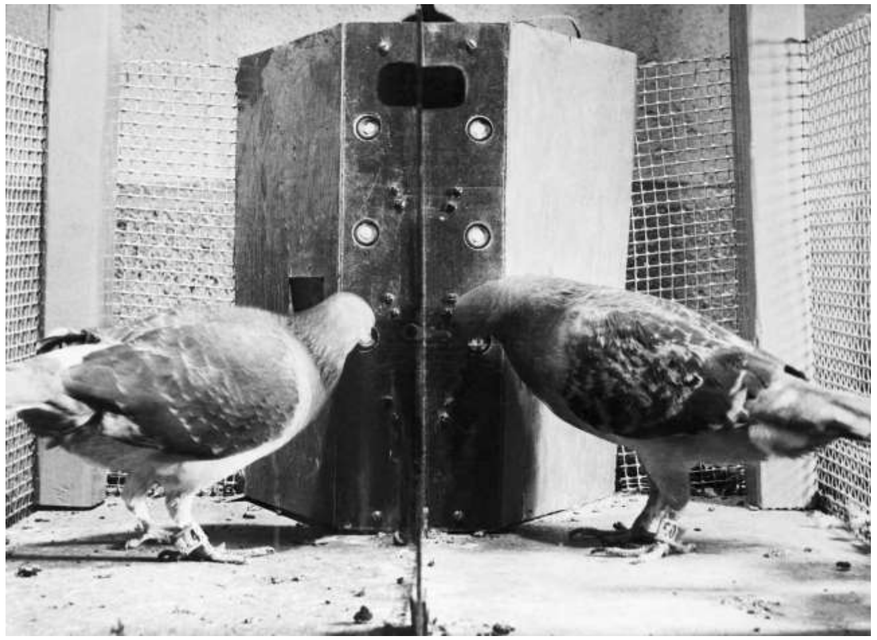
(Ley del Efecto)



Edward Thorndike (1898)

Apredizaje

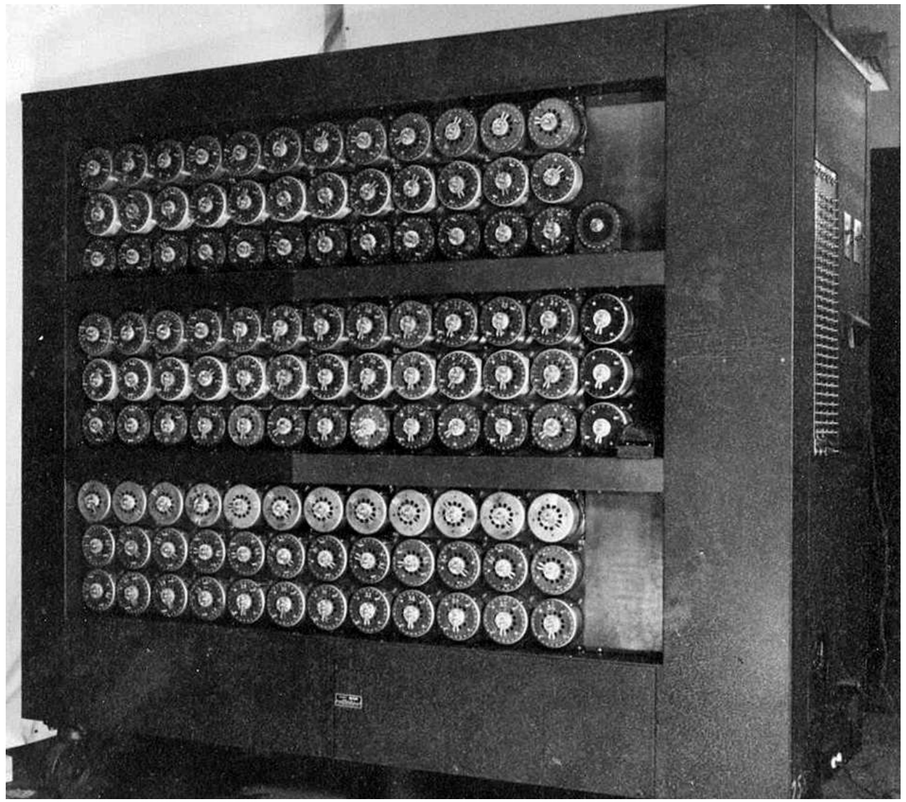
(condicionamiento operante)



Burrhus Frederic Skinner (1948)

Inteligencia *Artificial*

Turing machine



Alan Turing (1950)

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Inteligencia *Artificial*

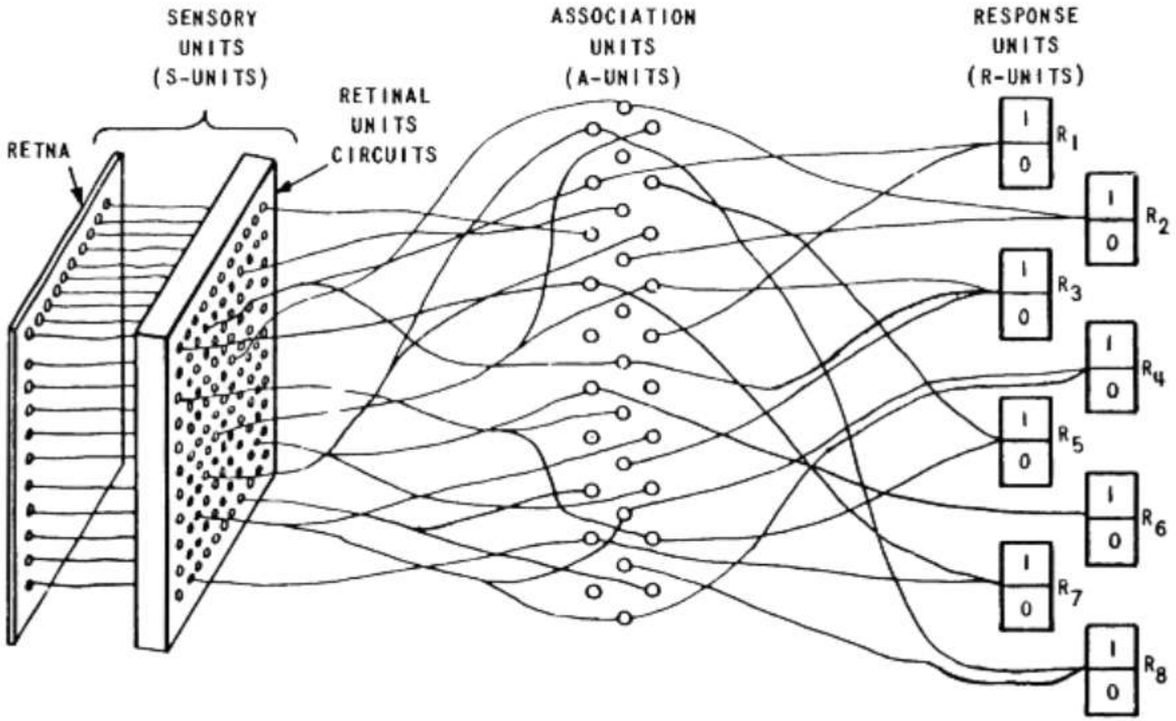
(Turing machine)

TRANSFORMATEC

Inteligencia *Artificial*

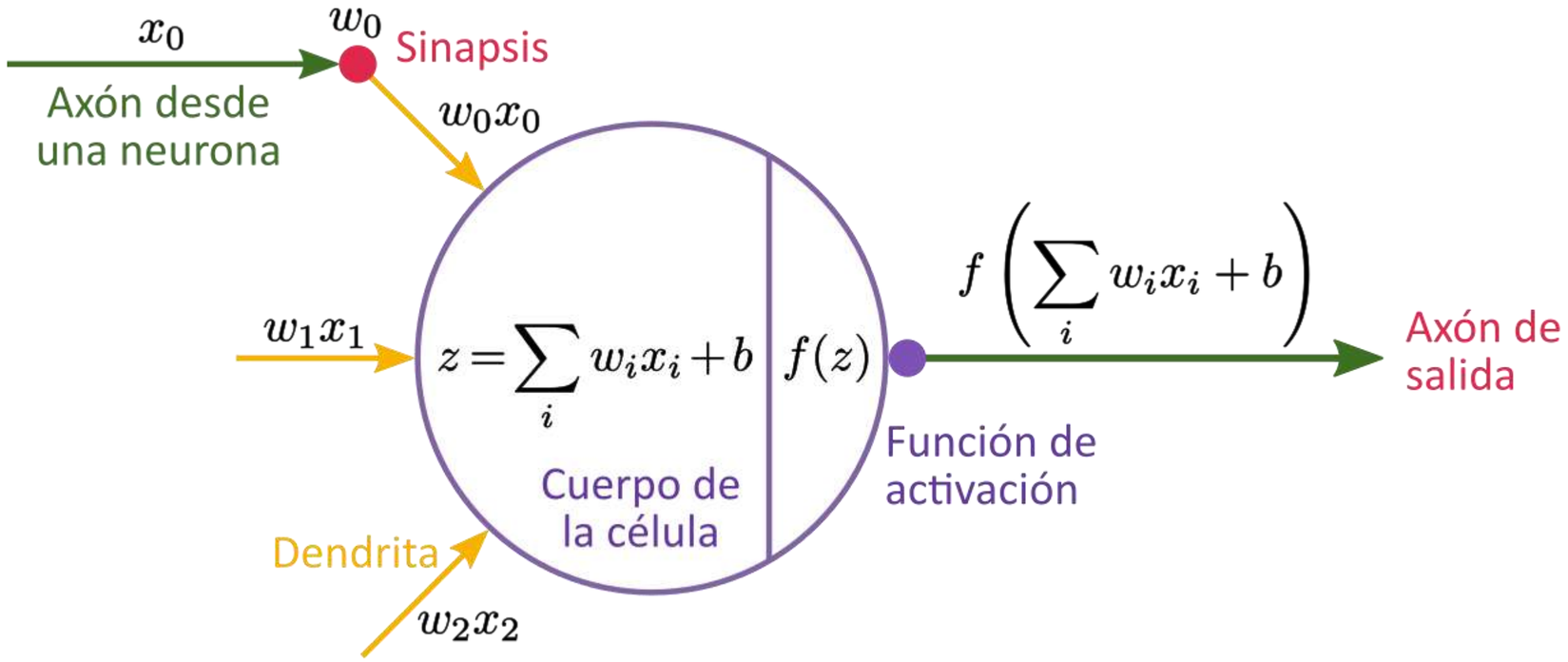
Dartmouth AI Conference 1956

Perceptron



F. Rosenblatt (1957)

Red neuronal *artificial*

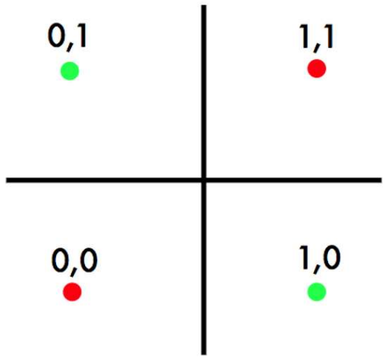


Frank Rosenblatt (1958) "The perceptron: a probabilistic model for information storage and organization in the brain". Psychological review 65.6 (1958): 386.

Perceptron

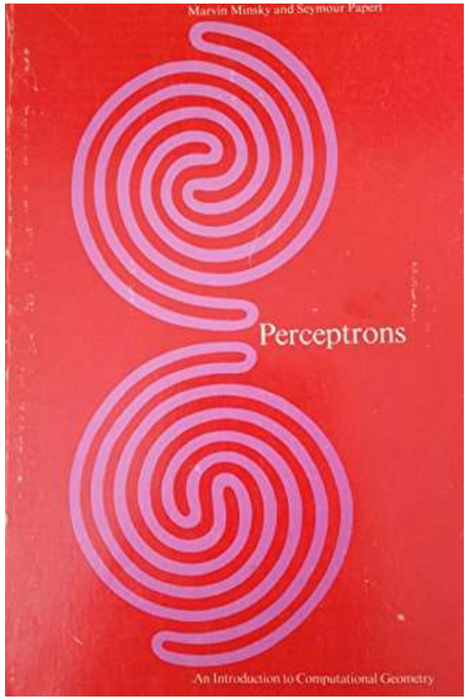
https://www.youtube.com/watch?v=cNxadbrN_aI

XOR *Affair*



XOR

El perceptrón no puede representar un XOR

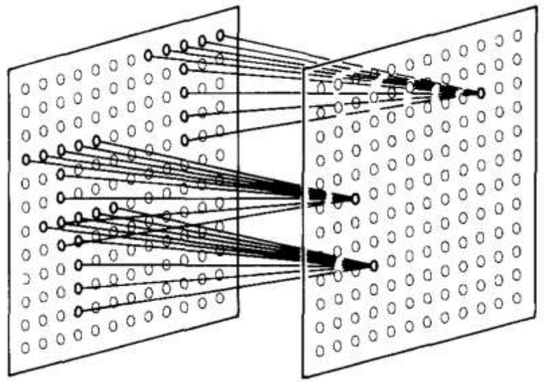


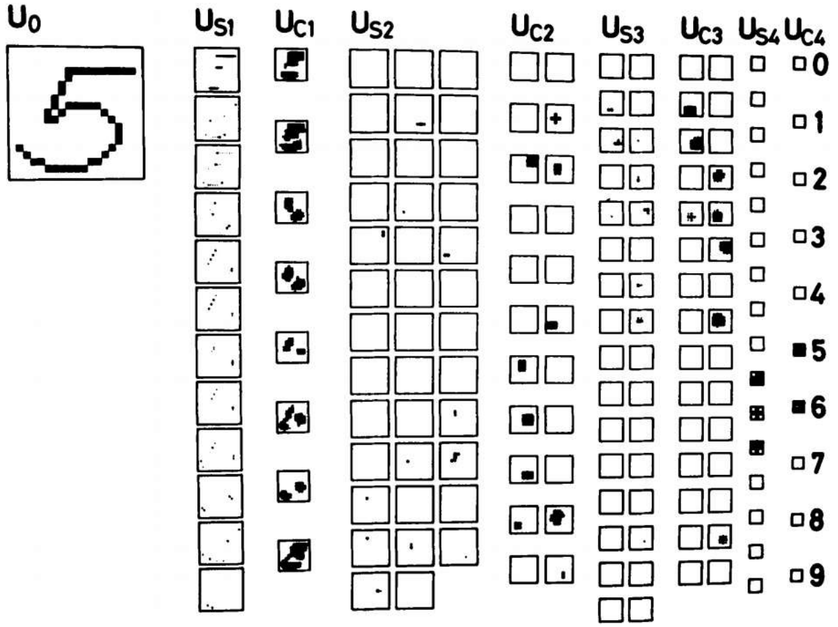
M. Minsky **S. Papert**

(1969)

AI winter

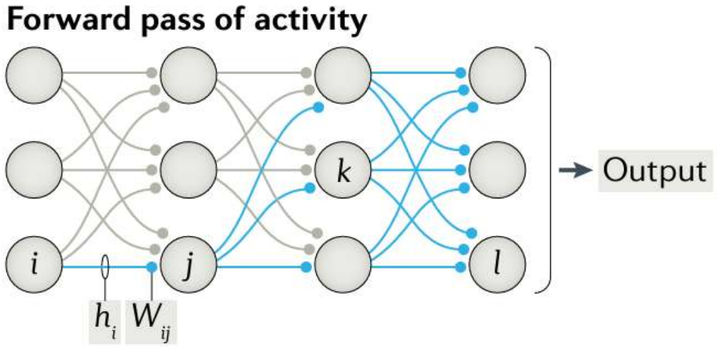
Neocognitron

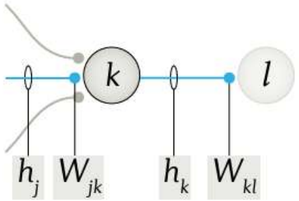




Kunihiko Fukushima (1980)

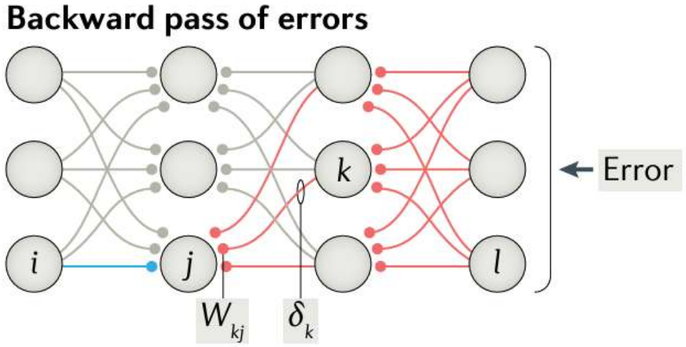
Back*propagation*

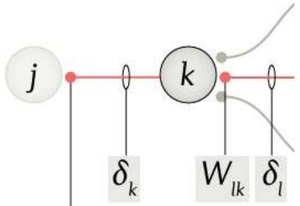




$$h_k = f(a_k)$$

$$a_k = \sum_j h_j W_{jk}$$





$$W_{kj} = W_{jk}$$

$$\delta_k = e_k f'(a_k)$$

$$e_k = \sum_l \delta_l W_{lk}$$

David Rumelhart

Geoffrey Hinton

Ronald Williams

David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams (1986) "Learning representations by back-propagating errors". nature, 323(6088), 533-536.

Universal *approximation*

Una red neuronal con al menos una hidden layer y activaciones no lineales continuas y acotadas puede aproximar cualquier función continua definida en un espacio compacto arbitrariamente bien, siempre que se le otorguen suficientes neuronas en la capa oculta.

Kurt Hornik (1989) "Multilayer feedforward networks are universal approximators". Neural networks, 2(5), 359-366.

Universal *approximation*

Una red neuronal con al menos una hidden layer y activaciones no lineales continuas y acotadas puede aproximar cualquier función continua definida en un espacio compacto arbitrariamente bien, siempre que se le otorguen suficientes neuronas en la capa oculta.

Formalmente:

Sea $K \subset R^n$ un conjunto compacto, y $f: K \to R$ una función continua. Entonces, para cualquier $\epsilon > 0$, existe una red neuronal $F(x; \theta)$, donde F es de la forma:

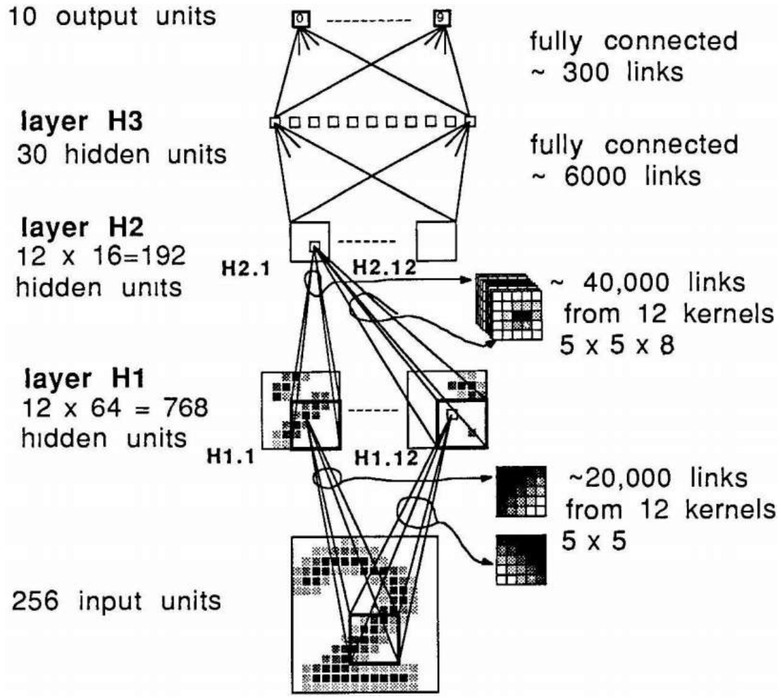
$$F(x;\theta) = \sum_{i=1}^{N} c_i \sigma\left(w_i^{\top} x + b_i\right) \qquad \text{tal que:} \qquad \sup_{x \in K} |f(x) - F(x;\theta)| < \epsilon$$

donde:

- $x \in R^n$: Entrada a la red.
- $w_i \in R^n$: Pesos de la capa de entrada.
- $b_i \in R$: Sesgos (biases).
- $c_i \in R$: Pesos de salida.
- $\sigma: R \to R$: Función de activación continua, acotada, no constante y no lineal.

Kurt Hornik (1989) "Multilayer feedforward networks are universal approximators". Neural networks, 2(5), 359-366.

Convolutional *neural network*



AT&T DSP-32C
capaz de 125m operaciones
de coma floating

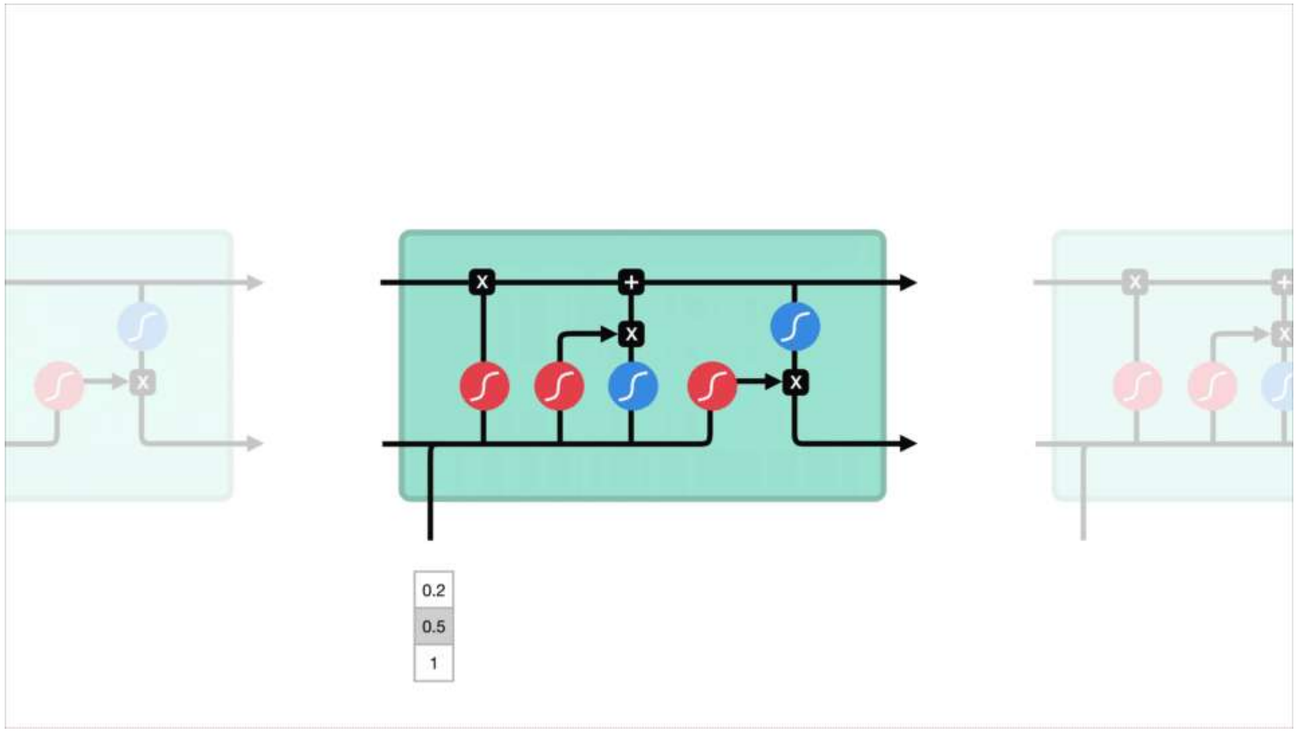
Yann LeCun (1989)

Yan LeCun (1989) "Generalization and network design strategies".
Connectionism in perspective.

Convolutional *neural network*

Yan LeCun (1989) "Handwritten digit recognition with a back-propagation network". Neural Information Process and System.

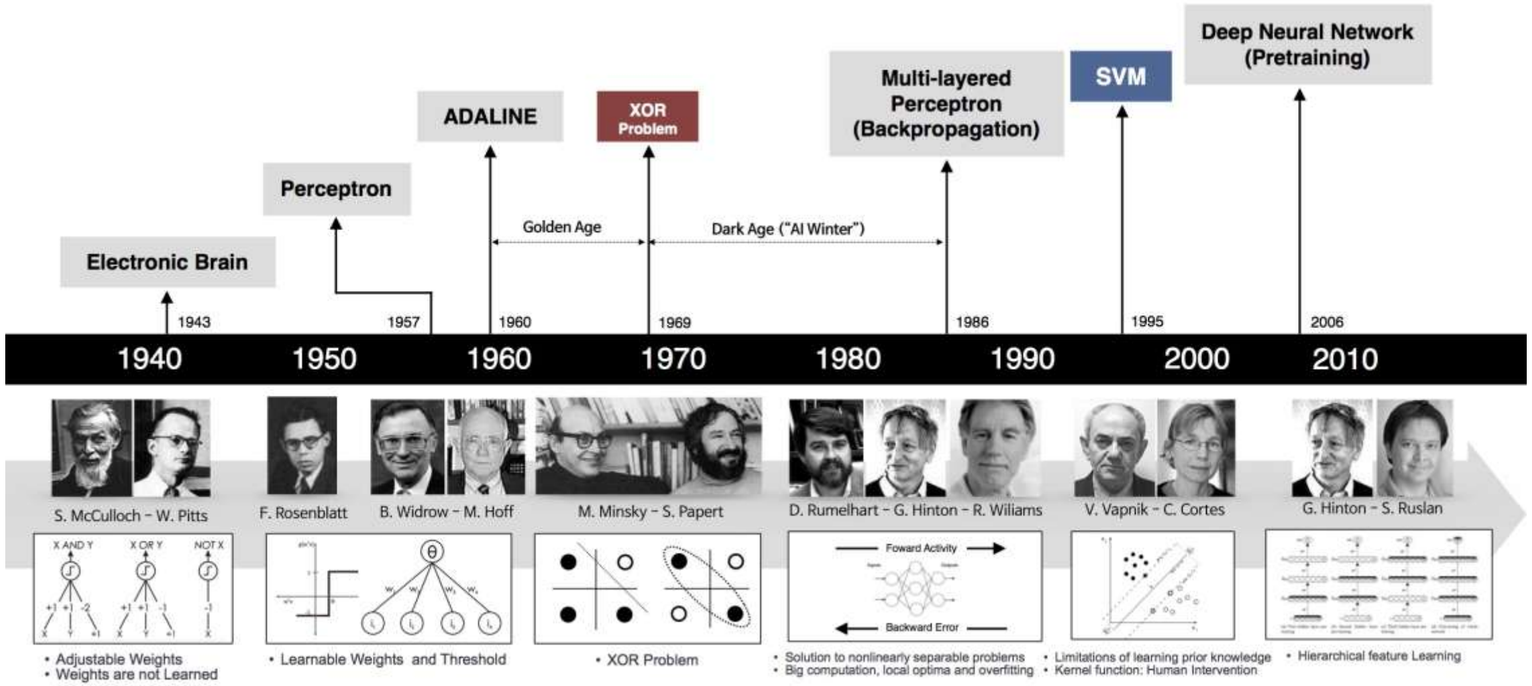
Long short-term *memory (LSTM)*



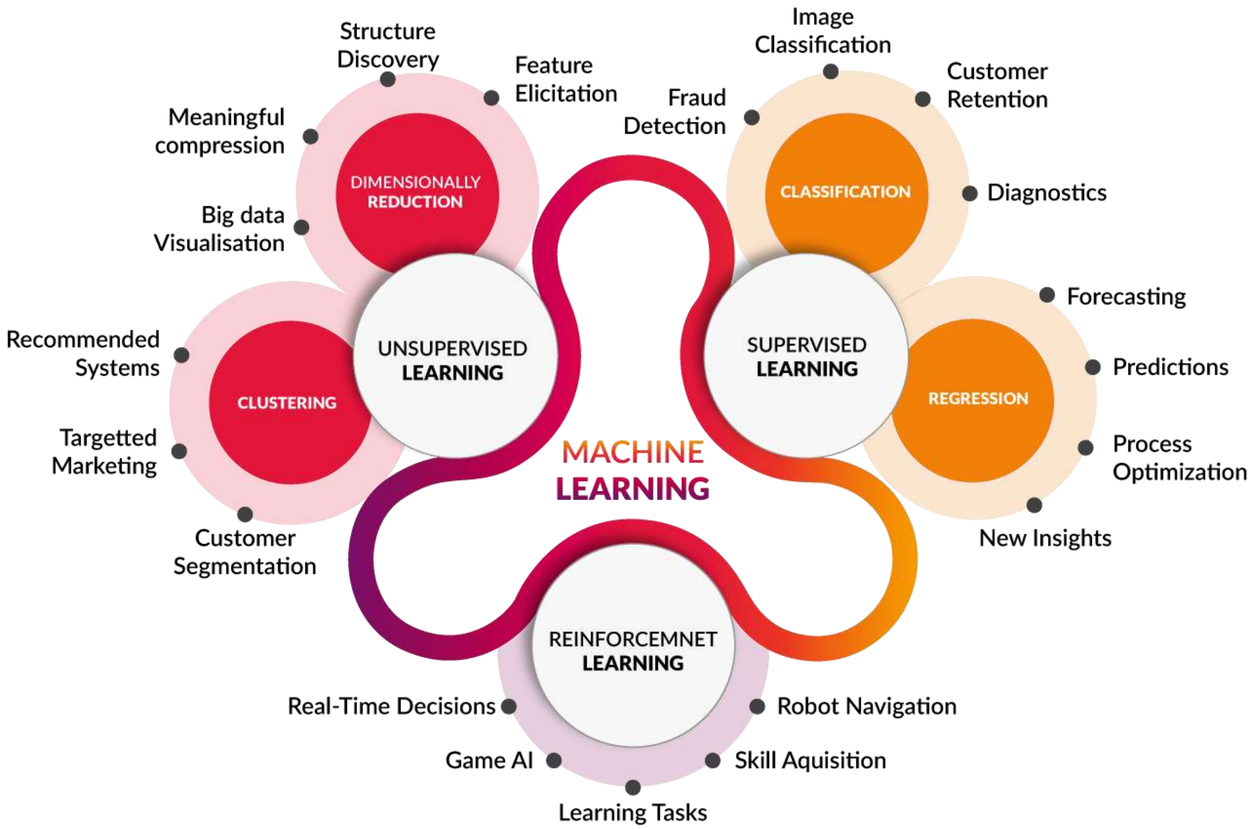
Sepp Hochreiter and Juergen Schmidhuber (1997)

Sepp Hochreiter and Juergen Schmidhuber (1997) "Long Short-term Memory". Neural Computation MIT-Press.

Inteligencia *artificial*

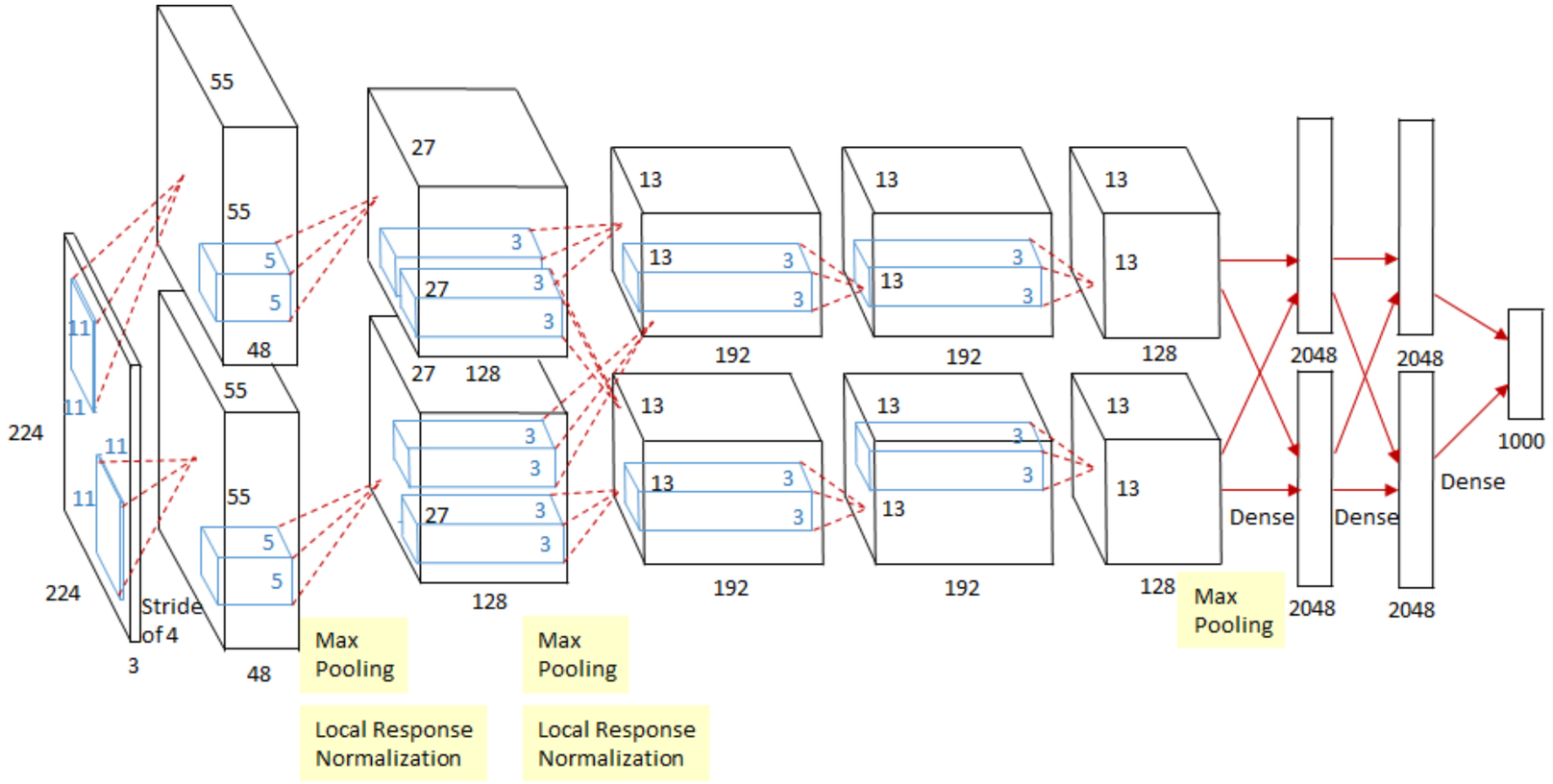


Machine *learning*



AlexNet

Alex Krizhevsky (2012)

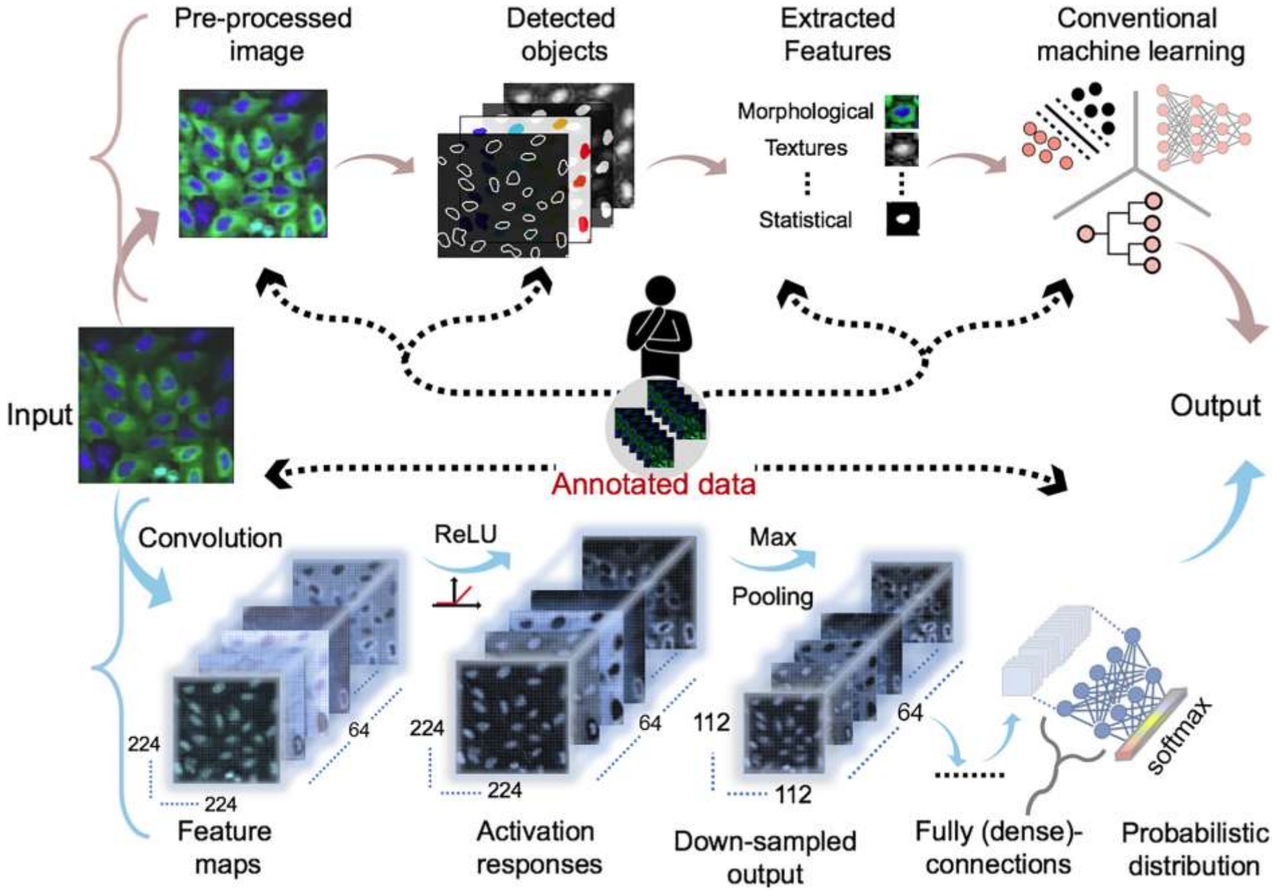


Alex Krizhevsky et al. (2012) "ImageNet Classification with Deep Convolutional Neural Networks". Advances in neural information processing systems.

Deep *learning*

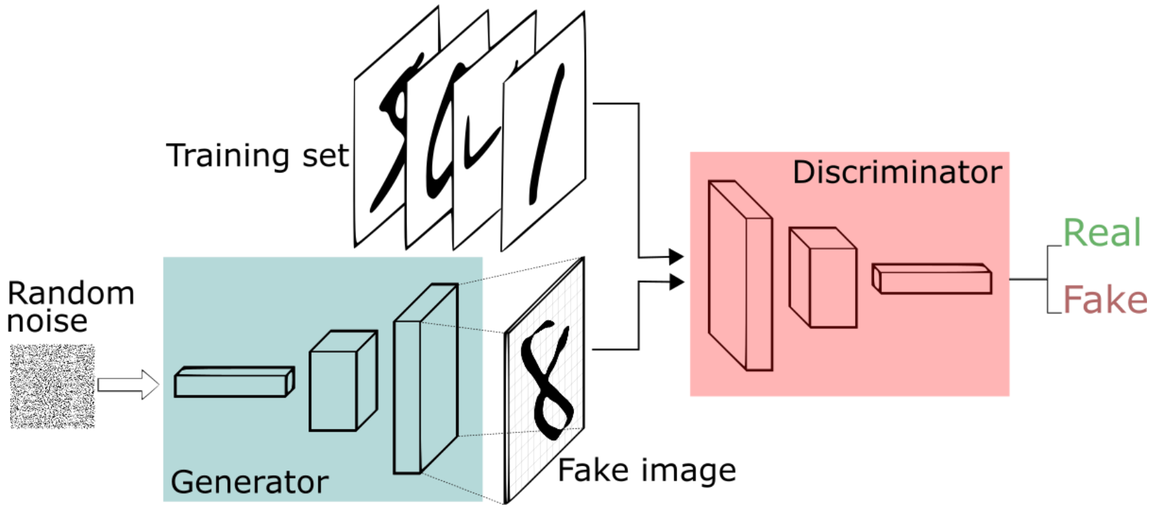
Handcraft method

Deep learning



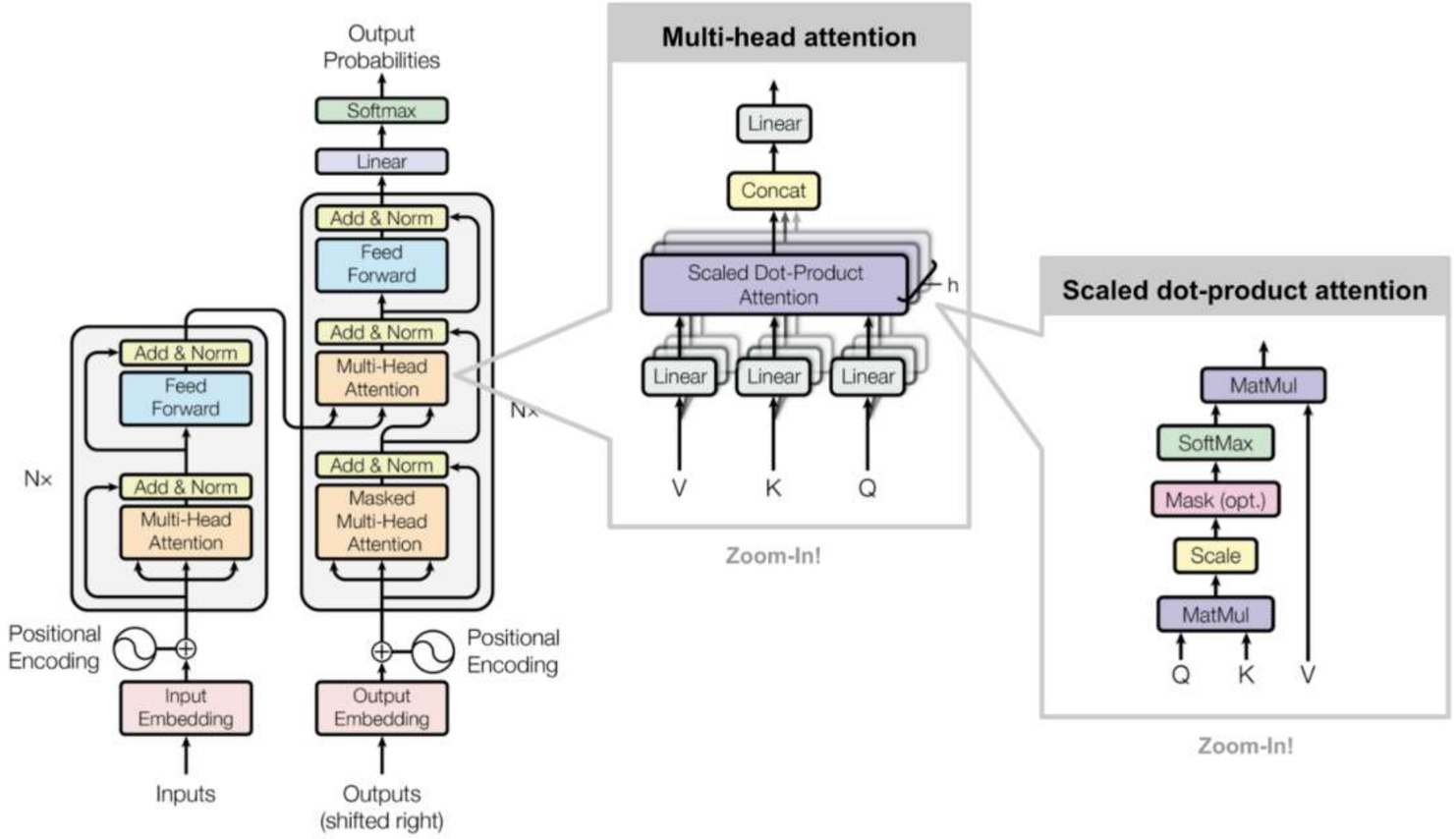
Generative Adversarial *Network (GAN)*

Ian J. Goodfellow (2014)



Ian J. Goodfellow (2014) "Generative Adversarial Networks". Advances in neural information processing systems, 27.

Attention is *all you need*



Ashish Vaswani (2017)

Ashish Vaswani et al. (2017) "Attention is all you need".
31st Conference on Neural Information Processing Systems (NIPS 2017)

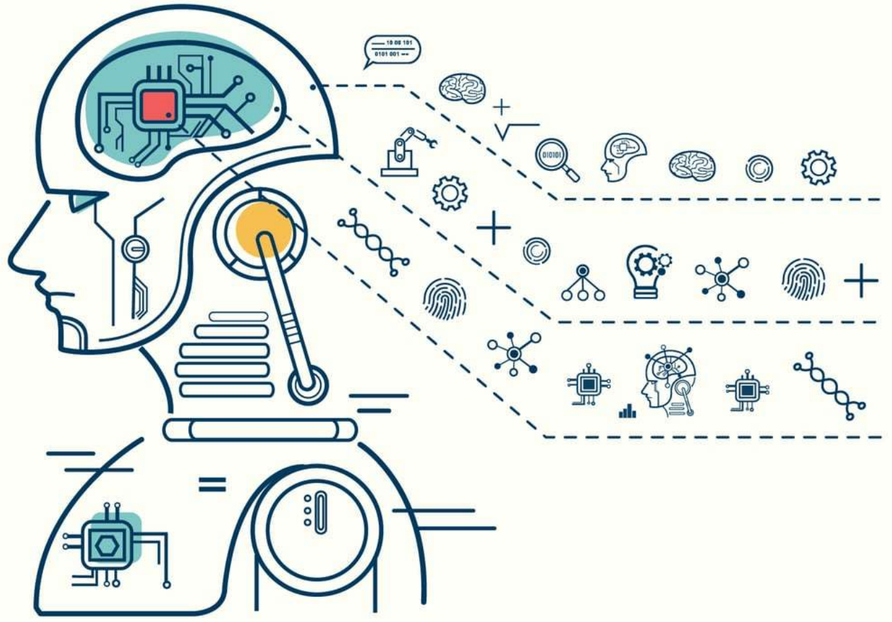
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

2.

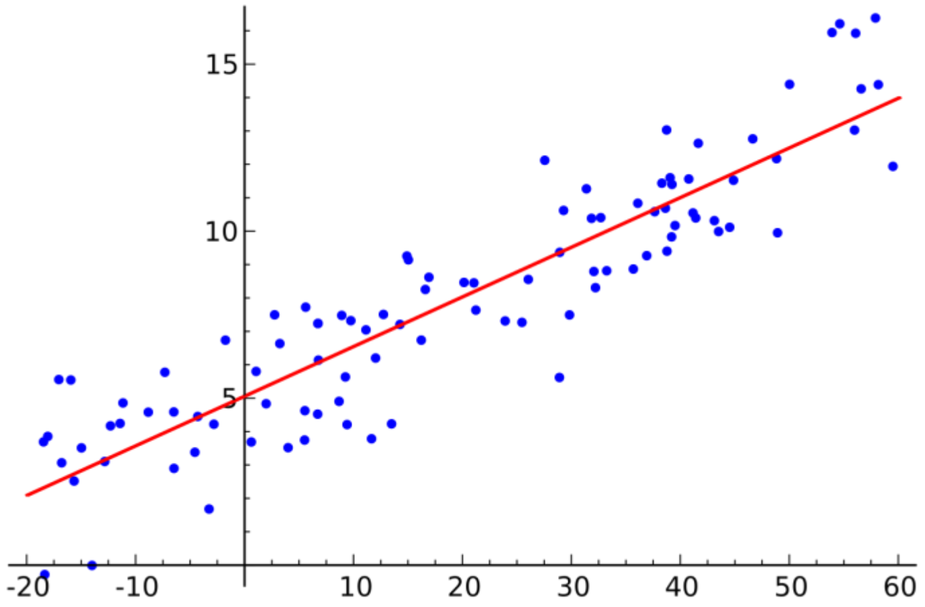
Mode*ling*

TRANSFORMATEC

Mod*eling*



Linear *model*



number of features

noise term

independence assumption

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i \qquad \varepsilon_i \overset{iid}{\sim} \mathbf{N}(0, \sigma^2)$$

response

global intercept

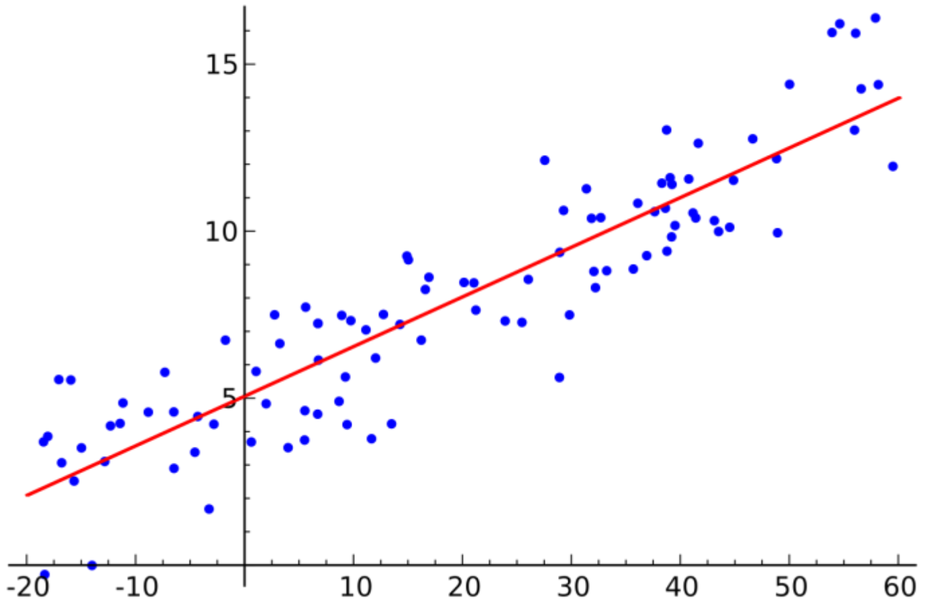
feature j of observation i

coefficient for feature j

noise level

Ordinary Least-*Squares Estimator*

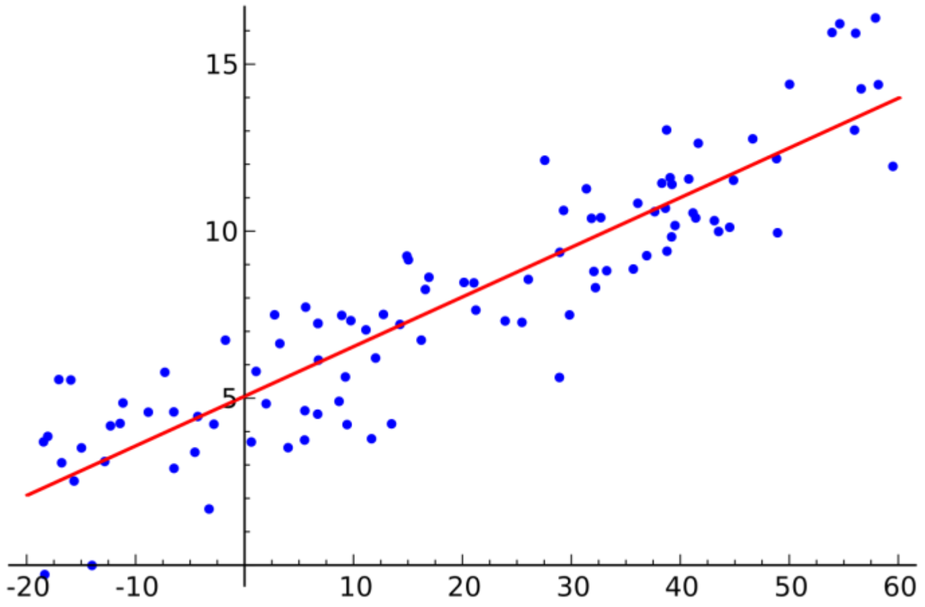
El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{\boldsymbol{y}} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.



$$J(w) = \frac{1}{2}\|\boldsymbol{Xw} - \boldsymbol{y}\|^2$$

Ordinary Least-*Squares Estimator*

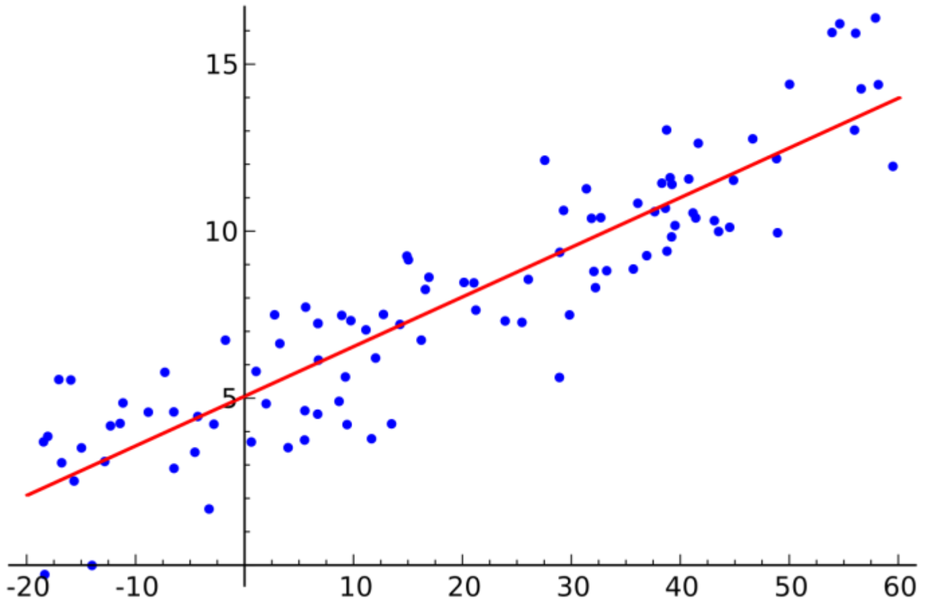
El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{\boldsymbol{y}} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.



$$J(w) = \frac{1}{2}\|\boldsymbol{Xw} - \boldsymbol{y}\|^2 = \frac{1}{2}(\mathbf{Xw} - \mathbf{y})^{\mathrm{T}}(\mathbf{Xw} - \mathbf{y})$$

Ordinary Least-*Squares Estimator*

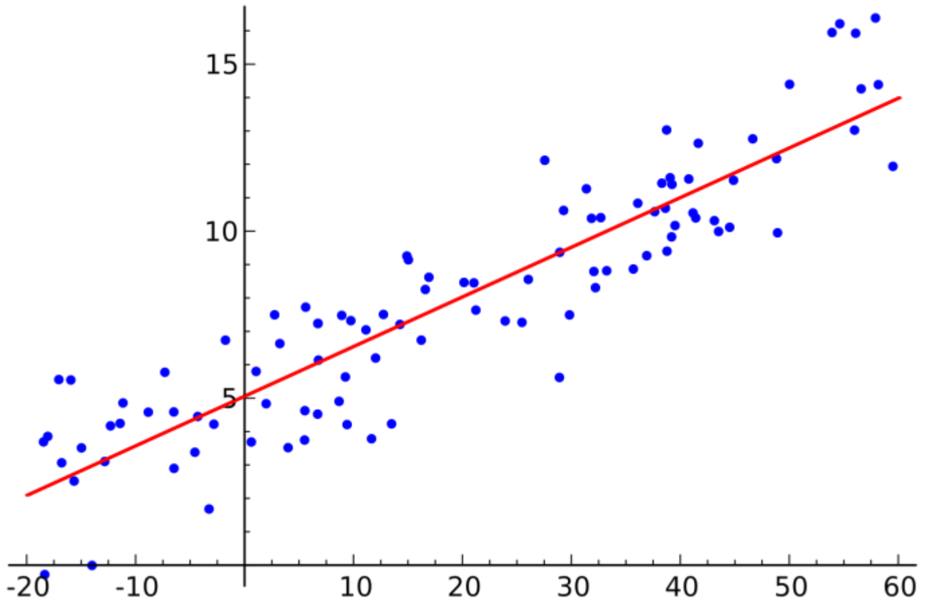
El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{\boldsymbol{y}} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.



$$J(w) = \frac{1}{2}\|\boldsymbol{Xw} - \boldsymbol{y}\|^2 = \frac{1}{2}(\mathbf{Xw} - \mathbf{y})^{\mathrm{T}}(\mathbf{Xw} - \mathbf{y}) = \frac{1}{2}[\boldsymbol{w^T X^T Xw} - 2\boldsymbol{w^T X^T y} + \boldsymbol{y^T y}]$$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{y} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.



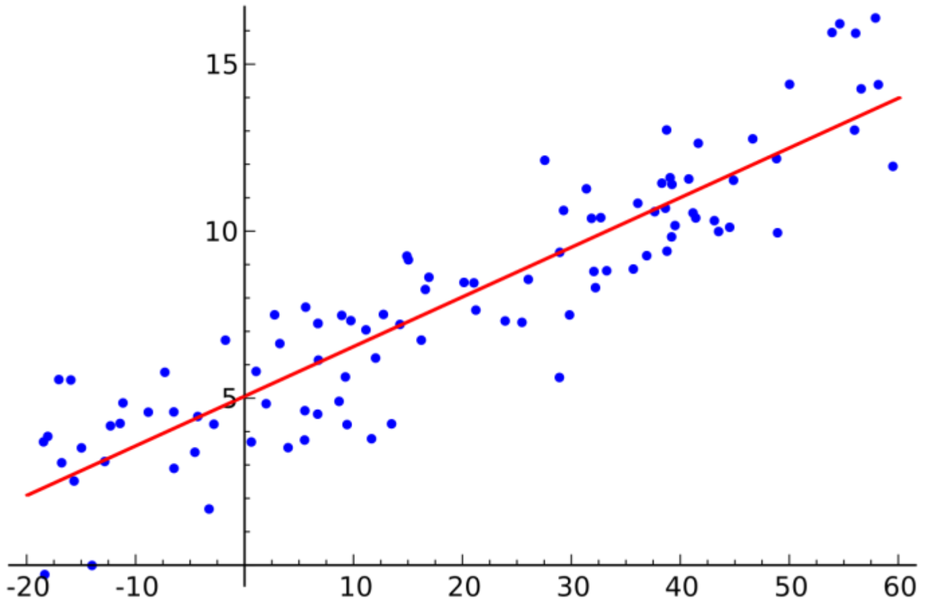
$$J(w) = \frac{1}{2}\|\boldsymbol{Xw} - \boldsymbol{y}\|^2 = \frac{1}{2}(\mathbf{Xw} - \mathbf{y})^{\mathrm{T}}(\mathbf{Xw} - \mathbf{y}) = \frac{1}{2}[\boldsymbol{w^T X^T Xw} - 2\boldsymbol{w^T X^T y} + \boldsymbol{y^T y}]$$

Derivando respecto a **w**:

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial \boldsymbol{w}}\left(\frac{1}{2}\boldsymbol{w^T X^T Xw}\right) + \frac{\partial}{\partial \boldsymbol{w}}(-\boldsymbol{w^T X^T y}) + \frac{\partial}{\partial \boldsymbol{w}}\left(\frac{1}{2}\boldsymbol{y^T y}\right)$$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{y} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.



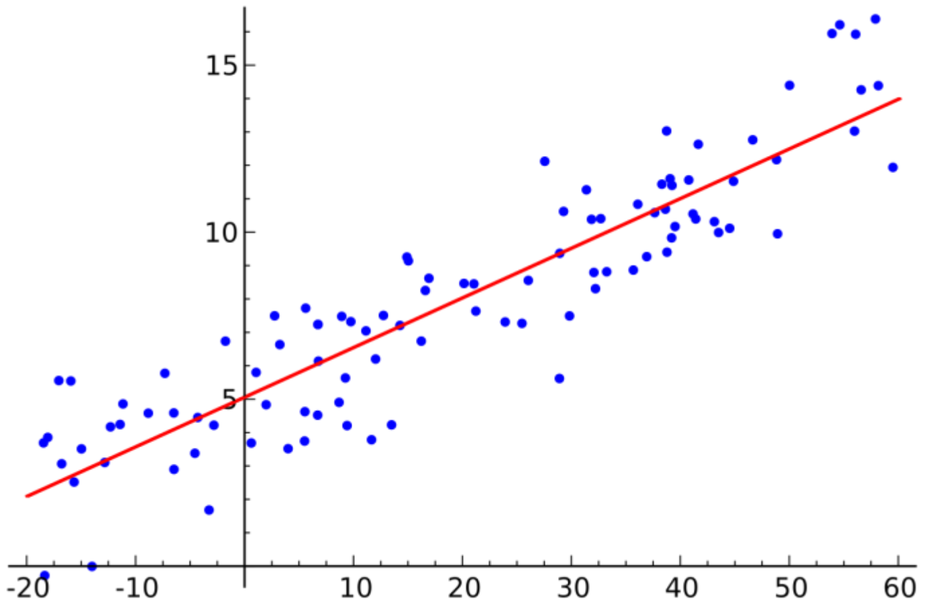
$$J(w) = \frac{1}{2}\|\boldsymbol{Xw} - \boldsymbol{y}\|^2 = \frac{1}{2}(\mathbf{Xw} - \mathbf{y})^{\mathrm{T}}(\mathbf{Xw} - \mathbf{y}) = \frac{1}{2}[\boldsymbol{w^T X^T Xw} - 2\boldsymbol{w^T X^T y} + \boldsymbol{y^T y}]$$

Derivando respecto a **w**:

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial \boldsymbol{w}}\left(\frac{1}{2}\boldsymbol{w^T X^T Xw}\right) + \frac{\partial}{\partial \boldsymbol{w}}(-\boldsymbol{w^T X^T y}) + \frac{\partial}{\partial \boldsymbol{w}}\left(\frac{1}{2}\boldsymbol{y^T y}\right) = \mathbf{X}^{\mathrm{T}}\mathbf{Xw} - \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{y} = Xw$ y los valores reales y.



$$J(w) = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) = \frac{1}{2}[w^T X^T X w - 2w^T X^T y + y^T y]$$

Derivando respecto a $\mathbf{w}$:

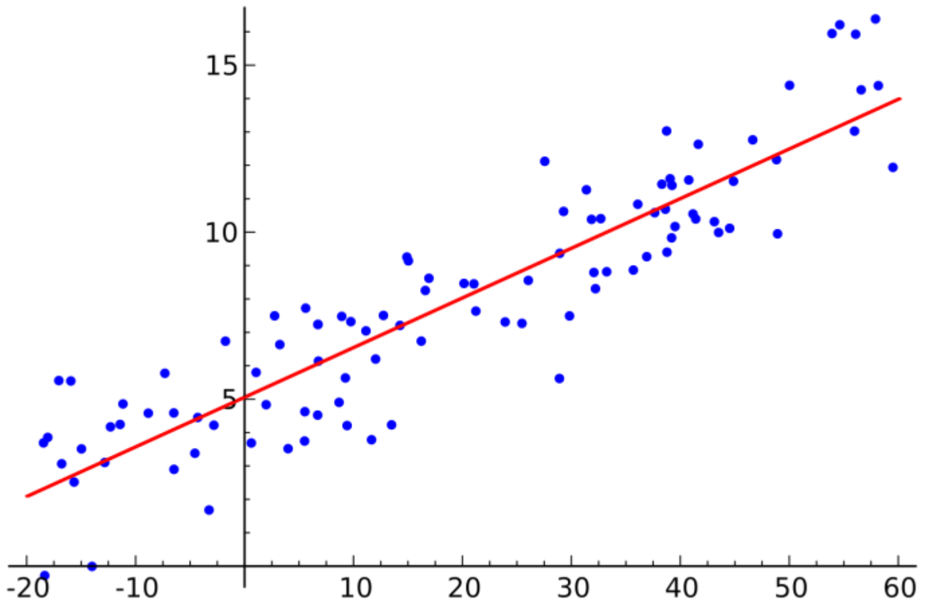
$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}w^T X^T X w\right) + \frac{\partial}{\partial w}(-w^T X^T y) + \frac{\partial}{\partial w}\left(\frac{1}{2}y^T y\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Buscamos minimizar $J(\boldsymbol{w})$, entonces igualamos la derivada a cero:

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y} = 0$$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{y} = Xw$ y los valores reales y.



$$J(w) = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) = \frac{1}{2}[w^T X^T X w - 2w^T X^T y + y^T y]$$

Derivando respecto a $\mathbf{w}$:

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}w^T X^T X w\right) + \frac{\partial}{\partial w}(-w^T X^T y) + \frac{\partial}{\partial w}\left(\frac{1}{2}y^T y\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

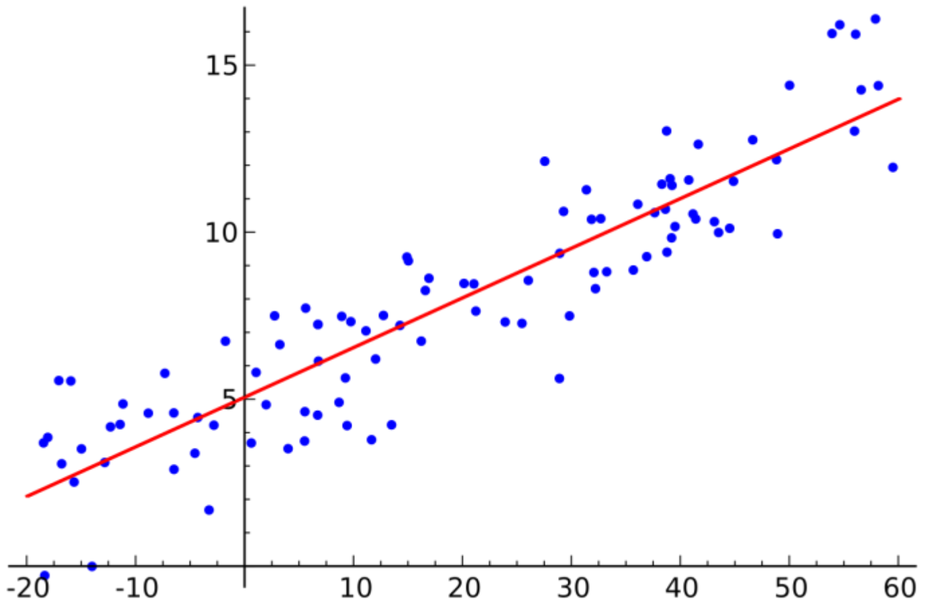
Buscamos minimizar $J(w)$, entonces igualamos la derivada a cero:

$$\frac{\partial J(w)}{\partial w} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y} = 0$$

Reorganizando: $X^T X w = X^T y$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{y} = Xw$ y los valores reales y.



$$J(w) = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathrm{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) = \frac{1}{2}[w^T X^T X w - 2w^T X^T y + y^T y]$$

Derivando respecto a $\mathbf{w}$:

$$\frac{\partial J(w)}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{2}w^T X^T X w\right) + \frac{\partial}{\partial w}\left(-w^T X^T y\right) + \frac{\partial}{\partial w}\left(\frac{1}{2}y^T y\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Buscamos minimizar $J(w)$, entonces igualamos la derivada a cero:

$$\frac{\partial J(w)}{\partial w} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y} = 0$$

Reorganizando: $X^T X w = X^T y$

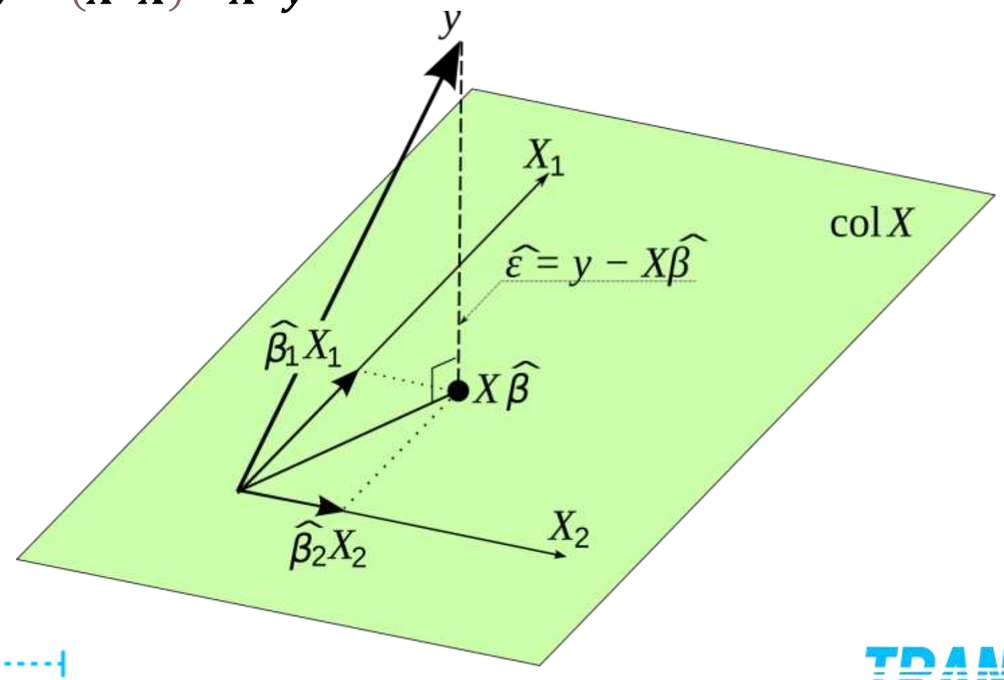
Despejamos w: $w = (X^T X)^{-1} X^T y$

Ordinary Least-*Squares Estimator*

El objetivo de OLS es minimizar la suma de los errores al cuadrado entre las predicciones $\hat{\boldsymbol{y}} = \boldsymbol{Xw}$ y los valores reales $\boldsymbol{y}$.

Solución analítica:

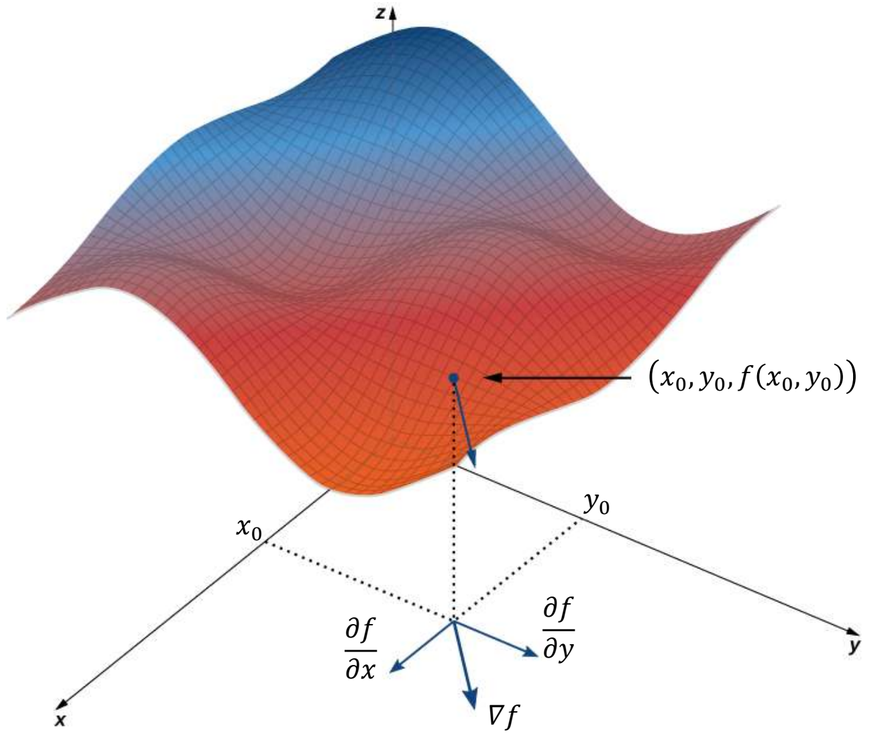
$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$



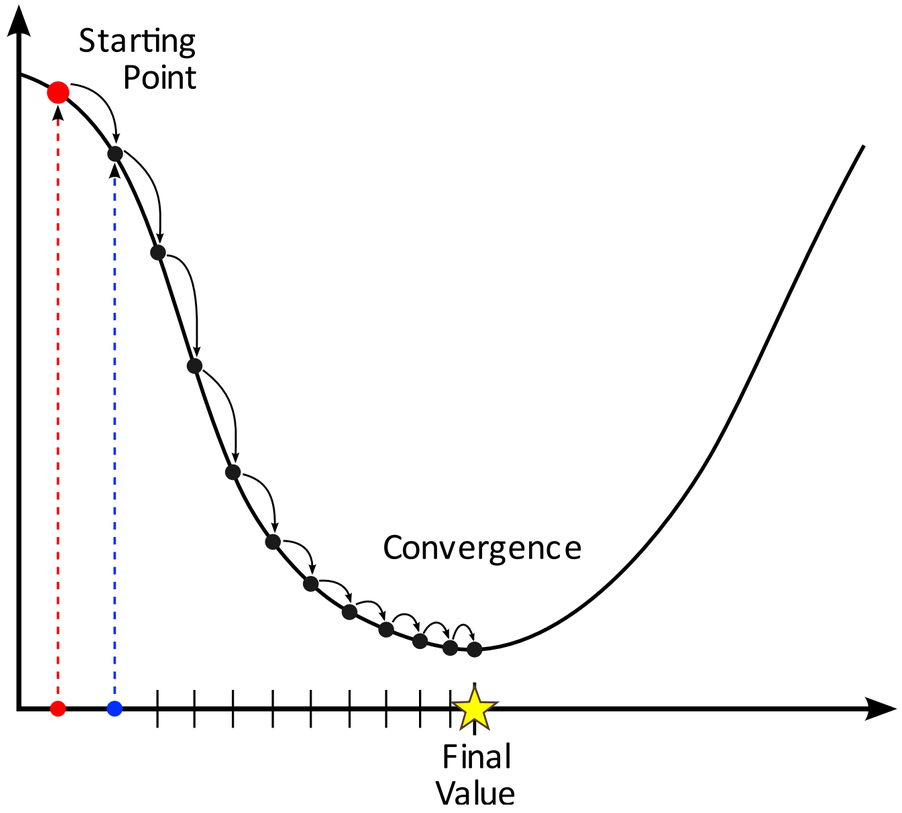
Gradient *descent*

Gradiente:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$



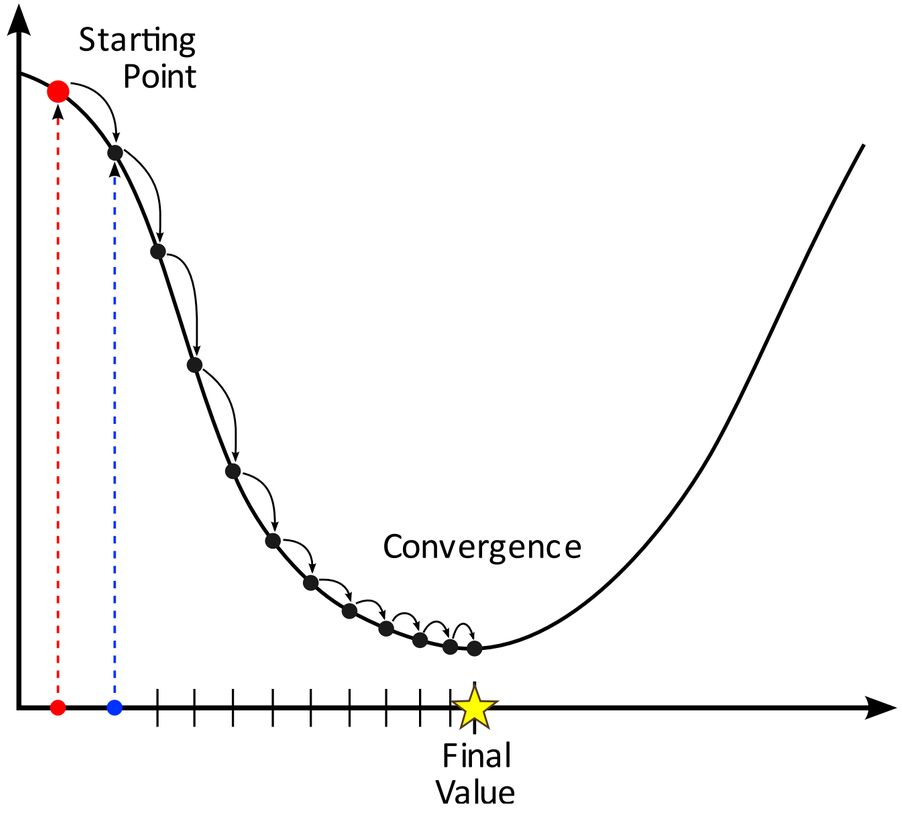
Gradient *descent*



$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

Gradient *descent*

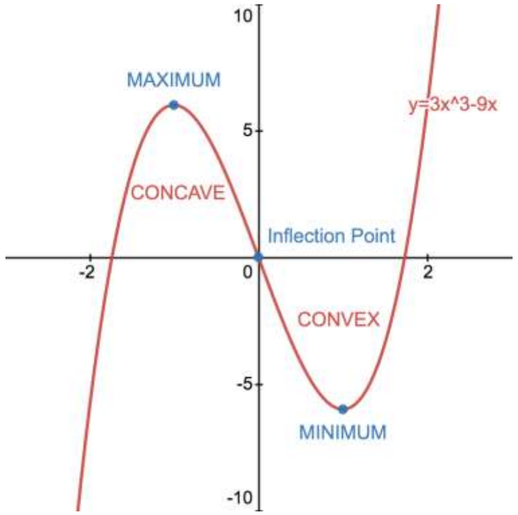


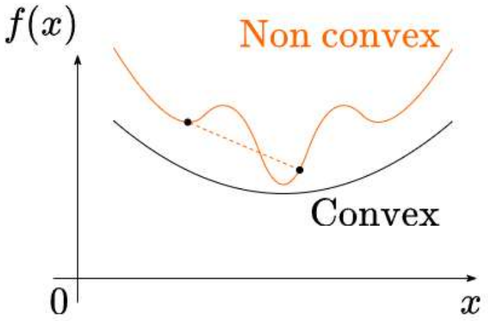
$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

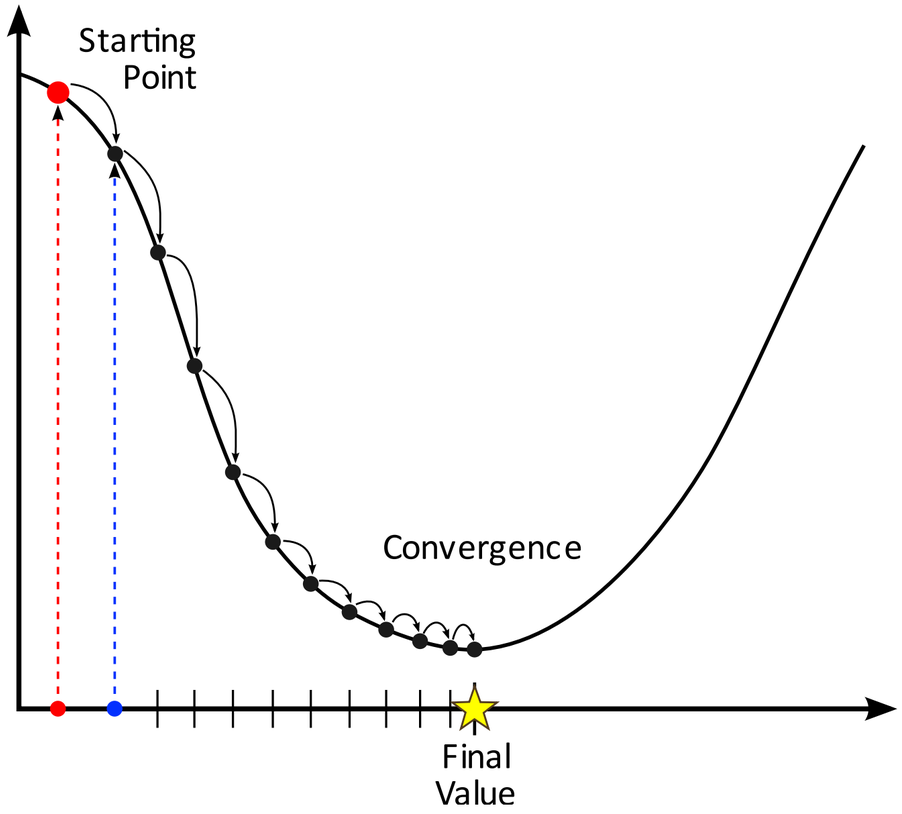
Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa:





Gradient *descent*

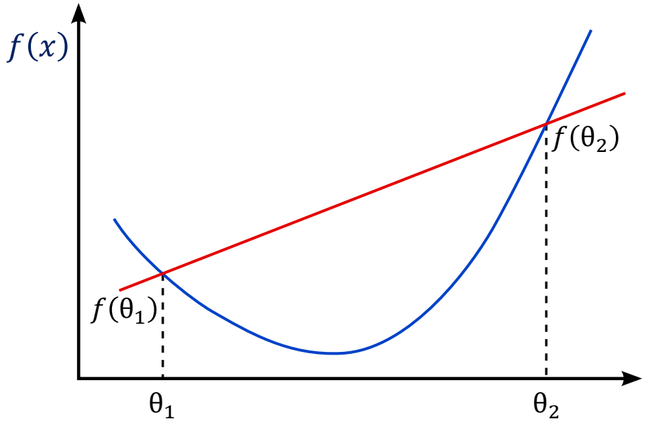


$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

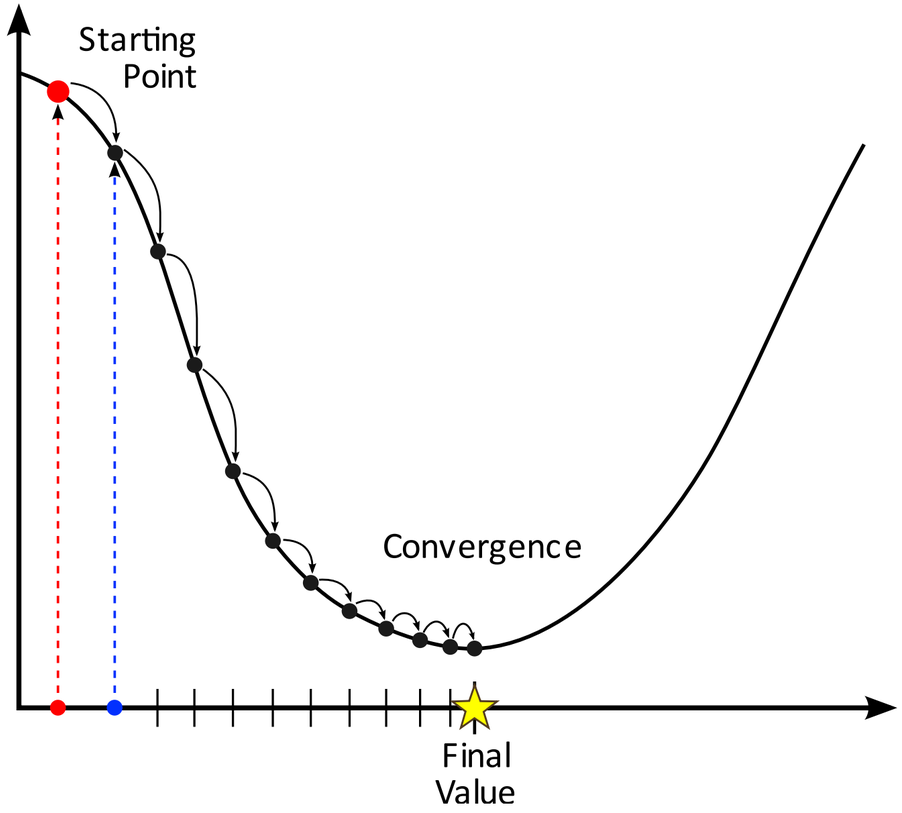
donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa:



Gradient *descent*

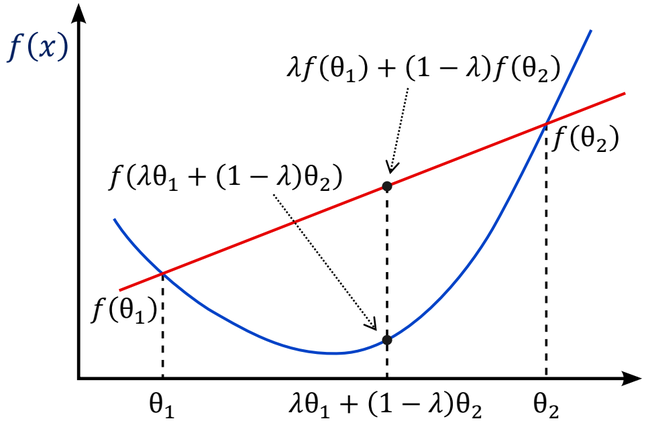


$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

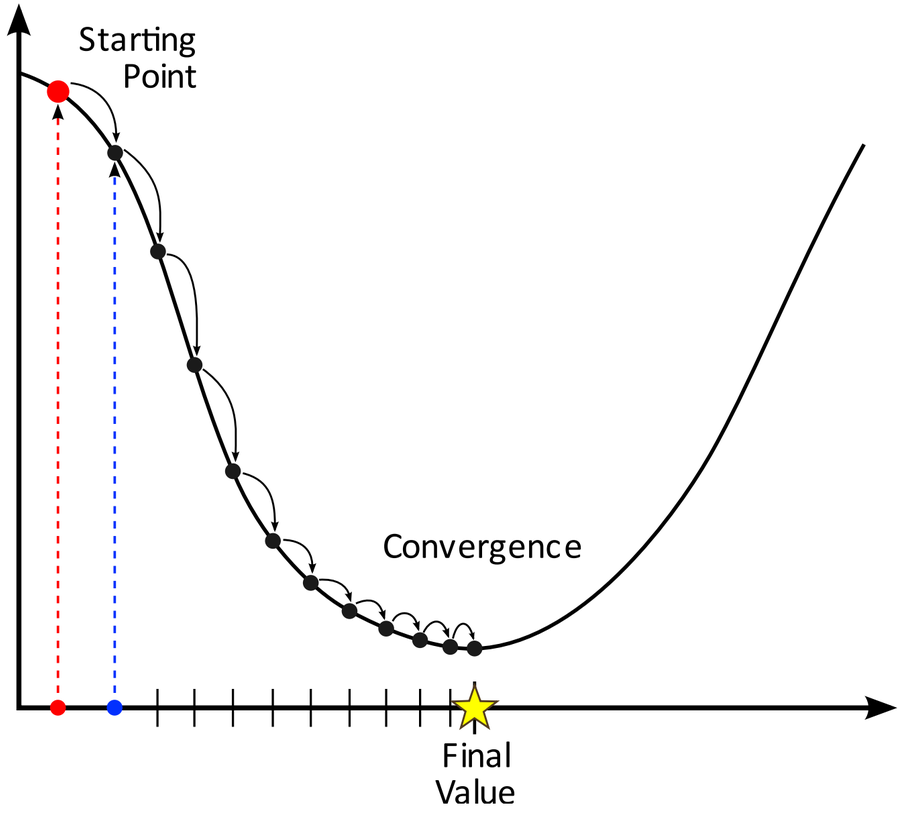
donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa:



Gradient *descent*

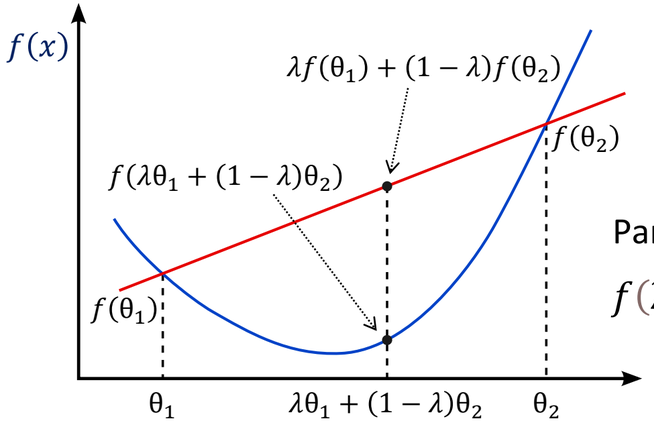


$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

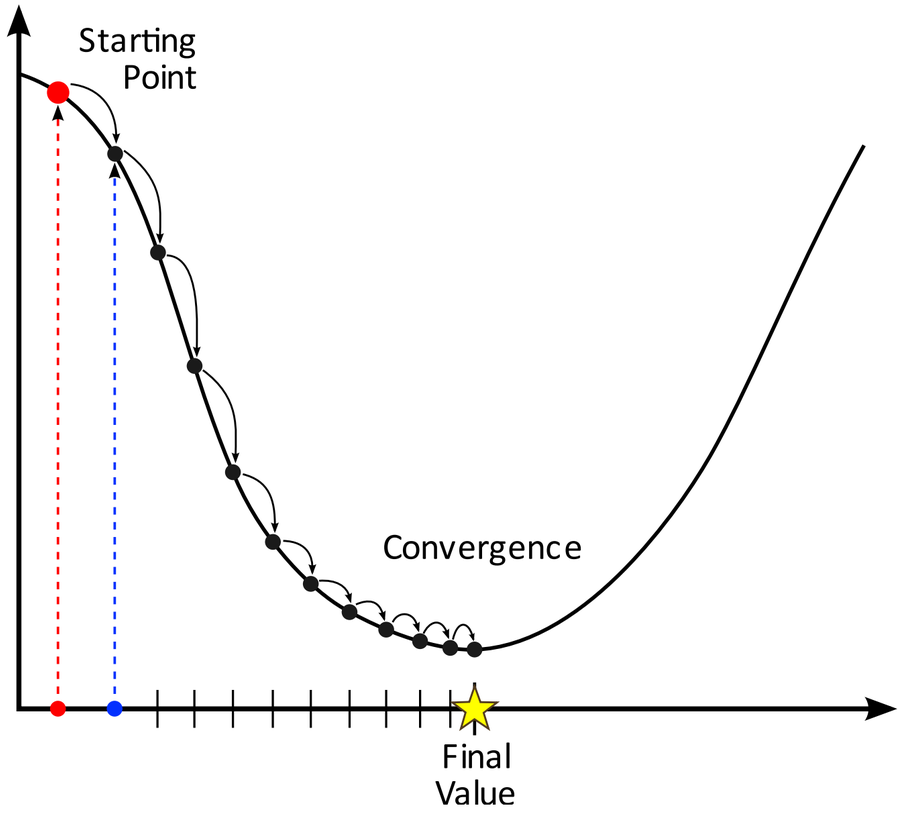
Función convexa:



Para cualquier θ_1, θ_2 y $\lambda \in [0,1]$, se cumple:

$$f(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda f(\theta_1) + (1-\lambda) f(\theta_2)$$

Gradient *descent*



$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

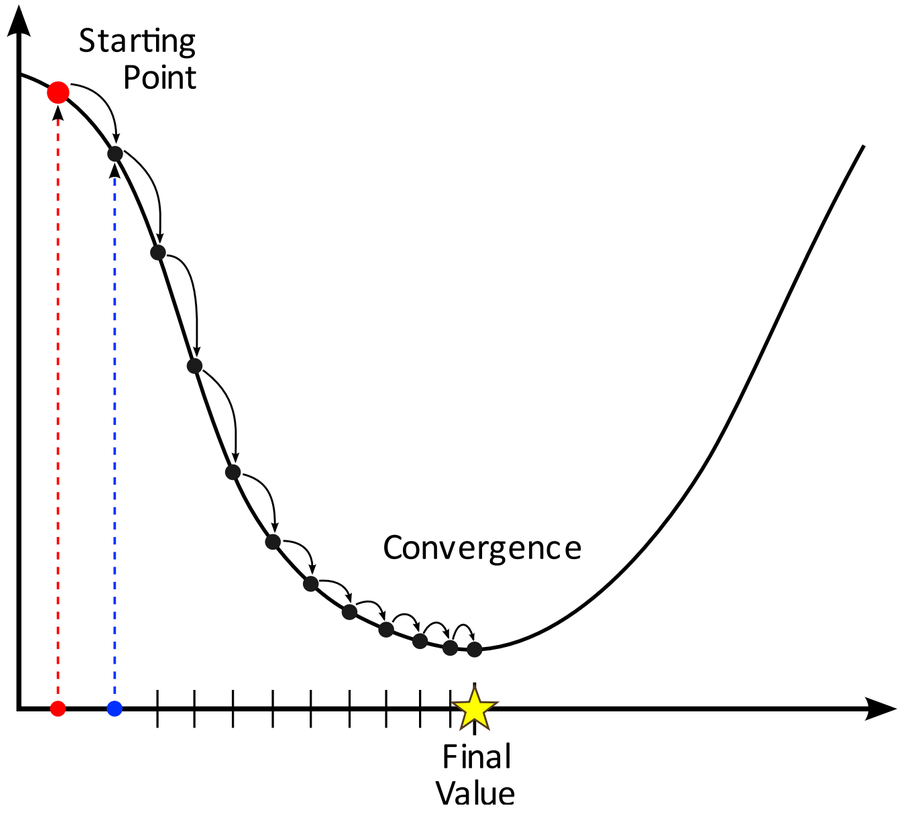
Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa: Para cualquier θ_1, θ_2 y $\lambda \in [0,1]$, se cumple:

$$J(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda J(\theta_1) + (1-\lambda) J(\theta_2)$$

Si $J(\theta)$ es estrictamente convexa, el mínimo será único.

Gradient *descent*



$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa: Para cualquier θ_1, θ_2 y $\lambda \in [0,1]$, se cumple:

$$J(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda J(\theta_1) + (1-\lambda) J(\theta_2)$$

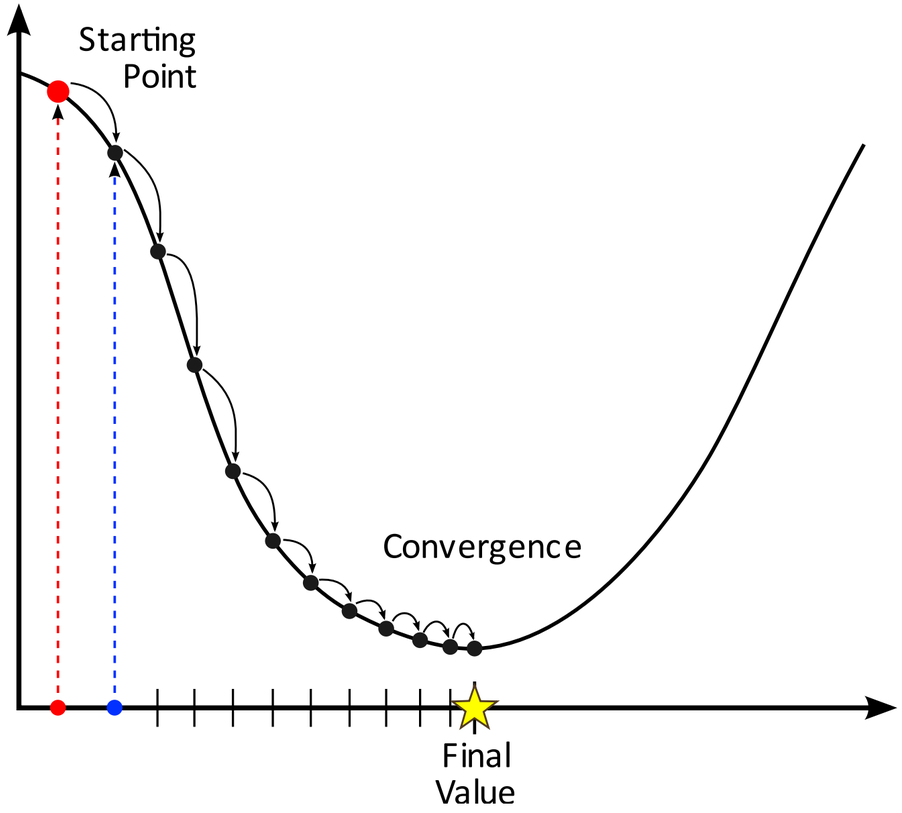
Si $J(\theta)$ es estrictamente convexa, el mínimo será único.

Lipschitz Continuity del Gradiente: Existe una constante $L > 0$ tal que:

$$|\nabla J(\theta_1) - \nabla J(\theta_2)| \leq L|\theta_1 - \theta_2|, \quad \forall \theta_1, \theta_2$$

Esto implica que $J(\theta)$ es una función suavemente diferenciable con una curvatura controlada.

Gradient *descent*



$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k, y el learning rate $\alpha > 0$.

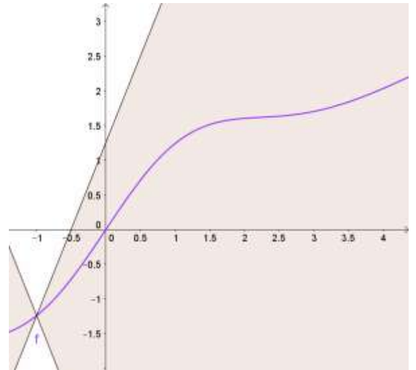
Se garantiza convergencia en el optimo global si $J(\theta_k)$ tiene las siguientes propiedades:

Función convexa: Para cualquier θ_1, θ_2 y $\lambda \in [0,1]$, se cumple:

$$J(\lambda\theta_1 + (1-\lambda)\theta_2) \leq \lambda J(\theta_1) + (1-\lambda) J(\theta_2)$$

Si $J(\theta)$ es estrictamente convexa, el mínimo será único.

Lipschitz Continuity del Gradiente:



Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Usamos el **Lema de Descenso**: $J(\theta_{k+1}) \leq J(\theta_k) + \nabla J(\theta_k)^T \|\theta_{k+1} - \theta_k\| + \frac{L}{2}\|\theta_{k+1} - \theta_k\|^2$

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Usamos el **Lema de Descenso**:

$$J(\theta_{k+1}) \leq J(\theta_k) + \nabla J(\theta_k)^T \|\theta_{k+1} - \theta_k\| + \frac{L}{2}\|\theta_{k+1} - \theta_k\|^2$$

$$= J(\theta_k) + \nabla J(\theta_k)^T \left(-\alpha \nabla J(\theta_k)\right) + \frac{L}{2}\left(-\alpha \nabla J(\theta_k)\right)^2$$

$$= J(\theta_k) - \alpha \|\nabla J(\theta_k)\|^2 + \frac{L\alpha^2}{2}\|\nabla J(\theta_k)\|^2$$

$$= J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2$$

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Usamos el **Lema de Descenso**:

$$\begin{aligned} J(\theta_{k+1}) &\leq J(\theta_k) + \nabla J(\theta_k)^T \|\theta_{k+1} - \theta_k\| + \frac{L}{2}\|\theta_{k+1} - \theta_k\|^2 \\ &= J(\theta_k) + \nabla J(\theta_k)^T \left(-\alpha \nabla J(\theta_k)\right) + \frac{L}{2}\left(-\alpha \nabla J(\theta_k)\right)^2 \\ &= J(\theta_k) - \alpha \|\nabla J(\theta_k)\|^2 + \frac{L\alpha^2}{2}\|\nabla J(\theta_k)\|^2 \\ &= J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \end{aligned}$$

Para garantizar que la sucesión $\{J(\theta_k)\}_{k=0}^{\infty}$ sea **monótonamente decreciente**, se requiere que el coeficiente delante de $\|\nabla J(\theta_k)\|^2$ sea positivo, es decir:

$$1 - \frac{L\,\alpha}{2} > 0 \quad \Leftrightarrow \quad 0 < \alpha < \frac{2}{L}$$

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$:

$$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$: $$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

De la desigualdad, ordenamos: $$\alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \leq J(\theta_k) - J(\theta_{k+1})$$

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$:

$$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

De la desigualdad, ordenamos: $\alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \leq J(\theta_k) - J(\theta_{k+1})$

$$\alpha\left(1 - \frac{L\alpha}{2}\right)\sum_{k=0}^{K-1}\|\nabla J(\theta_k)\|^2 \leq \sum_{k=0}^{K-1}[J(\theta_k) - J(\theta_{k+1})]$$

(suma telescópica)

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$:

$$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

De la desigualdad, ordenamos: $\alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \leq J(\theta_k) - J(\theta_{k+1})$

$$\alpha\left(1 - \frac{L\alpha}{2}\right)\sum_{k=0}^{K-1}\|\nabla J(\theta_k)\|^2 \leq J(\theta_0) - J(\theta_K)$$

Acotada porque $J(\theta_k)$ es monótona decreciente y tiene un límite inferior (función convexa)

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$: $$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

De la desigualdad, ordenamos: $$\alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \leq J(\theta_k) - J(\theta_{k+1})$$

$$\alpha\left(1 - \frac{L\alpha}{2}\right)\sum_{k=0}^{K-1}\|\nabla J(\theta_k)\|^2 \leq J(\theta_0) - J(\theta_K)$$

Acotada porque $J(\theta_k)$ es monótona decreciente y tiene un límite inferior (función convexa)

Entonces: $$\sum_{k=0}^{\infty}\|\nabla J(\theta_k)\|^2 \leq \infty$$

Convergencia de una serie:

Si $\sum_{k=0}^{\infty} a_k \leq \infty$, entonces $a_k \to 0$ cuando $k \to \infty$

Gradient *descent*

El objetivo es demostrar que el descenso de gradiente converge al óptimo global θ^*, donde: $\theta^* = \arg\min_{\theta} J(\theta)$

Para $0 < \alpha < \frac{2}{L}$: $$J(\theta_{k+1}) \leq J(\theta_k) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 < J(\theta_k)$$

Por lo tanto, $\{J(\theta_k)\}$ es una sucesión decreciente.

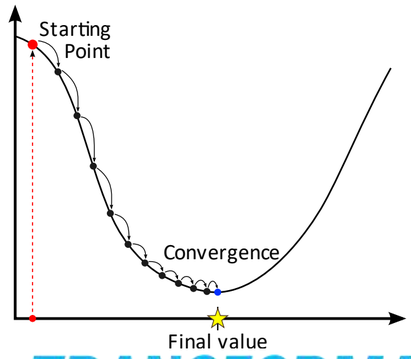
De la desigualdad, ordenamos: $$\alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla J(\theta_k)\|^2 \leq J(\theta_k) - J(\theta_{k+1})$$

$$\alpha\left(1 - \frac{L\alpha}{2}\right)\sum_{k=0}^{K-1}\|\nabla J(\theta_k)\|^2 \leq J(\theta_0) - J(\theta_K)$$

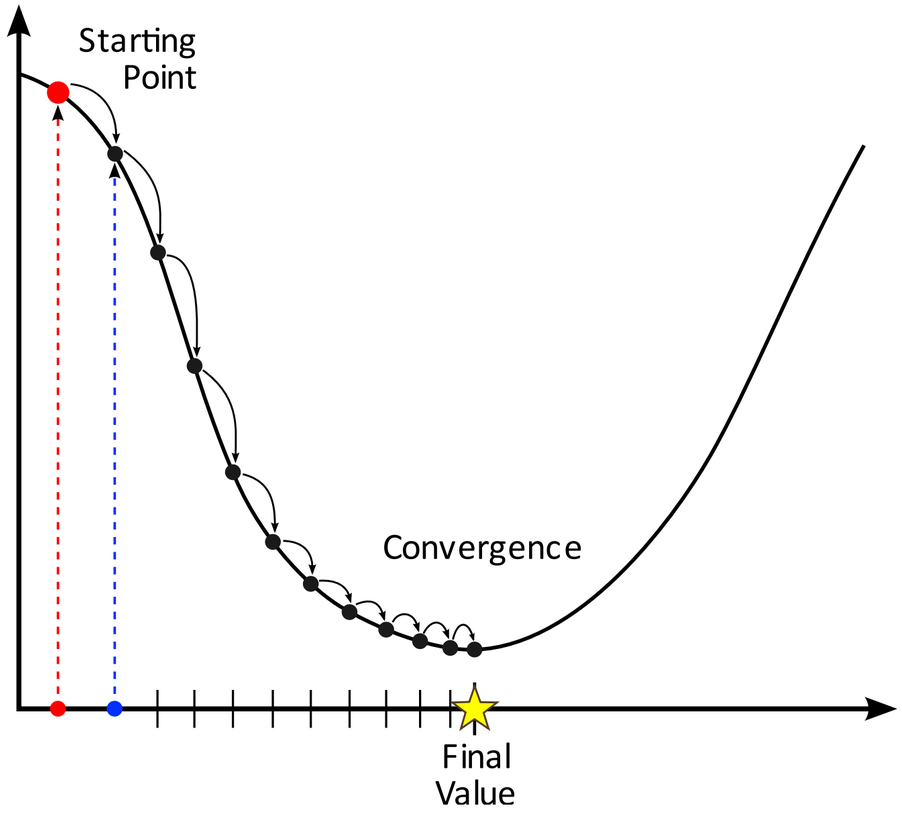
Acotada porque $J(\theta_k)$ es monótona decreciente y tiene un límite inferior (función convexa)

Entonces: $\sum_{k=0}^{\infty}\|\nabla J(\theta_k)\|^2 \leq \infty$. Por lo tanto: $\|\nabla J(\theta_k)\|^2 \to 0$ cuando $k \to \infty$

Esto implica que **cuando $k \to \infty$ se alcanza el mínimo global** ya que $J(\theta_k)$ es convexa y $\lim_{n \to k}\|\nabla J(\theta_k)\|^2 = 0$.



Gradient *descent*



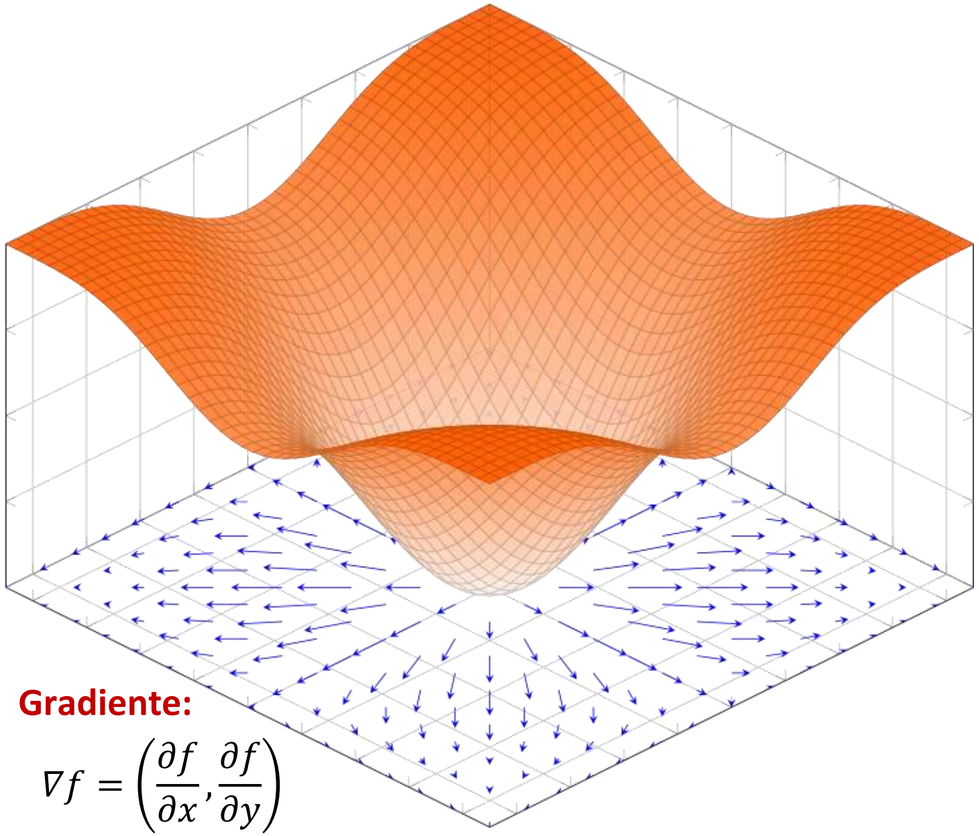
$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

donde θ_k representa los parámetros en la iteración k.

Se garantiza convergencia en el optimo global si:

- Si $J(\theta_k)$ es convexa.
- La gradiente $\nabla J(\theta_k)$ es Lipschitz continua.
- El learning rate está en el rango $0 < \alpha < \frac{2}{L}$

Gradient *descent*

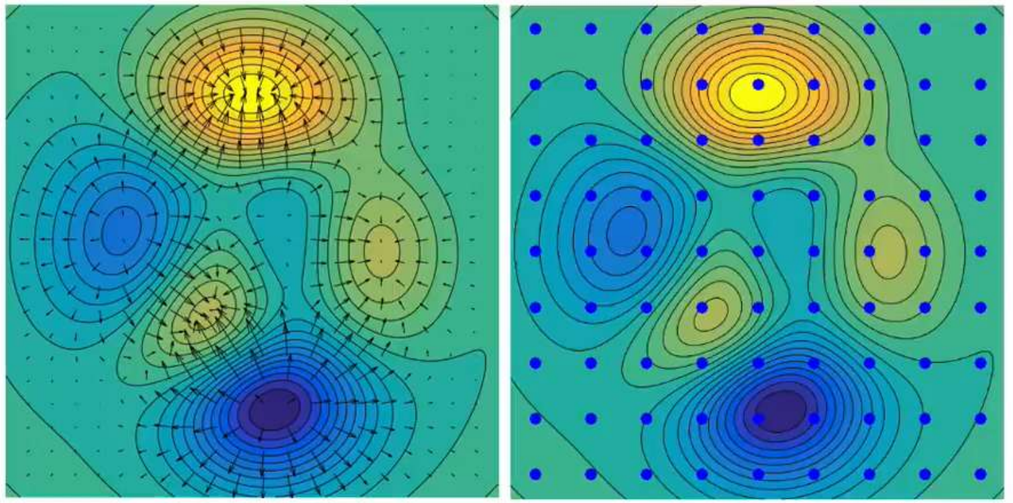


Gradiente:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

Gradient field: $\nabla f(x) \triangleq \left(\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \dots\right)$

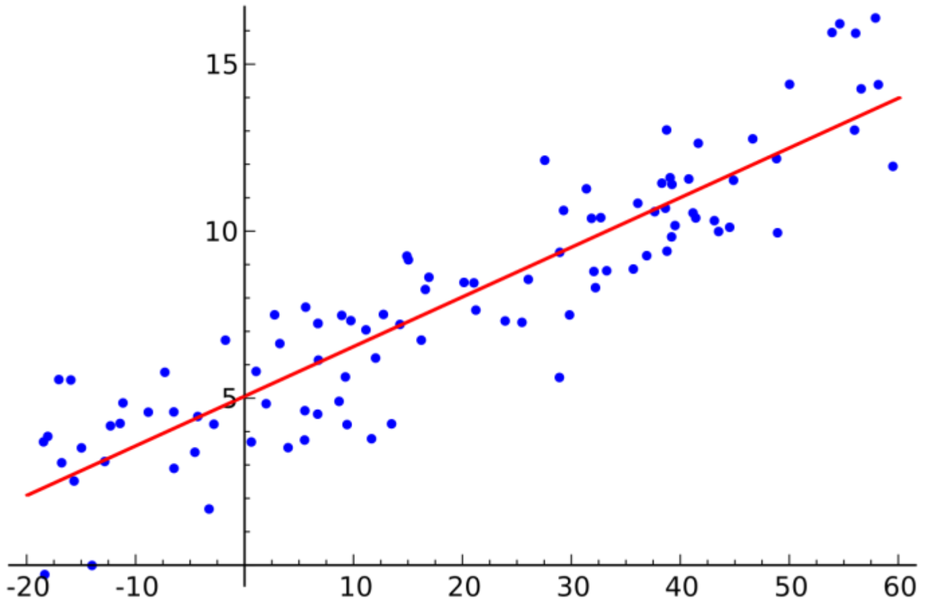
Gradient flow: $\frac{\mathrm{d}x}{\mathrm{d}t}(t) = -\nabla f(x(t))$



Gradient field ∇f

Gradient flows $x(t)$
$t = 0$ t medium t large

Linear *model*



number of features

noise term

independence assumption

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i \qquad \varepsilon_i \overset{iid}{\sim} \mathbf{N}(0, \sigma^2)$$

response

global intercept

feature j of observation i

coefficient for feature j

noise level

Generative *model*

Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad donde definimos tres objetos probabilísticos (variables aleatorias):

- $X: \Omega \to R^d$, la cual representa la variable de entrada.
- $w: \Omega \to R^d$, parámetros del modelo.
- $\epsilon: \Omega \to R$, una variable aleatoria tal que$\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A partir de estas variables, definimos una nueva variable aleatoria Y de la siguiente manera:

$$Y(\omega) = \sum_{j=1}^{d} w_j(\omega) X_j(\omega) + \epsilon(\omega) = w^{\top}(\omega) X(\omega) + \epsilon(\omega)$$

Generative *model*

Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad donde definimos tres objetos probabilísticos (variables aleatorias):

- $X: \Omega \to R^d$, la cual representa la variable de entrada.
- $w: \Omega \to R^d$, parámetros del modelo.
- $\epsilon: \Omega \to R$, una variable aleatoria tal que $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A partir de estas variables, definimos una nueva variable aleatoria Y de la siguiente manera:

$$Y(\omega) = \sum_{j=1}^{d} w_j(\omega) X_j(\omega) + \epsilon(\omega) = w^{\top}(\omega) X(\omega) + \epsilon(\omega)$$

Nosotros estudiamos $Y(\omega)$ para datos y pesos fijos, entonces estudiamos la variable aleatoria $p(Y|X(\omega) = x, w(\omega) = w)$.

Generative *model*

Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad donde definimos tres objetos probabilísticos (variables aleatorias):

- $X: \Omega \to R^d$, la cual representa la variable de entrada.
- $w: \Omega \to R^d$, parámetros del modelo.
- $\epsilon: \Omega \to R$, una variable aleatoria tal que $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A partir de estas variables, definimos una nueva variable aleatoria Y de la siguiente manera:

$$Y(\omega) = \sum_{j=1}^{d} w_j(\omega) X_j(\omega) + \epsilon(\omega) = w^{\top}(\omega) X(\omega) + \epsilon(\omega)$$

Nosotros estudiamos $Y(\omega)$ para datos y pesos fijos, entonces estudiamos la variable aleatoria $p(Y|X(\omega) = x, w(\omega) = w)$.

Sabemos que al aplicar una transformada lineal a una variable aleatoria normal obtenemos otra variable aleatoria normal, entonces:

$$p(y|x, w) \sim \mathcal{N}\left(\mu_y, \sigma_y^2\right)$$

Generative *model*

Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad donde definimos tres objetos probabilísticos (variables aleatorias):

- $X: \Omega \to R^d$, la cual representa la variable de entrada.
- $w: \Omega \to R^d$, parámetros del modelo.
- $\epsilon: \Omega \to R$, una variable aleatoria tal que$\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A partir de estas variables, definimos una nueva variable aleatoria Y de la siguiente manera:

$$Y(\omega) = \sum_{j=1}^{d} w_j(\omega) X_j(\omega) + \epsilon(\omega) = w^\top(\omega) X(\omega) + \epsilon(\omega)$$

Nosotros estudiamos $Y(\omega)$ para datos y pesos fijos, entonces estudiamos la variable aleatoria $p(Y|X(\omega) = x, w(\omega) = w)$.

Sabemos que al aplicar una transformada lineal a una variable aleatoria normal obtenemos otra variable aleatoria normal, entonces:

$$p(y|x, w) \sim \mathcal{N}\left(\mu_y, \sigma_y^2\right) = \mathcal{N}(w^\top x, \sigma^2)$$

- $\mu_y = E[y] = E[w^\top x + n] = w^\top x + E[n] = w^\top x + 0 = w^\top x$
- $\mathrm{Var}(y) = \mathrm{Var}(w^\top x + n) = \mathrm{Var}(n) = \sigma^2$

Generative *model*

Sea $(\Omega, \mathcal{F}, P)$ un espacio de probabilidad donde definimos tres objetos probabilísticos (variables aleatorias):

- $X: \Omega \to R^d$, la cual representa la variable de entrada.
- $w: \Omega \to R^d$, parámetros del modelo.
- $\epsilon: \Omega \to R$, una variable aleatoria tal que$\epsilon \sim \mathcal{N}(0, \sigma^2)$.

A partir de estas variables, definimos una nueva variable aleatoria Y de la siguiente manera:

$$Y(\omega) = \sum_{j=1}^{d} w_j(\omega) X_j(\omega) + \epsilon(\omega) = w^\top(\omega) X(\omega) + \epsilon(\omega)$$

Nosotros estudiamos $Y(\omega)$ para datos y pesos fijos, entonces estudiamos la variable aleatoria $p(Y|X(\omega) = x, w(\omega) = w)$.

Sabemos que al aplicar una transformada lineal a una variable aleatoria normal obtenemos otra variable aleatoria normal, entonces:

$$p(y|x, w) \sim \mathcal{N}(w^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y - w^\top x)^2}{2\sigma^2}}$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados los datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} p(w|D)$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados los datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} p(w|D)$$

Usando el teorema de Bayes, la distribución posterior se puede expresar como:

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)}$$

- $p(D \mid w)$: **Verosimilitud**, mide qué tan bien los datos D son explicados por los parámetros w.
- $p(w)$: **Prior**, refleja nuestras creencias iniciales sobre w antes de observar los datos.
- $p(D)$: **Evidencia**, es un factor de normalización que asegura que $p(w \mid D)$ sea una distribución de probabilidad válida.

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados los datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} p(w|D)$$

Usando el teorema de Bayes, la distribución posterior se puede expresar como:

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)}$$

- $p(D \mid w)$: **Verosimilitud**, mide qué tan bien los datos D son explicados por los parámetros w.
- $p(w)$: **Prior**, refleja nuestras creencias iniciales sobre w antes de observar los datos.
- $p(D)$: **Evidencia**, es un factor de normalización que asegura que $p(w \mid D)$ sea una distribución de probabilidad válida.

Reemplazando: $$\hat{w} = \arg\max_{w} p(w|D) = \arg\max_{w} \frac{p(D \mid w)p(w)}{p(D)}$$

Nota que la evidencia $p(D)$ no depende de w, por lo tanto podemos simplificar: $$\hat{w} = \arg\max_{w} p(D \mid w)p(w)$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \quad \boldsymbol{p(D \mid w)} \quad \boldsymbol{p(w)}$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\widehat{w} = \arg\max_{w} \quad \boldsymbol{p(D \mid w)} \quad \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = p(\,(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \mid w\,) = \prod_{i=1}^{N} p\big((x_i, y_i)|w\big)$$

Asumiendo que los datos son independientes.

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = p(\,(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \mid w\,) = \prod_{i=1}^{N} p\big((x_i, y_i)|w\big)$$

Desarrollamos $p\big((x_i, y_i)|w\big)$:

$$p\big((x_i, y_i)|w\big) = p(x_i|w)p(y_i|x_i, w) = p(x_i)p(y_i|x_i, w)$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Aplicamos logaritmo a $p(D|w)$:

$$\log p(D|w) = \sum_{i=1}^{N} \log p(x_i) + \sum_{i=1}^{N} \log p(y_i|x_i, w)$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Aplicamos logaritmo a $p(D|w)$:

$$\log p(D|w) = \sum_{i=1}^{N} \log p(x_i) + \sum_{i=1}^{N} \log p(y_i|x_i, w)$$

Nota que $\log p(x_i)$ es independiente de w, por lo que participa en MAP.

Además: $p(y|x, w) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y - w^\top x)^2}{2\sigma^2}}$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Aplicamos logaritmo a $p(D|w)$:

$$\log p(D|w) = \sum_{i=1}^{N} \log p(x_i) + \sum_{i=1}^{N} \log p(y_i|x_i, w)$$

$$= k_1 + \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w_i^{\top} x_i)^2}{2\sigma^2}} \right]$$

$$= k_1 - \frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - w_i^{\top} x_i)^2$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Aplicamos logaritmo a $p(D|w)$:

$$\log p(D|w) = \sum_{i=1}^{N} \log p(x_i) + \sum_{i=1}^{N} \log p(y_i|x_i, w)$$

$$= k_1 + \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w_i^\top x_i)^2}{2\sigma^2}} \right]$$

$$= k_1 - \frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - w_i^\top x_i)^2$$

Introducimos un prior gaussiano sobre w:

$$p(w) = \frac{1}{(2\pi\tau^2)^{d/2}} e^{-\frac{\|w\|^2}{2\tau^2}}$$

Aplicamos logaritmo a $p(w)$:

$$\log p(w) = -\frac{d}{2}\log(2\pi\tau^2) - \frac{\|w\|^2}{2\tau^2}$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \; \boldsymbol{p(D \mid w)} \; \boldsymbol{p(w)}$$

Expandimos:

$$p(D|w) = \prod_{i=1}^{N} p(x_i)\, p(y_i|x_i, w) = \prod_{i=1}^{N} p(x_i) \prod_{i=1}^{N} p(y_i|x_i, w)$$

Aplicamos logaritmo a $p(D|w)$:

$$\log p(D|w) = \sum_{i=1}^{N} \log p(x_i) + \sum_{i=1}^{N} \log p(y_i|x_i, w)$$

$$= k_1 + \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w_i^\top x_i)^2}{2\sigma^2}} \right]$$

$$= k_1 - \frac{N}{2}\log(2\pi\sigma^2) - \boldsymbol{\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - w_i^\top x_i)^2}$$

Introducimos un prior gaussiano sobre w:

$$p(w) = \frac{1}{(2\pi\tau^2)^{d/2}} e^{-\frac{\|w\|^2}{2\tau^2}}$$

Aplicamos logaritmo a $p(w)$:

$$\log p(w) = -\frac{d}{2}\log(2\pi\tau^2) - \boldsymbol{\frac{\|w\|^2}{2\tau^2}}$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\max_{w} \left[-\frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(y_i - w_i^\top x_i \right)^2 - \frac{\|w\|^2}{2\tau^2} \right]$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\widehat{w} = \arg\min_{w} \left[\frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(y_i - w_i^{\top} x_i \right)^2 + \frac{\|w\|^2}{2\tau^2} \right]$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

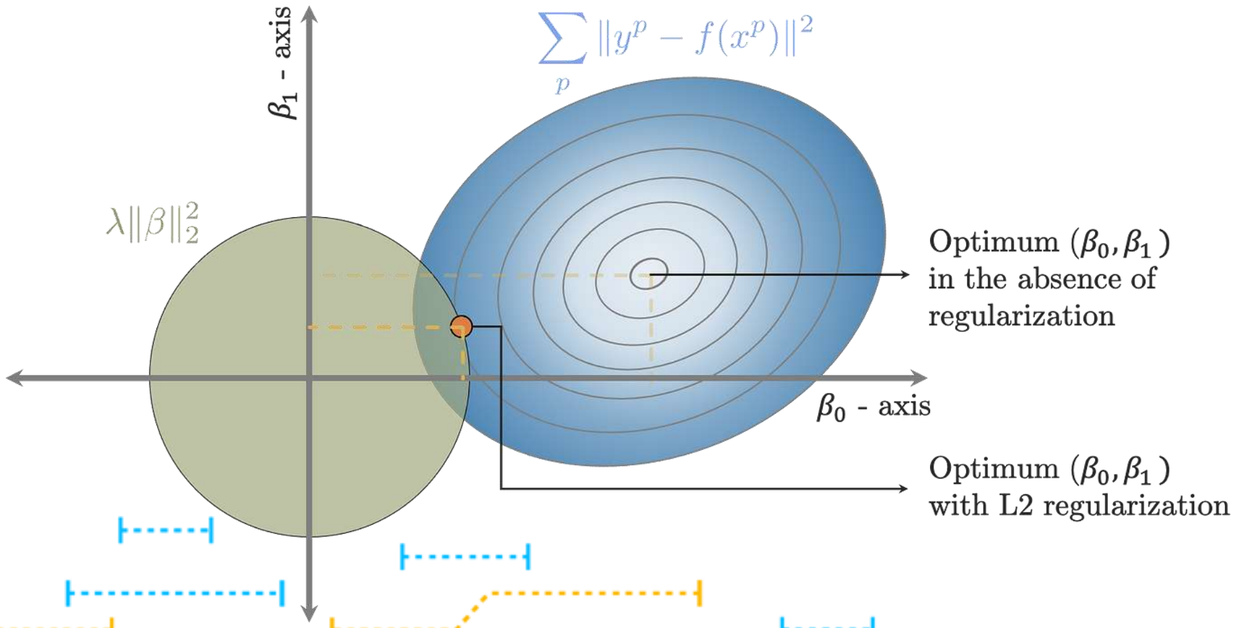
$$\hat{w} = \arg\min_{w} \frac{1}{2\sigma^2}\left[\sum_{i=1}^{N}\left(y_i - w_i^{\top} x_i\right)^2 + \frac{\sigma^2}{\tau^2}\|w\|^2\right]$$

Maximum a *Posteriori (MAP)*

Maximum a Posteriori (MAP) es un criterio estadístico que buscar los parámetros w que son más probables dados un conjunto de datos observados D. Buscamos:

$$\hat{w} = \arg\min_{w} \left[\sum_{i=1}^{N} \left(y_i - w_i^{\top} x_i \right)^2 + \lambda \|w\|^2 \right], \quad \text{donde } \lambda = \frac{\sigma^2}{\tau^2} \text{ controla la fuerza de la regularización.}$$

Función Loss **Término regularizador**



Función *Loss*

Distribución del ruido	Densidad de probabilidad	Función Loss	Propiedades	Aplicaciones típicas
Gaussiana ($\mathcal{N}(0, \sigma^2)$)	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{n^2}{2\sigma^2}\right)$	$\sum_i (y_i - w^\top x_i)^2$ Mínimos cuadrados	Penaliza errores cuadráticamente, sensible a valores atípicos.	Regresión estándar, modelos lineales.
Laplaciano (Laplace$(0, b)$)	$\frac{1}{2b} \exp\left(-\frac{\lvert n\rvert}{b}\right)$	$\sum_i \lvert y_i - w^\top x_i\rvert$ Error absoluto medio	Penaliza proporcionalmente, robusta a valores atípicos.	Análisis robusto, series temporales.
Cauchy (Cauchy$(0, \gamma)$)	$\frac{1}{\pi\gamma\left(1 + \frac{n^2}{\gamma^2}\right)}$	$\sum_i \log\left(1 + \frac{(y_i - w^\top x_i)^2}{\gamma^2}\right)$ Cauchy loss	Colas largas: alta robustez a valores atípicos.	Situaciones con ruido extremo o datos ruidosos.
Uniforme (Uniform$(-c, c)$)	$\frac{1}{2c}, \quad \lvert n\rvert \le c$	$\max(0, \lvert y_i - w^\top x_i\rvert - c)^2$ Huber-like	Penaliza solo errores dentro de un rango, ignora errores extremos.	Problemas con restricciones en los errores.
Poisson (Pois(λ))	$\lambda^k \frac{e^{-\lambda}}{k!}, \quad k \ge 0$	$\sum_i \max\left(0, \lambda(y_i - w^\top x_i)\right)$ Asimétrica	Penalización direccional: favorece errores positivos o negativos.	Modelos predictivos sesgados, distribuciones asimétricas.

Regularization

Distribución del ruido	Densidad de probabilidad	Termino de regularización	Propiedades	Aplicaciones típicas
Gaussiana ($\mathcal{N}(0,\tau^2)$)	$\prod_j \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{w_j^2}{2\tau^2}\right)$	$\lambda\|\|w\|\|^2$ ℓ_2 regularización	Penaliza magnitudes grandes de w. Suavidad.	Regresión Ridge, modelos lineales regulares.
Laplaciana (Laplace$(0,b)$)	$\prod_j \frac{1}{2b} \exp\left(-\frac{\|w_j\|}{b}\right)$	$\lambda\|\|w\|\|_1$ ℓ_1 regularización	Favorece sparsidad. Algunos w_j se vuelven exactamente cero.	Lasso, selección de características.
Cauchy (Cauchy$(0,\gamma)$)	$\prod_j \frac{1}{\pi\gamma\left(1+\frac{w_j^2}{\gamma^2}\right)}$	$\sum_j \log\left(1+\frac{w_j^2}{\gamma^2}\right)$ Cauchy regularization	Penalización suave para valores grandes de w. Robusta a outliers en w.	Regularización robusta, situaciones con colas largas.
Spike-and-Slab	$\pi\delta(w_j) + (1-\pi)\mathcal{N}(w_j \mid 0,\tau^2)$	Mixta: combina sparsidad (ℓ_0) y magnitud (ℓ_2).	Algunos w_j se fijan exactamente a cero. Sparsidad estricta.	Modelos bayesianos, selección automática de características.
t-Student	$\prod_j \left(1+\frac{w_j^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	$\sum_j \log\left(1+\frac{w_j^2}{\nu}\right)$ t-Student regularization	Controla robustez mediante ν. Interpola entre ℓ_1 y ℓ_2.	Modelos con datos ruidosos o parámetros dispersos.
Poisson (Pois(λ))	$\prod_j \lambda \exp(-\lambda w_j), \quad w_j \geq 0$	$\sum_j w_j$ Regularización no negativa	Penaliza magnitudes positivas. Restringe $w_j \geq 0$.	Optimización con restricciones no negativas.

Learning *theory*

Expected risk

Dada una función $h: \mathcal{X} \to \mathcal{Y}$ que pertenece a un espacio de hipótesis $\mathcal{H}$; una función de pérdida $\mathcal{L}: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$; y una base de datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ con distribución de probabilidad p; definimos el riegos esperado de f como:

$$\mathcal{R}(h) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\mathcal{L}(h(X), Y)] = \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(h(x), y)\partial p(x, y)$$

- $\mathcal{X}$: Espacio de entrada.
- $\mathcal{Y}$: Espacio de salida.

Learning *theory*

Expected risk

Dada una función $h: \mathcal{X} \to \mathcal{Y}$ que pertenece a un espacio de hipótesis $\mathcal{H}$; una función de pérdida $\mathcal{L}: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$; y una base de datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ con distribución de probabilidad p; definimos el riegos esperado de f como:

$$\mathcal{R}(h) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\mathcal{L}(h(X), Y)] = \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(h(x), y)\partial p(x, y)$$

- $\mathcal{X}$: Espacio de entrada.
- $\mathcal{Y}$: Espacio de salida.

Empirical risk

Dada una función $h: \mathcal{X} \to \mathcal{Y}$ que pertenece a un espacio de hipótesis $\mathcal{H}$; una función de pérdida $\mathcal{L}: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$; y una base de datos $(x_i, y_i): \mathcal{X} \times \mathcal{Y}, i = 1, \dots, n$; definimos el riegos empírico de f como:

$$\mathcal{R}(h) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\mathcal{L}(h(X), Y)] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(x_i), y_i)$$

Learning *theory*

Supongamos que los datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ están definidos mediante la relación $Y = f(X) + \epsilon$, donde:

- $f(X)$ es la función verdadera (o esperanza condicional $f(X) = \mathbb{E}[\, Y \mid X \,]$).
- ϵ es un ruido independiente de X, con $\mathbb{E}[\epsilon] = 0$ y $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$.

Sea $\hat{h}_S(X)$ el predictor construido a partir de una muestra $S \subseteq \mathcal{D}$. Si consideramos $\mathcal{L}\left(\hat{h}_S(x_i), y_i\right) = \left(y_i - \hat{h}_S(x_i)\right)^2$, entonces el riesgo condicional para un predictor entrenado en S está dado por: :

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_{(X,Y)}\left[\left(Y - \hat{h}_S(X)\right)^2\right]$$

Learning *theory*

Supongamos que los datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ están definidos mediante la relación $Y = f(X) + \epsilon$, donde:

- $f(X)$ es la función verdadera (o esperanza condicional $f(X) = \mathbb{E}[\, Y \mid X \,]$).
- ϵ es un ruido independiente de X, con $\mathbb{E}[\epsilon] = 0$ y $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$.

Sea $\hat{h}_S(X)$ el predictor construido a partir de una muestra $S \subseteq \mathcal{D}$. Si consideramos $\mathcal{L}\left(\hat{h}_S(x_i), y_i\right) = \left(y_i - \hat{h}_S(x_i)\right)^2$, entonces el riesgo condicional para un predictor entrenado en S está dado por: :

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_{(X,Y)}\left[\left(Y - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_{(X,Y)}\left[\left(f(X) + \epsilon - \hat{h}_S(X)\right)^2\right]$$

Learning *theory*

Supongamos que los datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ están definidos mediante la relación $Y = f(X) + \epsilon$, donde:

- $f(X)$ es la función verdadera (o esperanza condicional $f(X) = \mathbb{E}[\, Y \mid X \,]$).
- ϵ es un ruido independiente de X, con $\mathbb{E}[\epsilon] = 0$ y $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$.

Sea $\hat{h}_S(X)$ el predictor construido a partir de una muestra $S \subseteq \mathcal{D}$. Si consideramos $\mathcal{L}\left(\hat{h}_S(x_i), y_i\right) = \left(y_i - \hat{h}_S(x_i)\right)^2$, entonces el riesgo condicional para un predictor entrenado en S está dado por: :

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_{(X,Y)}\left[\left(Y - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_{(X,Y)}\left[\left(f(X) + \epsilon - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2 + 2\epsilon\left(f(X) - \hat{h}_S(X)\right) + \epsilon^2\right]$$

$$= \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right] + \mathbb{E}_X\left[2\epsilon\left(f(X) - \hat{h}_S(X)\right)\right] + \mathbb{E}_X[\epsilon^2]$$

Learning *theory*

Supongamos que los datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ están definidos mediante la relación $Y = f(X) + \epsilon$, donde:

- $f(X)$ es la función verdadera (o esperanza condicional $f(X) = \mathbb{E}[\, Y \mid X \,]$).
- ϵ es un ruido independiente de X, con $\mathbb{E}[\epsilon] = 0$ y $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$.

Sea $\hat{h}_S(X)$ el predictor construido a partir de una muestra $S \subseteq \mathcal{D}$. Si consideramos $\mathcal{L}\left(\hat{h}_S(x_i), y_i\right) = \left(y_i - \hat{h}_S(x_i)\right)^2$, entonces el riesgo condicional para un predictor entrenado en S está dado por: :

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_{(X,Y)}\left[\left(Y - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_{(X,Y)}\left[\left(f(X) + \epsilon - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2 + 2\epsilon\left(f(X) - \hat{h}_S(X)\right) + \epsilon^2\right]$$

$$= \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right] + \mathbb{E}_X\left[2\epsilon\left(f(X) - \hat{h}_S(X)\right)\right] + \mathbb{E}_X[\epsilon^2]$$

Usamos las propiedades de la esperanza matemática:

- $\mathbb{E}[\epsilon] = 0 \;\rightarrow\; \mathbb{E}\left[\epsilon\left(f(X) - \hat{h}_S(X)\right)\right] = \left(f(X) - \hat{h}_S(X)\right)\mathbb{E}[\epsilon] = 0.$
- $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2.$

Por lo tanto, el riesgo condicional se puede reescribir como:

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right] + \sigma_\epsilon^2$$

Learning *theory*

Supongamos que los datos $\mathcal{D}: \mathcal{X} \times \mathcal{Y}$ están definidos mediante la relación $Y = f(X) + \epsilon$, donde:

- $f(X)$ es la función verdadera (o esperanza condicional $f(X) = \mathbb{E}[\, Y \mid X \,]$).
- ϵ es un ruido independiente de X, con $\mathbb{E}[\epsilon] = 0$ y $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2$.

Sea $\hat{h}_S(X)$ el predictor construido a partir de una muestra $S \subseteq \mathcal{D}$. Si consideramos $\mathcal{L}\left(\hat{h}_S(x_i), y_i\right) = \left(y_i - \hat{h}_S(x_i)\right)^2$, entonces el riesgo condicional para un predictor entrenado en S está dado por: :

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_{(X,Y)}\left[\left(Y - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_{(X,Y)}\left[\left(f(X) + \epsilon - \hat{h}_S(X)\right)^2\right] = \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2 + 2\epsilon\left(f(X) - \hat{h}_S(X)\right) + \epsilon^2\right]$$

$$= \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right] + \mathbb{E}_X\left[2\epsilon\left(f(X) - \hat{h}_S(X)\right)\right] + \mathbb{E}_X[\epsilon^2]$$

Usamos las propiedades de la esperanza matemática:

- $\mathbb{E}[\epsilon] = 0 \;\rightarrow\; \mathbb{E}\left[\epsilon\left(f(X) - \hat{h}_S(X)\right)\right] = \left(f(X) - \hat{h}_S(X)\right)\mathbb{E}[\epsilon] = 0.$
- $\mathbb{E}[\epsilon^2] = \sigma_\epsilon^2.$

Por lo tanto, el riesgo condicional se puede reescribir como:

$$\mathcal{R}\left(\hat{h}_S\right) = \mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right] + \sigma_\epsilon^2$$

Discrepancia entre $f(X)$ y $\hat{h}_S(X)$. **Contribución del ruido irreducible en los datos.**

Learning *theory*

Ahora calculamos el riesgo esperado:

$$\mathbb{E}_S[\mathcal{R}(\hat{h}_S)] = \mathbb{E}_S\left[\mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2 = \mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2$$

Learning *theory*

Ahora calculamos el riesgo esperado:

$$\mathbb{E}_S[\mathcal{R}(\hat{h}_S)] = \mathbb{E}_S\left[\mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2 = \mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2$$

Despejamos $\left(f(X) - \hat{h}_S(X)\right)^2$:

$$\left(f(X) - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)] + \mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2 + 2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right) + \left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2$$

Learning *theory*

Ahora calculamos el riesgo esperado:

$$\mathbb{E}_S[\mathcal{R}(\hat{h}_S)] = \mathbb{E}_S\left[\mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2 = \mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2$$

Despejamos $\left(f(X) - \hat{h}_S(X)\right)^2$:

$$\left(f(X) - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)] + \mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2 + 2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right) + \left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2$$

Ahora despejamos la esperanza matemática de cada termino de la ecuación

- **Primer termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]\right] = \mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]$

  Ya que $\mathbb{E}_S[\hat{h}_S(X)]$ es una constante con respecto a S, y $f(X)$ no depende de S.

Learning *theory*

Ahora calculamos el riesgo esperado:

$$\mathbb{E}_S[\mathcal{R}(\hat{h}_S)] = \mathbb{E}_S\left[\mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2 = \mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2$$

Despejamos $\left(f(X) - \hat{h}_S(X)\right)^2$:

$$\left(f(X) - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)] + \mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2 + 2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right) + \left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2$$

Ahora despejamos la esperanza matemática de cada termino de la ecuación

- **Primer termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]\right] = \mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]$

  Ya que $\mathbb{E}_S[\hat{h}_S(X)]$ es una constante con respecto a S, y $f(X)$ no depende de S.

- **Segundo termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)\right]\right] = 0$

  Son constantes con respecto a S. $\mathbb{E}_S[\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)] = \mathbb{E}_S[\hat{h}_S(X)] - \mathbb{E}_S[\hat{h}_S(X)]$=0

Learning *theory*

Ahora calculamos el riesgo esperado:

$$\mathbb{E}_S[\mathcal{R}(\hat{h}_S)] = \mathbb{E}_S\left[\mathbb{E}_X\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2 = \mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \hat{h}_S(X)\right)^2\right]\right] + \sigma_\epsilon^2$$

Despejamos $\left(f(X) - \hat{h}_S(X)\right)^2$:

$$\left(f(X) - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)] + \mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2 = \left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2 + 2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right) + \left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2$$

Ahora despejamos la esperanza matemática de cada termino de la ecuación

- **Primer termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]\right] = \mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]$

  Ya que $\mathbb{E}_S[\hat{h}_S(X)]$ es una constante con respecto a S, y $f(X)$ no depende de S.

- **Segundo termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[2\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)\right]\right] = 0$

  Son constantes con respecto a S. $\mathbb{E}_S[\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)] = \mathbb{E}_S[\hat{h}_S(X)] - \mathbb{E}_S[\hat{h}_S(X)]$=0

- **Tercer termino:** $\mathbb{E}_X\left[\mathbb{E}_S\left[\left(\mathbb{E}_S[\hat{h}_S(X)] - \hat{h}_S(X)\right)^2\right]\right] = \mathbb{E}_X\left[\mathrm{Var}_S[\hat{h}_S(X)]\right]$

Bias-variance-noise *decomposition*

Finalmente:

$$\mathbb{E}_S\left[\mathcal{R}\left(\hat{h}_S\right)\right] = \underbrace{\mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_X\left[\mathbb{E}_S\left[\left(\hat{h}_S(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\sigma_\epsilon^2}_{\text{Noise}}$$

Bias-variance-noise *decomposition*

Finalmente:

$$\mathbb{E}_S\left[\mathcal{R}(\hat{h}_S)\right] = \underbrace{\mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_X\left[\mathbb{E}_S\left[\left(\hat{h}_S(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\sigma_\epsilon^2}_{\text{Noise}}$$

- **Bias²** $= \mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]$

  Mide la diferencia entre la verdadera función objetivo $f(X)$ y la hipótesis promedio $\mathbb{E}_S\left[\hat{h}_S(X)\right]$. Representa el error sistemático debido a la incapacidad del modelo para aproximarse a $f(X)$.

- **Variance** $= \mathbb{E}_X\left[\mathbb{E}_S\left[\left(\hat{h}_S(X) - \mathbb{E}_S\left[\hat{h}_S(X)\right]\right)^2\right]\right]$

  Cuantifica la sensibilidad del modelo a las fluctuaciones en la muestra de entrenamiento S. Un valor alto indican que el modelo cambia significativamente al cambiar de muestras.

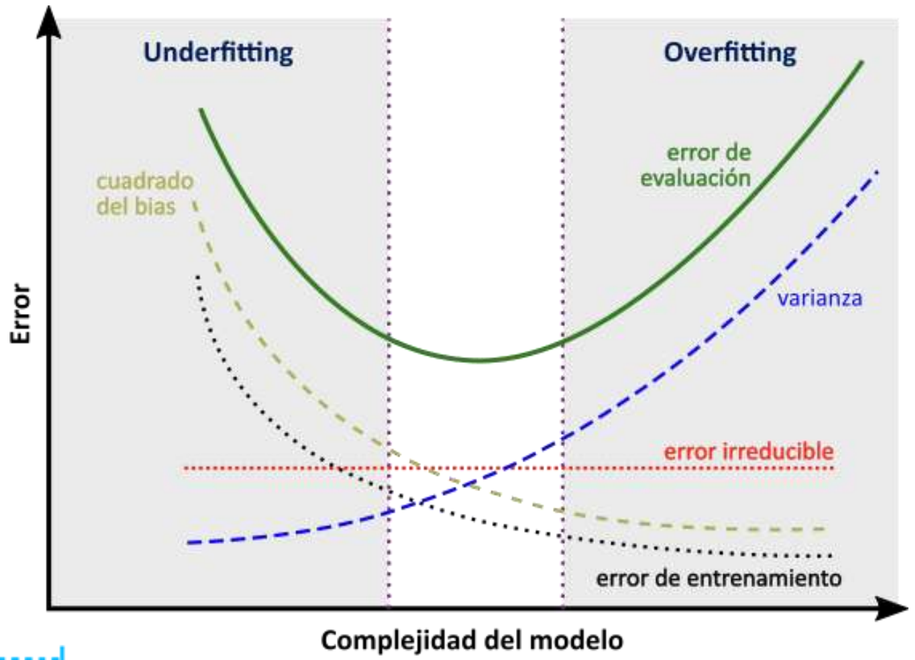
- **Noise** $= \sigma_\epsilon^2$

  Es un término irreducible que depende únicamente de la varianza intrínseca del ruido en los datos.

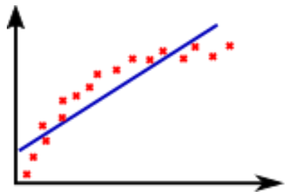
Bias–variance *tradeoff*

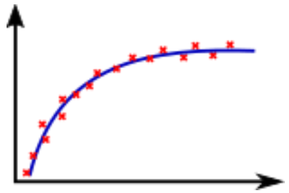
Finalmente:

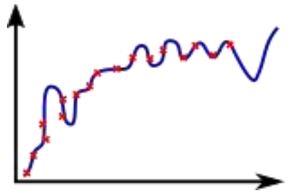
$$\mathbb{E}_S\left[\mathcal{R}(\hat{h}_S)\right] = \underbrace{\mathbb{E}_X\left[\left(f(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_X\left[\mathbb{E}_S\left[\left(\hat{h}_S(X) - \mathbb{E}_S[\hat{h}_S(X)]\right)^2\right]\right]}_{\text{Variance}} + \underbrace{\sigma_\epsilon^2}_{\text{Noise}}$$

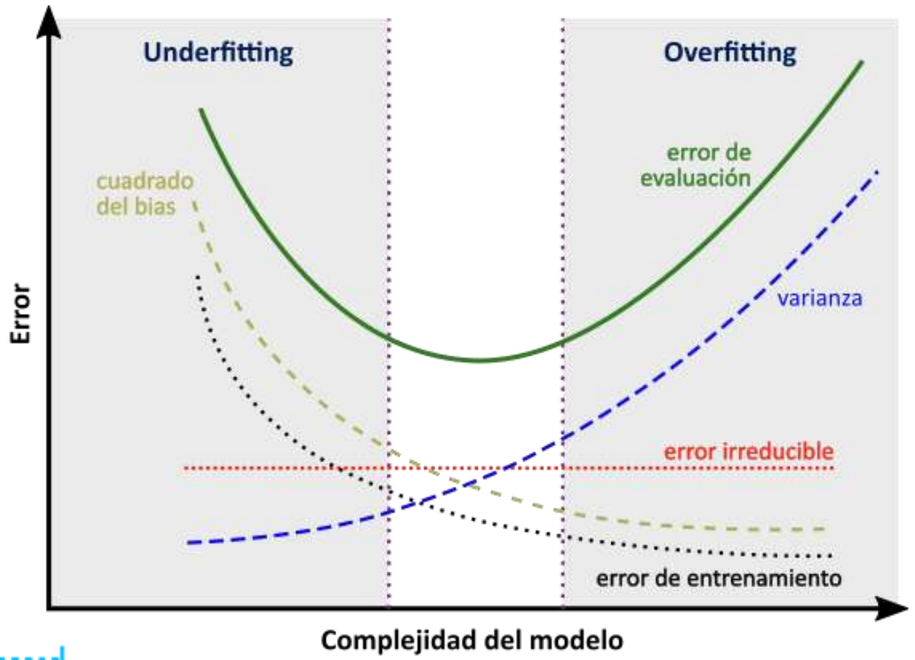


Bias–variance *tradeoff*

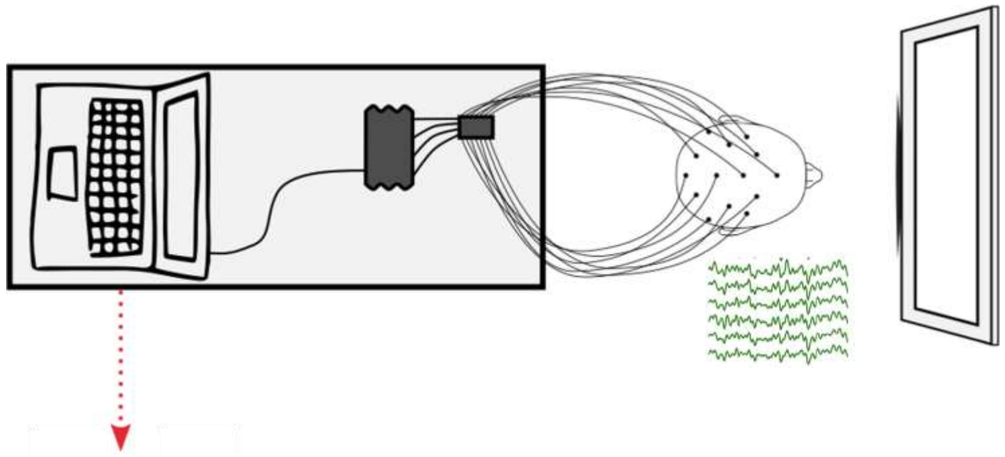








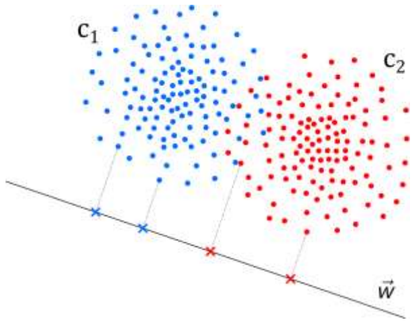
Modeling



Feature extraction

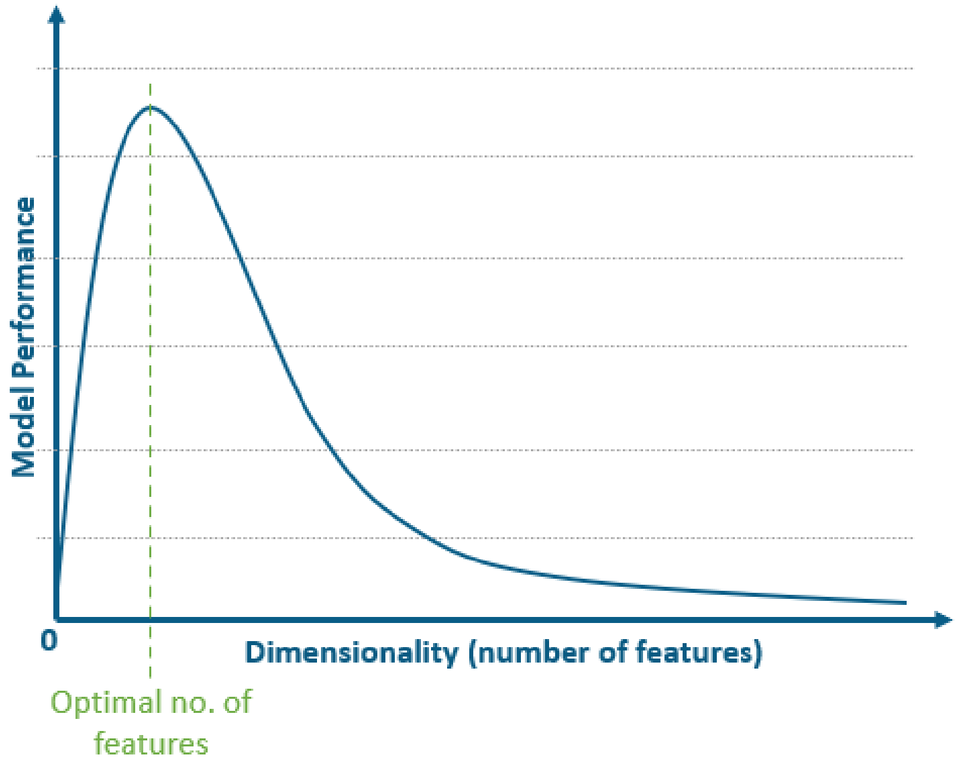
- Raw data
- Standardized moment
- Hjorth parameters
- Discrete Fourier coefficients
- Discrete wavelet coefficients

Linear Model



Curse of *dimensionality*

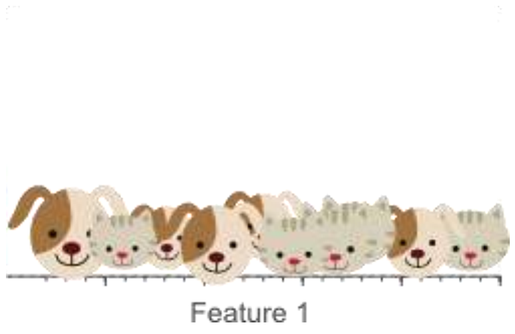
Hughes Phenomenon



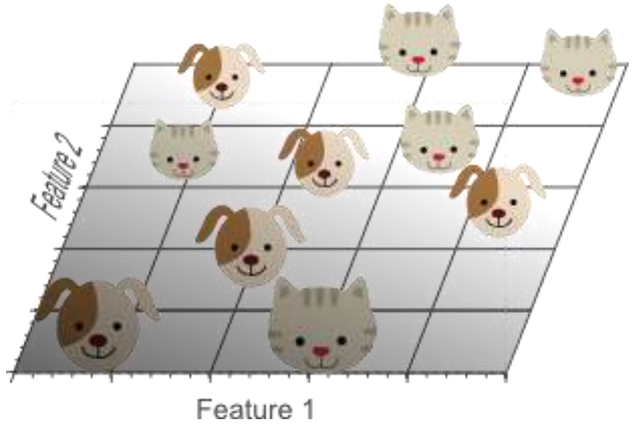
Curse of *dimensionality*

Sparse data space

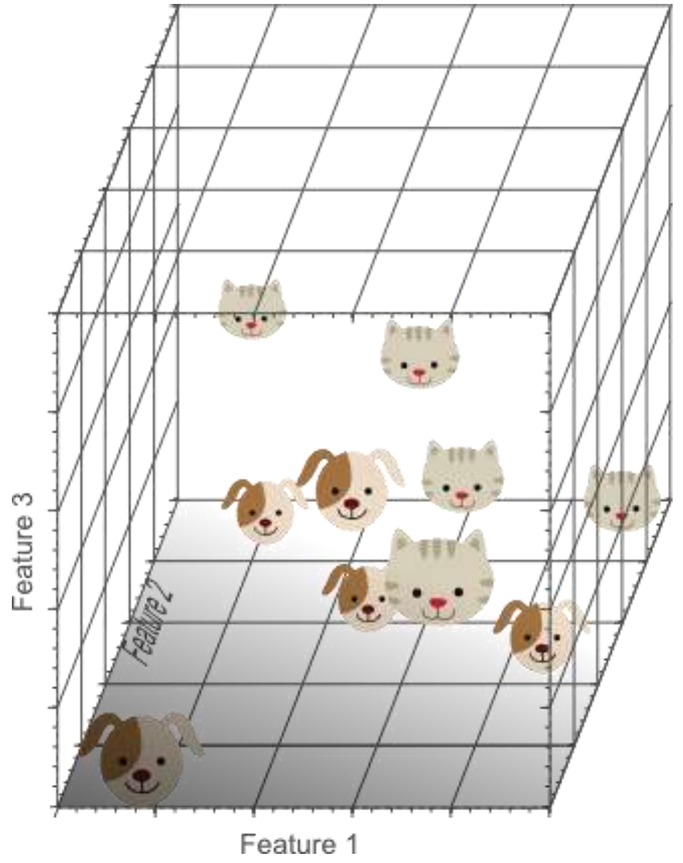
Disminución de la Densidad de Datos



Una dimensión

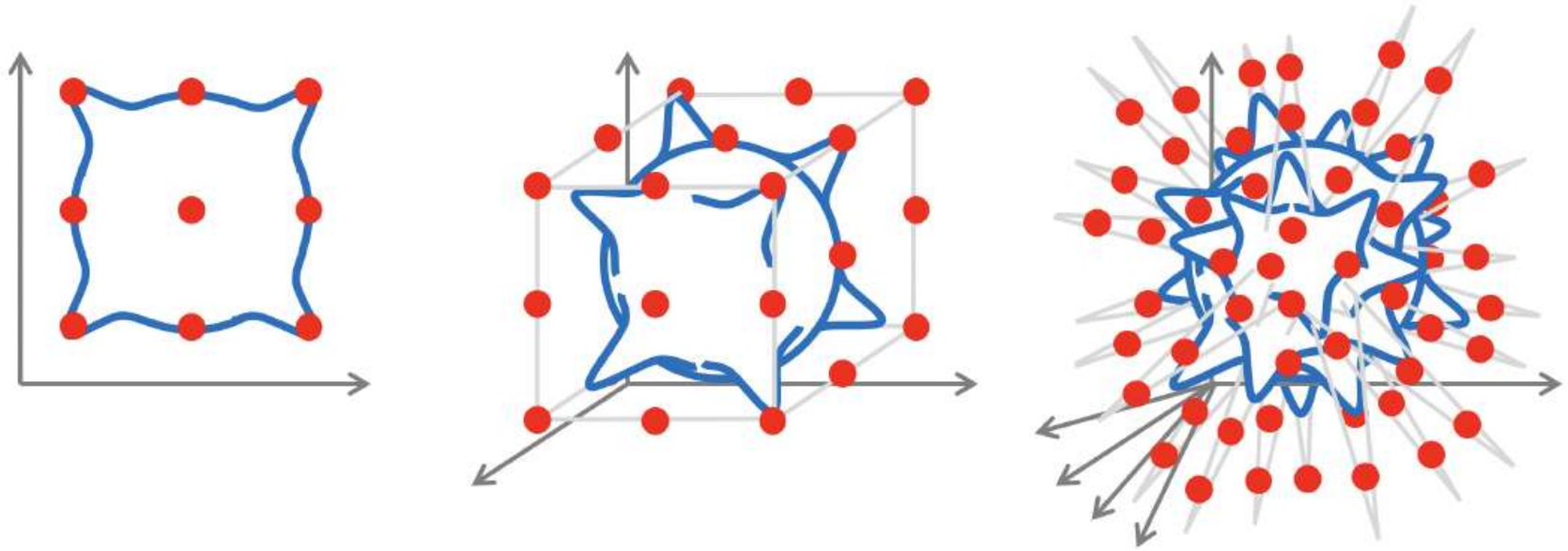


Dos dimensiones



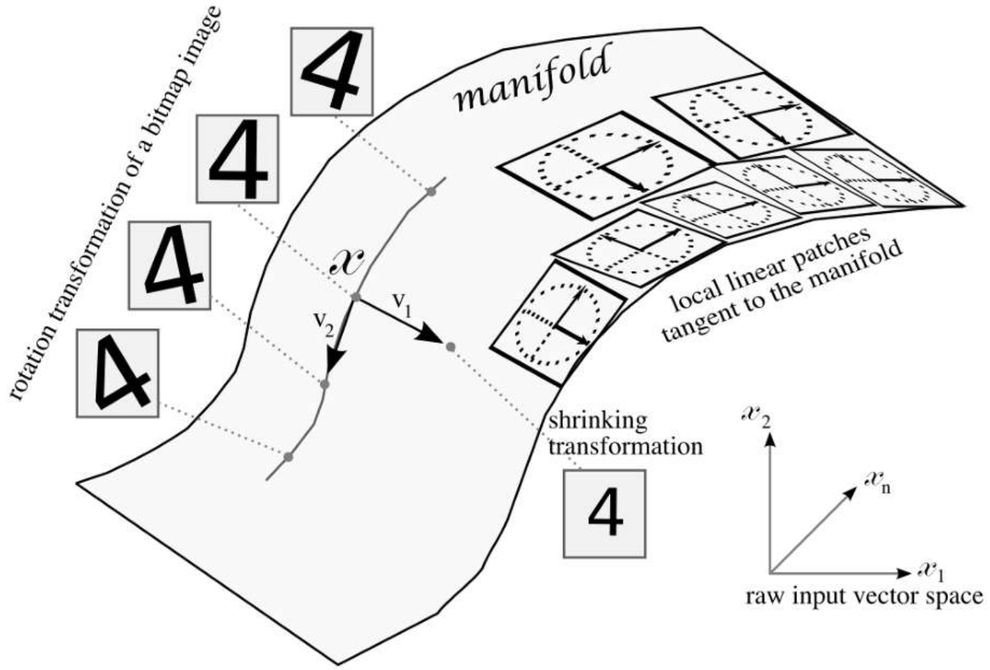
Tres dimensiones

Curse of *dimensionality*



Manifold *hypothesis*

En aplicaciones de machine learning se suele partir de datos en un espacio de alta dimensión $\mathbb{R}^D$. Manifold Hypothesis plantea que, aunque los datos se describan con muchas features (D dimensiones), estos en realidad "viven" en (o muy cerca de) un subconjunto de dimensión intrínseca mucho menor que D. Dicho subconjunto se denomina manifold.

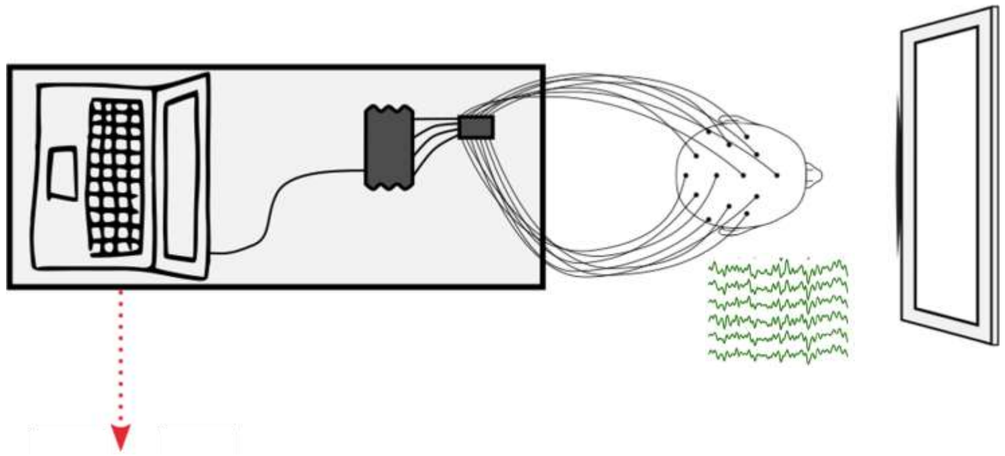


Existe un manifold $M \subset R^D$ de dimensión $d \ll D$ tal que:

$$\mathcal{D}(\{x \in R^D : d(\boldsymbol{x}, M) \leq \epsilon\}) \approx 1,$$

para algún $\epsilon > 0$ pequeño y donde $d(\boldsymbol{x}, M)$ es la distancia mínima de $\boldsymbol{x}$ a M. Así, casi todos los datos se encuentran dentro de un "tubo" de radio ϵ alrededor de M.

Modeling



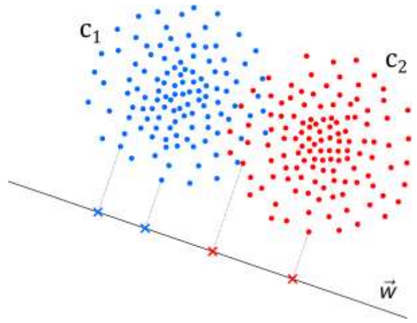
Feature extraction

- Raw data
- Standardized moment
- Hjorth parameters
- Discrete Fourier coefficients
- Discrete wavelet coefficients

Dimensionality reduction

Feature Selection

Linear Model



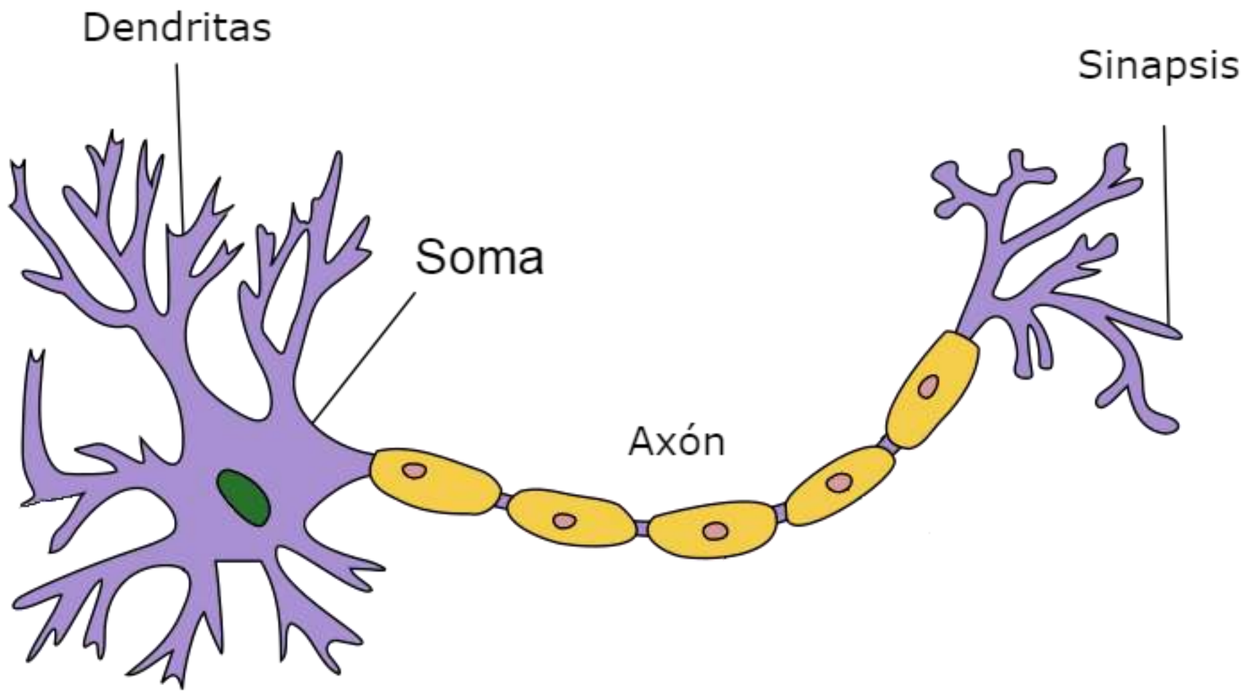
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

3.

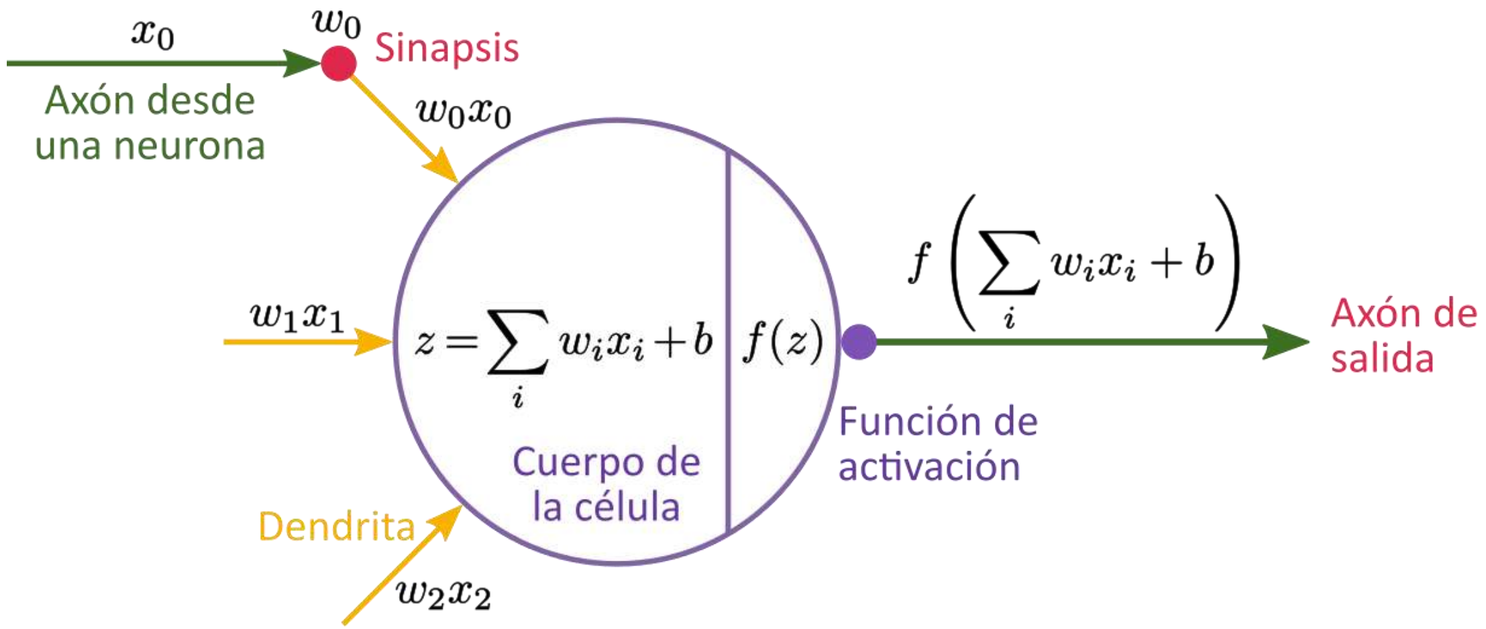
Neural *networks*

TRANSFORMATEC

Neurona

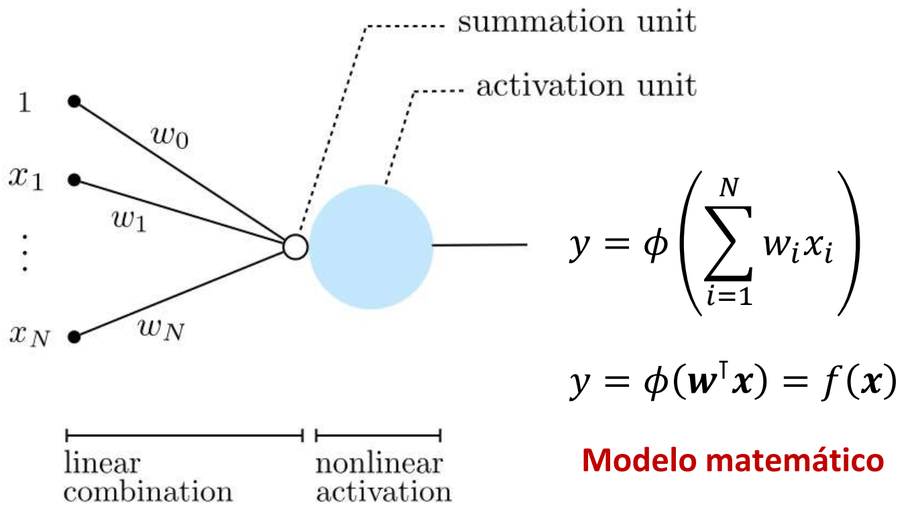


Artificial neural network



Frank Rosenblatt (1958) "The perceptron: a probabilistic model for information storage and organization in the brain". Psychological review 65.6 (1958): 386.

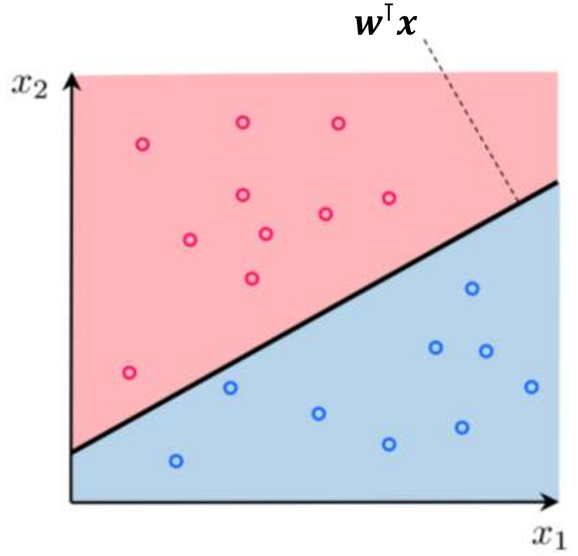
Perceptron



$$y = \phi\left(\sum_{i=1}^{N} w_i x_i\right)$$

$$y = \phi(\boldsymbol{w}^\intercal \boldsymbol{x}) = f(\boldsymbol{x})$$

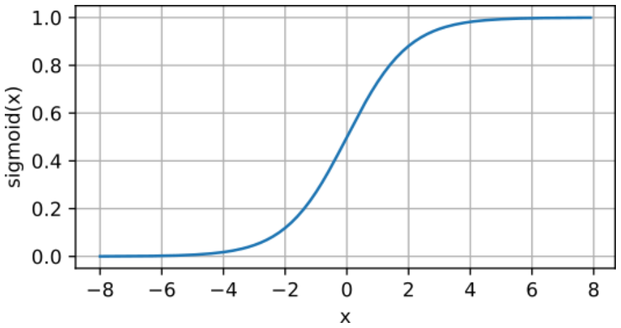
Modelo matemático



Frank Rosenblatt (1958) "The perceptron: a probabilistic model for information storage and organization in the brain". Psychological review 65.6 (1958): 386.

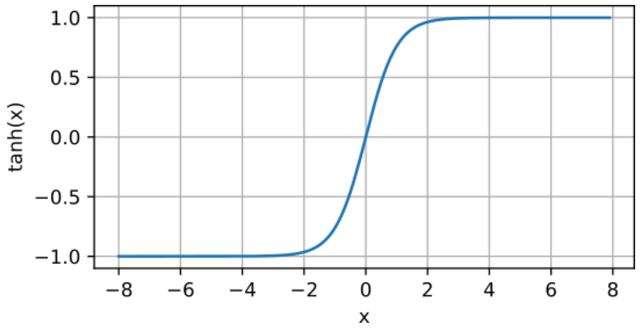
Activation *Function*

Sigmoid



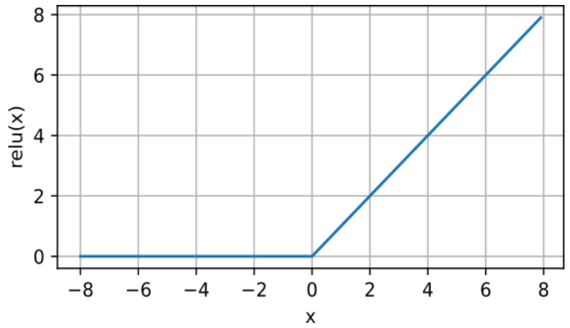
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Hyperbolic tangent



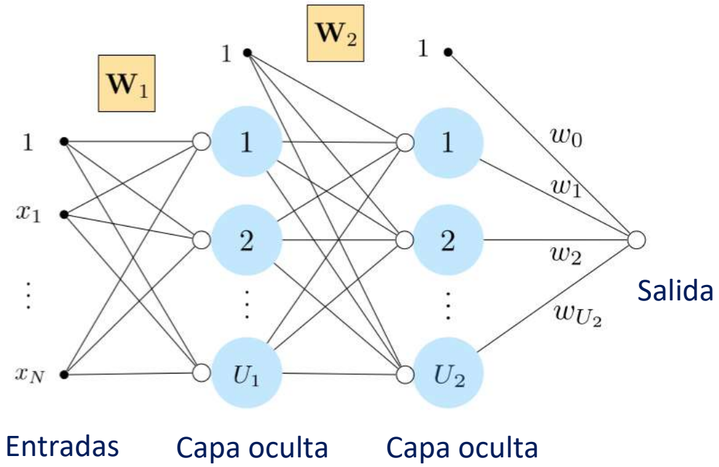
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Rectified Linear Unit



$$\mathrm{ReLU}(x) = \max\{0, x\}$$

Multilayer *perceptron*

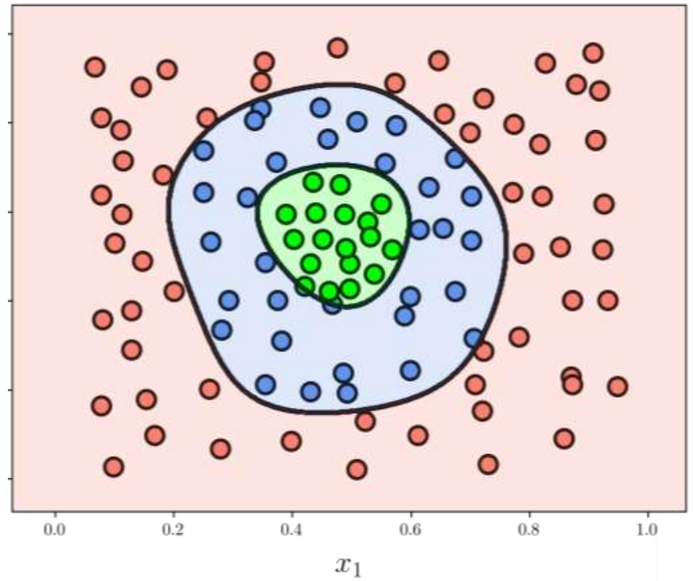


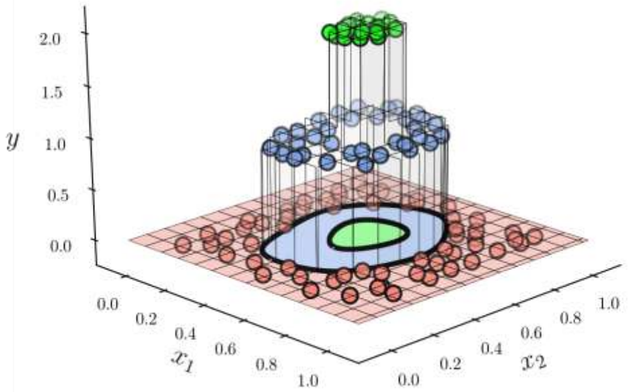
Single layer unit

$$f_j^{(l)} = \phi\left(w_{0,j}^{(l)} + \sum_{i=1}^{U_{l-1}} w_{i,j}^{(l)} f_i^{(l-1)}(\boldsymbol{x})\right)$$

Model

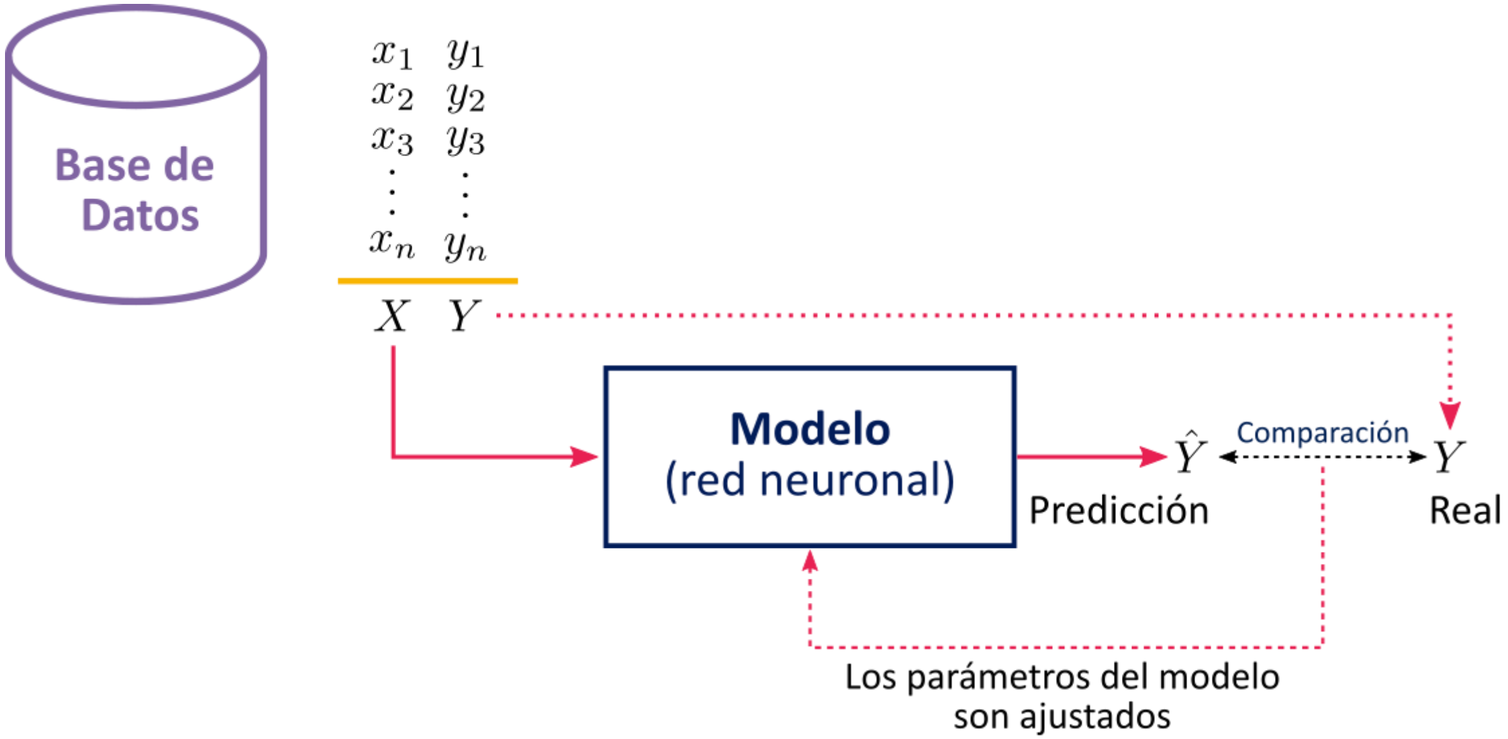
$$y = f_2(f_1(\boldsymbol{x}))$$
$$y = (f_2 \circ f_1)(\boldsymbol{x})$$



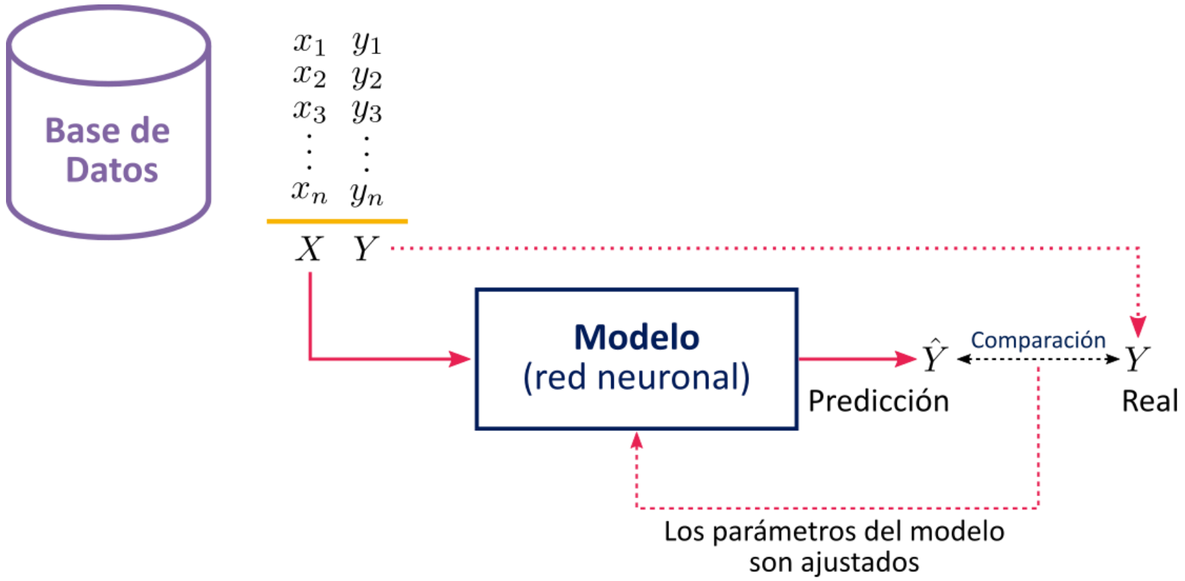


TRANSFORMATEC

Training

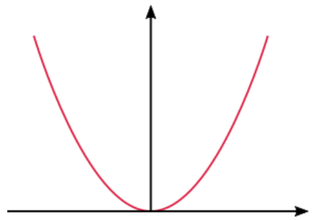

Base de Datos
x_1 y_1
x_2 y_2
x_3 y_3
x_n y_n
X Y
Modelo
(red neuronal)
$\hat{Y}$
Comparación
Y
Predicción
Real
Los parámetros del modelo son ajustados

Training



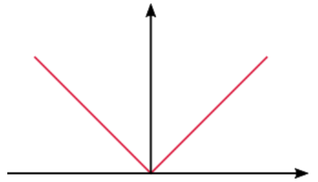
Error cuadrático medio

Mean squared error (MSE) $\mathcal{L}(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$



Error absoluto medio

Mean absolute error (MAE) $\mathcal{L}(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

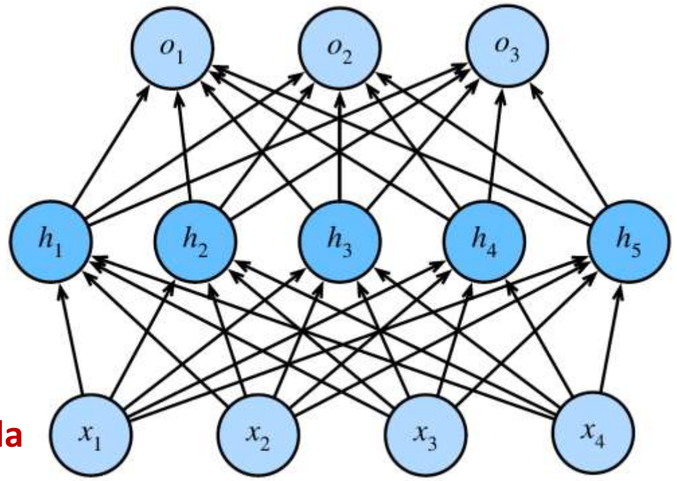


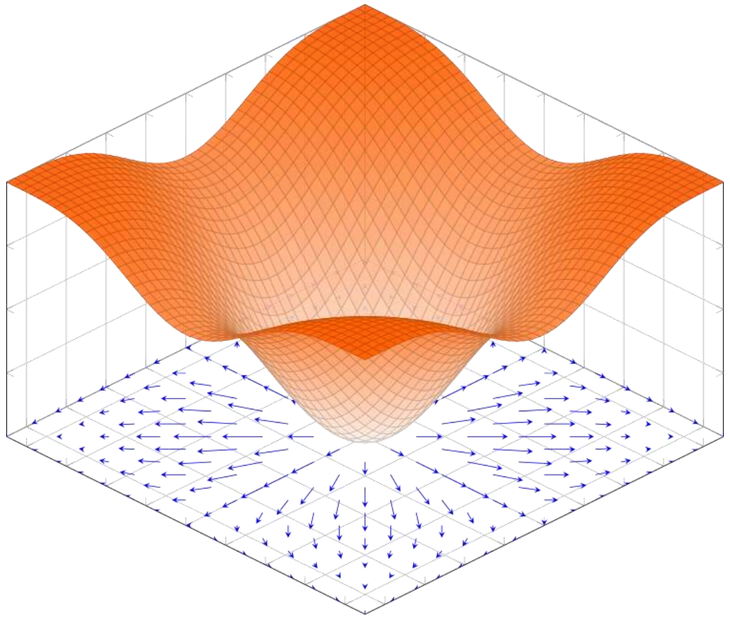
Multilayer *perceptron*

Capa de salida

Capa oculta

Capa de entrada





$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta_k)$$

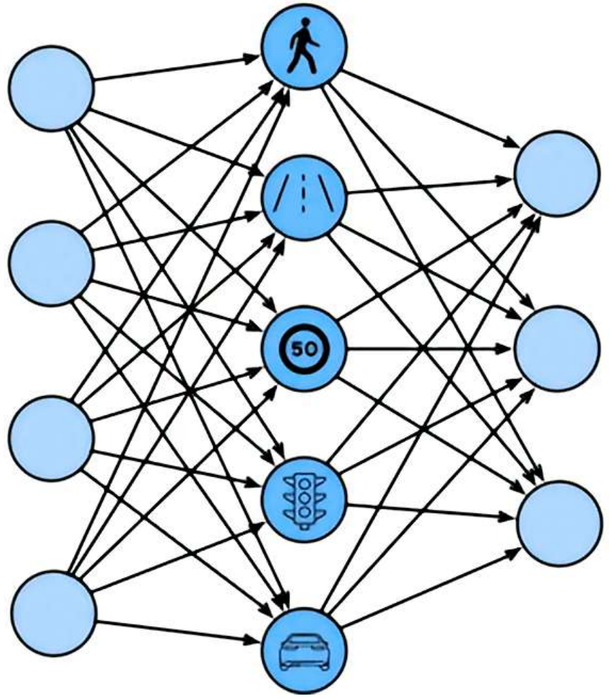
Multilayer *perceptron*

Cámara

Velocidad

Aceleración

Rotación

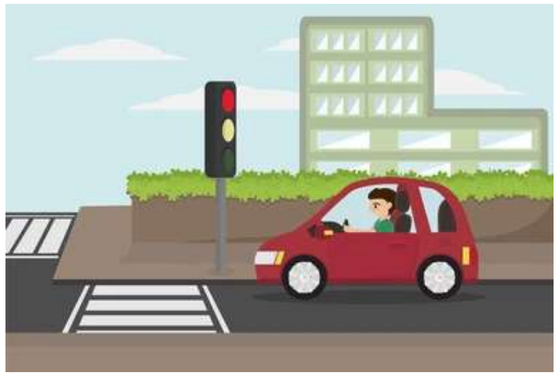


Volante

Acelerador

Freno

Multilayer *perceptron*

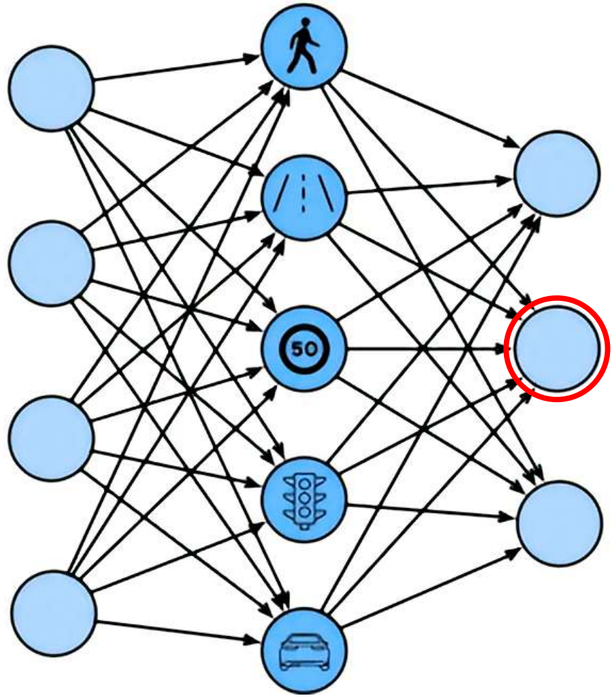


Cámara

Velocidad

Aceleración

Rotación

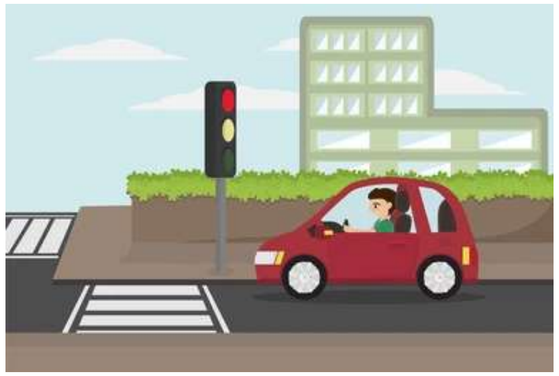


Volante

Acelerador

Freno

Backpropagation

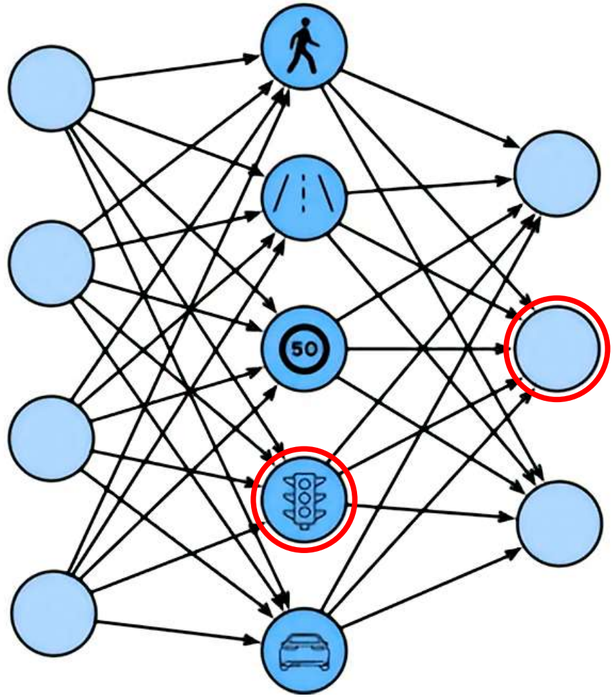


Cámara

Velocidad

Aceleración

Rotación

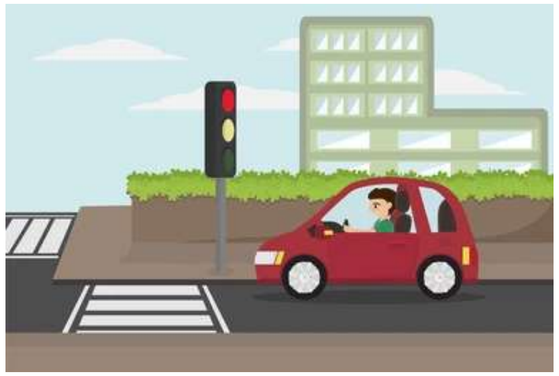


Volante

Acelerador

Freno

Backpropagation

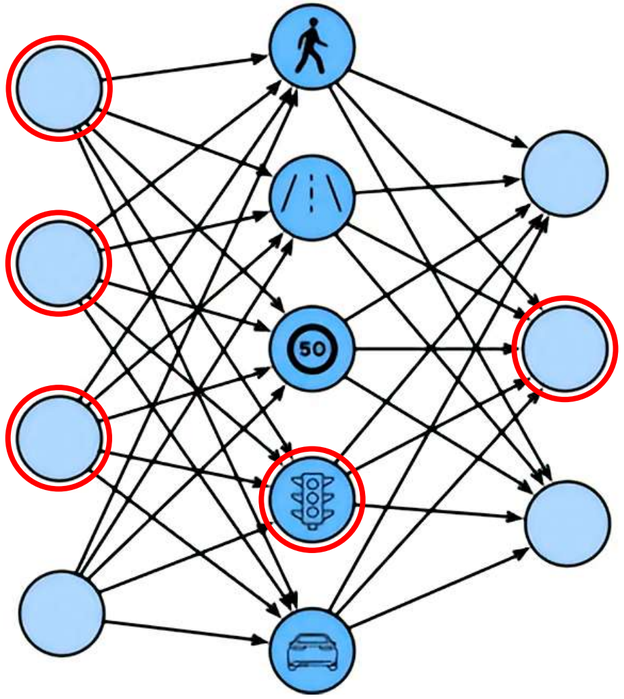


Cámara

Velocidad

Aceleración

Rotación

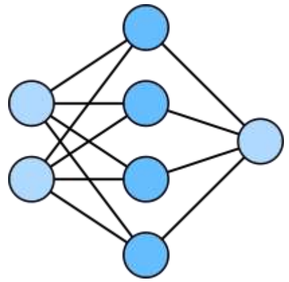


Volante

Acelerador

Freno

Backpropagation



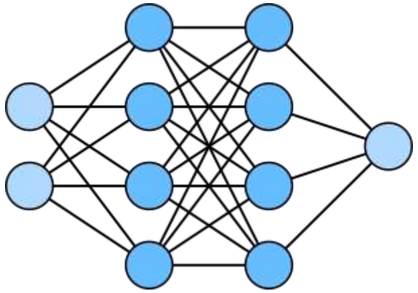
$$w \searrow$$
$$x \longrightarrow z = xw^{\intercal} \xrightarrow{f(\cdot)} y = f(z) \longrightarrow \mathcal{L} = \frac{1}{2}|y - \hat{y}|^2$$
$$\hat{y} \searrow$$

Regla de la Cadena

$$\frac{d\mathcal{L}}{dw} = \frac{d\mathcal{L}}{dy}\frac{dy}{dz}\frac{dz}{dw}$$

$$\mathcal{L}'(w) = \underbrace{(y_2 - \hat{y})}f'(z_2)x$$

Backpropagation



$$x \xrightarrow{w_1} z_1 = xw_1^\intercal \xrightarrow{f(\cdot)} y_1 = f(z_1) \xrightarrow{w_2} z_2 = y_1 w_2^\intercal \xrightarrow{f(\cdot)} y_2 = f(z_2) \xrightarrow{\hat{y}} \mathcal{L} = \frac{1}{2}|y_2 - \hat{y}|^2$$

Regla de la Cadena

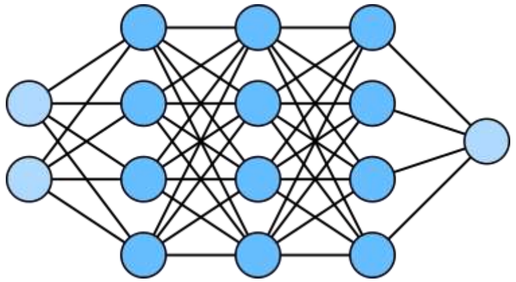
$$\frac{d\mathcal{L}}{dw_2} = \frac{d\mathcal{L}}{dy_2}\frac{dy_2}{dz_2}\frac{dz_2}{dw_2}$$

$$\frac{d\mathcal{L}}{dw_1} = \frac{d\mathcal{L}}{dy_2}\frac{dy_2}{dz_2}\frac{dz_2}{dy_1}\frac{dy_1}{dz_1}\frac{dz_1}{dw_1}$$

$$\mathcal{L}'(w_2) = \underbrace{(y_2 - \hat{y})}f'(z_2)y_1$$

$$\mathcal{L}'(w_1) = \underbrace{(y_2 - \hat{y})}\underbrace{f'(z_2)w_2 f'(z_1)x}$$

Backpropagation



$$x \xrightarrow{w_1} z_1 = x w_1^\intercal \xrightarrow{f(\cdot)} y_1 = f(z_1) \xrightarrow{w_2} z_2 = y_1 w_2^\intercal \xrightarrow{f(\cdot)} y_2 = f(z_2) \xrightarrow{w_3} z_3 = y_2 w_3^\intercal \xrightarrow{f(\cdot)} y_3 = f(z_3) \xrightarrow{\hat{y}} \mathcal{L} = \frac{1}{2}|y_3 - \hat{y}|^2$$

Regla de la Cadena

$$\frac{d\mathcal{L}}{dw_3} = \frac{d\mathcal{L}}{dy_3}\frac{dy_3}{dz_3}\frac{dz_3}{dw_3}$$

$$\frac{d\mathcal{L}}{dw_2} = \frac{d\mathcal{L}}{dy_3}\frac{dy_3}{dz_3}\frac{dz_3}{dy_2}\frac{dy_2}{dz_2}\frac{dz_2}{dw_2}$$

$$\frac{d\mathcal{L}}{dw_1} = \frac{d\mathcal{L}}{dy_3}\frac{dy_3}{dz_3}\frac{dz_3}{dy_2}\frac{dy_2}{dz_2}\frac{dz_2}{dy_1}\frac{dy_1}{dz_1}\frac{dz_1}{dw_1}$$

$$\mathcal{L}'(w_3) = \underbrace{(y_3 - \hat{y})}f'(z_3)y_2$$

$$\mathcal{L}'(w_2) = \underbrace{(y_3 - \hat{y})}\underbrace{f'(z_3)w_3 f'(z_2)}y_1$$

$$\mathcal{L}'(w_1) = \underbrace{(y_3 - \hat{y})}\underbrace{f'(z_3)w_3 f'(z_2)}\underbrace{w_2 f'(z_1)}x$$

Gradient *Descent*

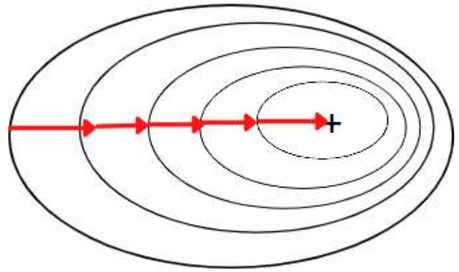
Batch Gradient Descent

Actualiza los parámetros θ en cada iteración t según la siguiente regla:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J\left(\theta^{(t)}\right)$$

donde:

- $\eta > 0$ es la tasa de aprendizaje.
- $\nabla J\left(\theta^{(t)}\right) = \frac{1}{N}\sum_{i=1}^{N} \nabla \mathcal{L}\left(f\left(x_i; \theta^{(t)}\right), y_i\right)$ es la gradiente de la función loss.



Propiedades:

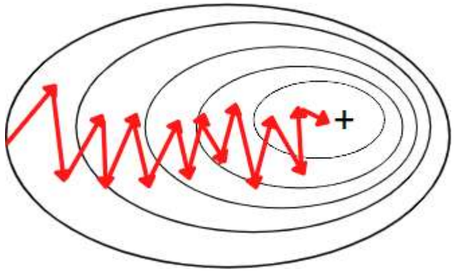
- Utiliza todo el conjunto de datos para calcular el gradiente en cada actualización.
- Computacionalmente costoso para grandes conjuntos de datos.
- Convergencia estable, pero puede ser lenta.

Gradient *Descent*

Stochastic Gradient Descent

Actualiza utilizando una única muestra de entrenamiento (x_i, y_i) en cada iteración t. La actualización se realiza con:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}\left(f\left(x_{i_t}; \theta^{(t)}\right), y_{i_t}\right)$$



donde:

- i_t es un índice seleccionado aleatoriamente entre $\{1, 2, \ldots, N\}$ en la iteración t.
- $\eta > 0$ es la tasa de aprendizaje.
- $\nabla \mathcal{L}\left(f\left(x_{i_t}; \theta^{(t)}\right), y_{i_t}\right)$ es la gradiente de la función loss para la muestra $\left(x_{i_t}, y_{i_t}\right)$.

Propiedades:

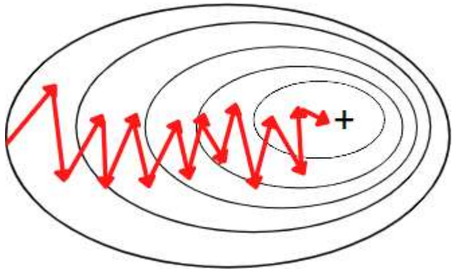
- Actualiza los parámetros con mayor frecuencia, lo que puede conducir a una convergencia más rápida en términos de iteraciones.
- Introduce ruido en el proceso de optimización, lo que puede ayudar a escapar de mínimos locales, pero también puede dificultar la convergencia precisa.
- Más eficiente en términos de memoria y computación para grandes conjuntos de datos.

Gradient *Descent*

Mini Batch Gradient Descent

Combina los métodos anteriores: actualiza los parámetros θ utilizando un subconjunto pequeño de muestras, llamado mini-batch, en cada iteración. Sea $B \subset \mathcal{D}$ un mini-batch de tamaño m. La actualización se realiza según:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J_B\left(\theta^{(t)}\right)$$



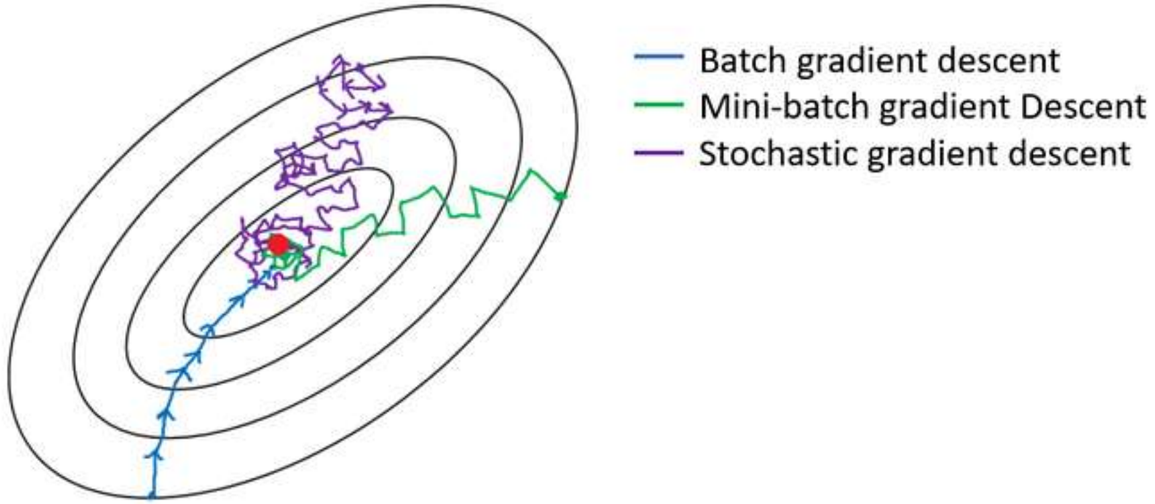
donde:

- $\nabla J_B\left(\theta^{(t)}\right) = \frac{1}{m}\sum_{i \in B} \nabla \mathcal{L}\left(f\left(x_i; \theta^{(t)}\right), y_i\right)$ es el gradiente promedio calculado sobre el mini-batch B.
- $\eta > 0$ es la tasa de aprendizaje.

Propiedades:

- Balancea la eficiencia computacional y la estabilidad de la convergencia.
- Reduce la varianza de actualizaciones, con respecto al Stochastic Gradient Descent, lo que conduce a una convergencia más estable.

Gradient *Descent*



Optimizers in *deep learning*

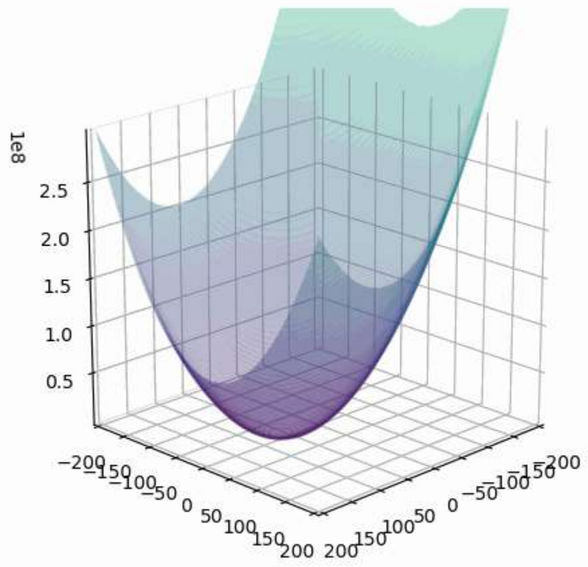
SGD (Stochastic Gradient Descent)

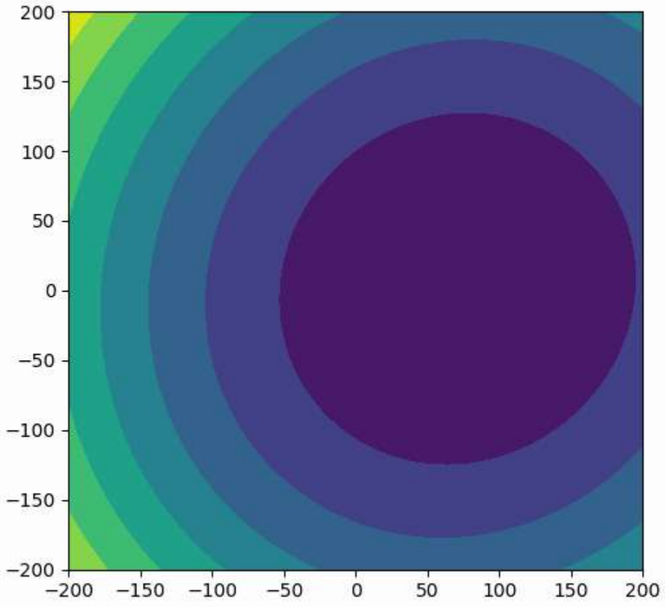
Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \eta g_t$$

η: Tasa de aprendizaje.

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.





Optimizers in *deep learning*

Momentum

Velocidad acumulada en el tiempo t.

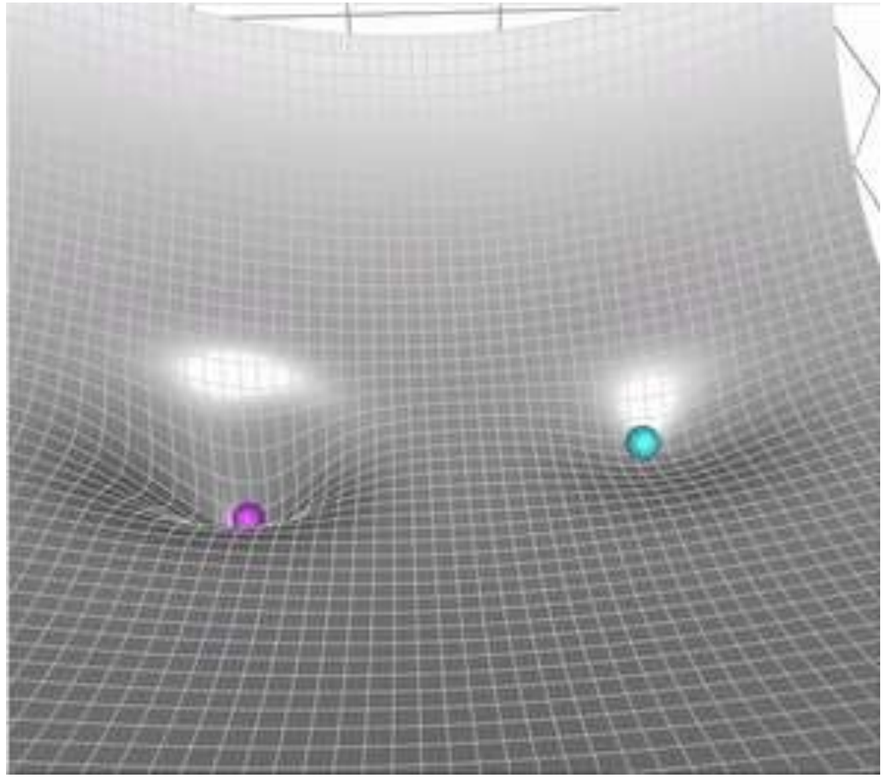
$$v_{t+1} = \mu v_t + g_t$$

Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \eta v_{t+1}$$

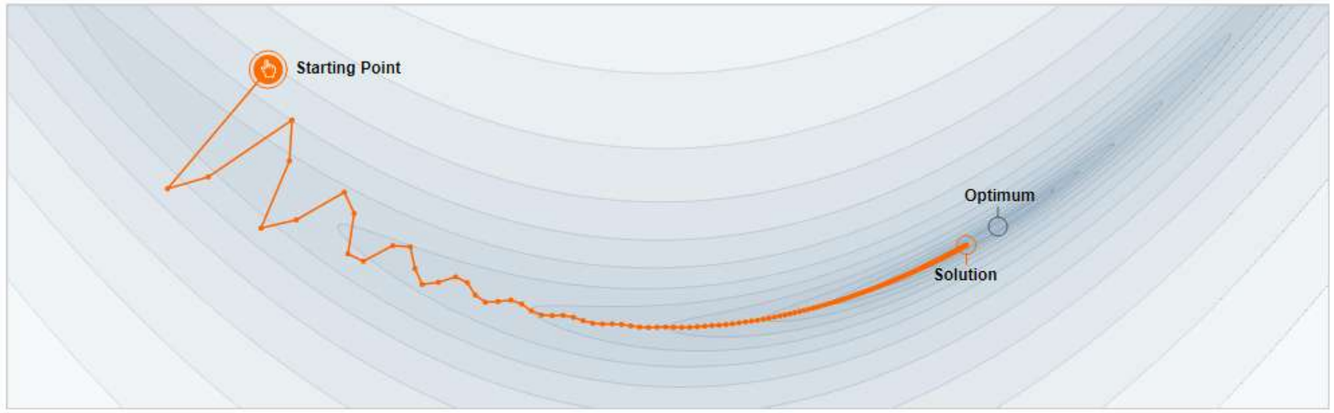
μ: Coeficiente de momentum (generalmente cercano a 0.9).

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.



Momentum vs Gradient Descent

Optimizers in *deep learning*



https://distill.pub/2017/momentum/

Optimizers in *deep learning*

Adagrad (adaptive gradient algorithm)

Acumulación de los gradientes al cuadrado.

$$G_{t+1} = G_t + g_t^2$$

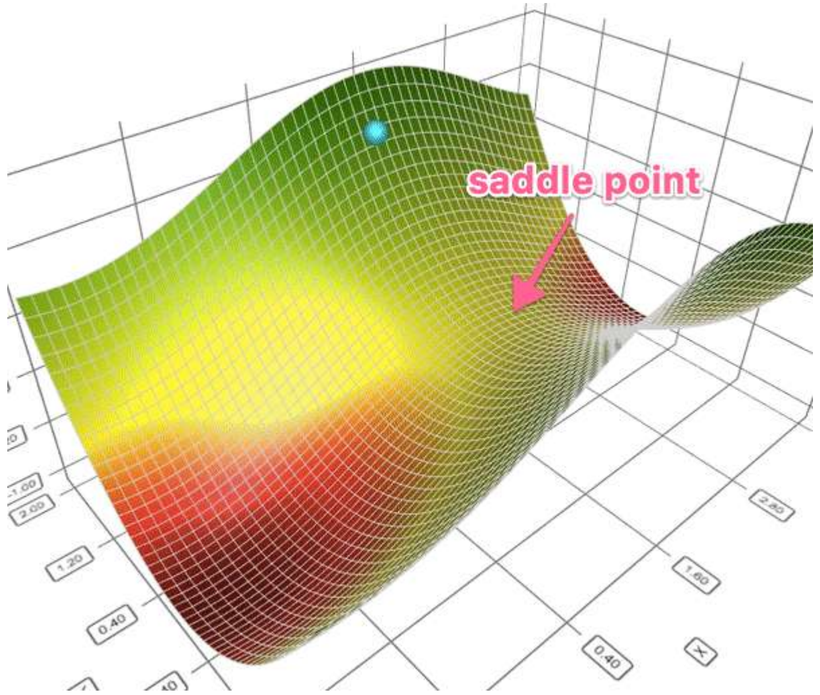
Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1} + \epsilon}} g_t$$

ϵ: Término de estabilidad numérica (típicamente muy pequeño, como 10^{-8})

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.

g_t^2: Cuadrado elemento a elemento de la gradiente.



AdaGrad **vs** Gradient Descent

Optimizers in *deep learning*

Adagrad (adaptive gradient algorithm)

Acumulación de los gradientes al cuadrado.

$$G_{t+1} = G_t + g_t^2$$

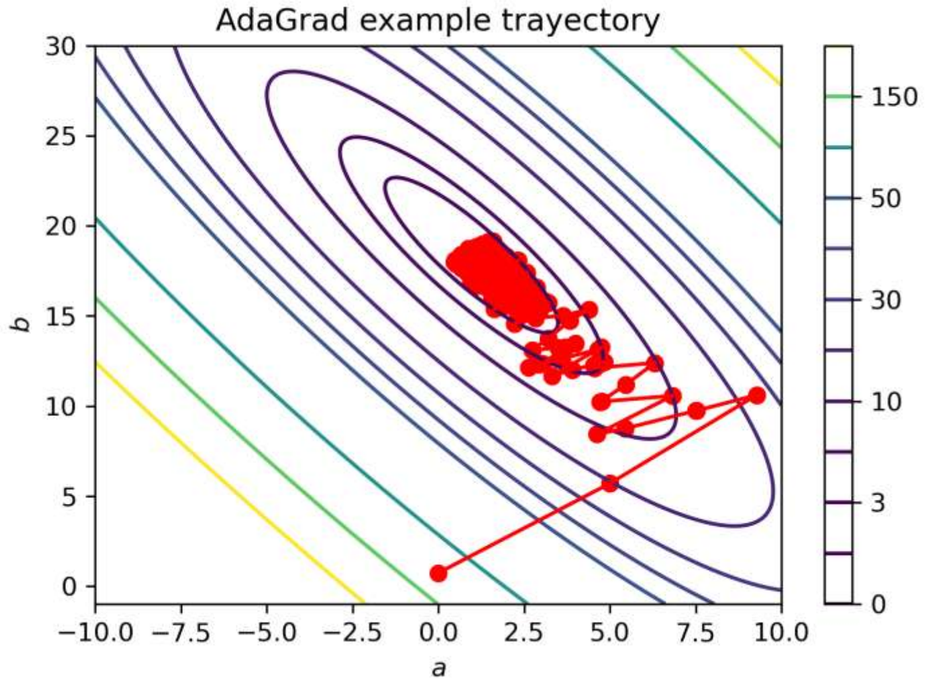
Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1} + \epsilon}} g_t$$

ϵ: Término de estabilidad numérica (típicamente muy pequeño, como 10^{-8})

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.

g_t^2: Cuadrado elemento a elemento de la gradiente.



Optimizers in *deep learning*

RMSprop (Root Mean Square Propagation)

Promedio móvil exponencial de los gradientes al cuadrado.

$$\mathbb{E}[g^2]_{t+1} = \beta_2 \mathbb{E}[g^2]_t + (1 - \beta_2) g_t^2$$

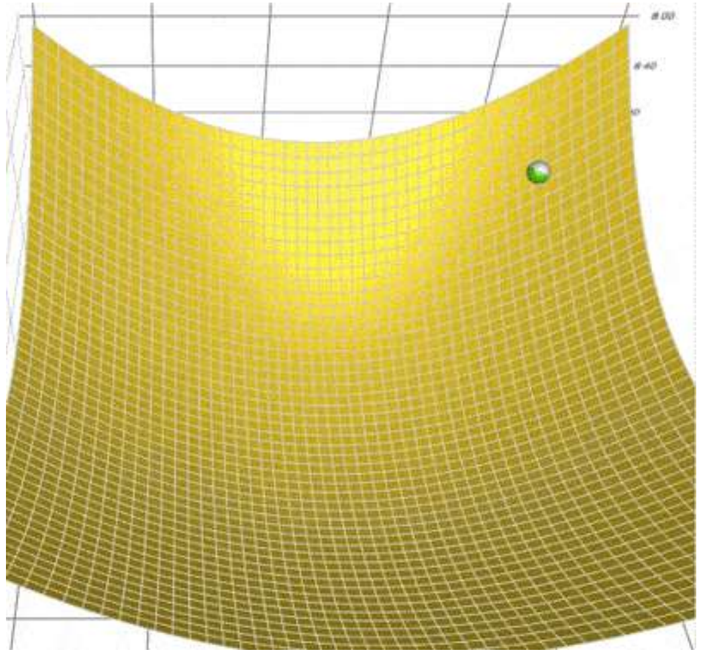
Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_{t+1} + \epsilon}} g_t$$

β_2: Coeficiente de promedio móvil (normalmente 0.9).

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.

g_t^2: Cuadrado elemento a elemento de la gradiente.



RMSProp vs AdaGrad

Optimizers in *deep learning*

Adam (Adaptive Moment Estimation)

Promedio móvil exponencial de los gradientes (primer momento):

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t$$

Promedio móvil exponencial de los gradientes al cuadrado (segundo momento):

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2$$

Correcciones de sesgo para los momentos m_t y v_t:

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 + \beta_1^{t+1}} \qquad \hat{v}_{t+1} = \frac{v_{t+1}}{1 + \beta_2^{t+1}}$$

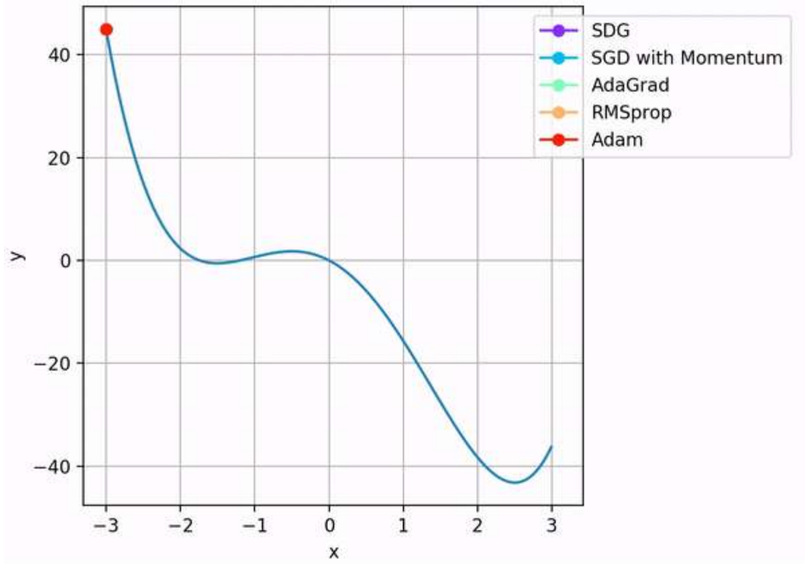
Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}} + \epsilon} \hat{m}_{t+1}$$

$\beta_2 = 0.9$, $\beta_2 = 0.999$: Coeficientes típicos para los promedios móviles.

g_t: Gradiente estocástica de la función de pérdida con respecto a θ_t.

g_t^2: Cuadrado elemento a elemento de la gradiente.



Optimizers in *deep learning*

AdamW (Adam with decoupled weight decay)

Promedio móvil exponencial de los gradientes (primer momento):

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t$$

Promedio móvil exponencial de los gradientes al cuadrado (segundo momento):

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2$$

Correcciones de sesgo para los momentos m_t y v_t:

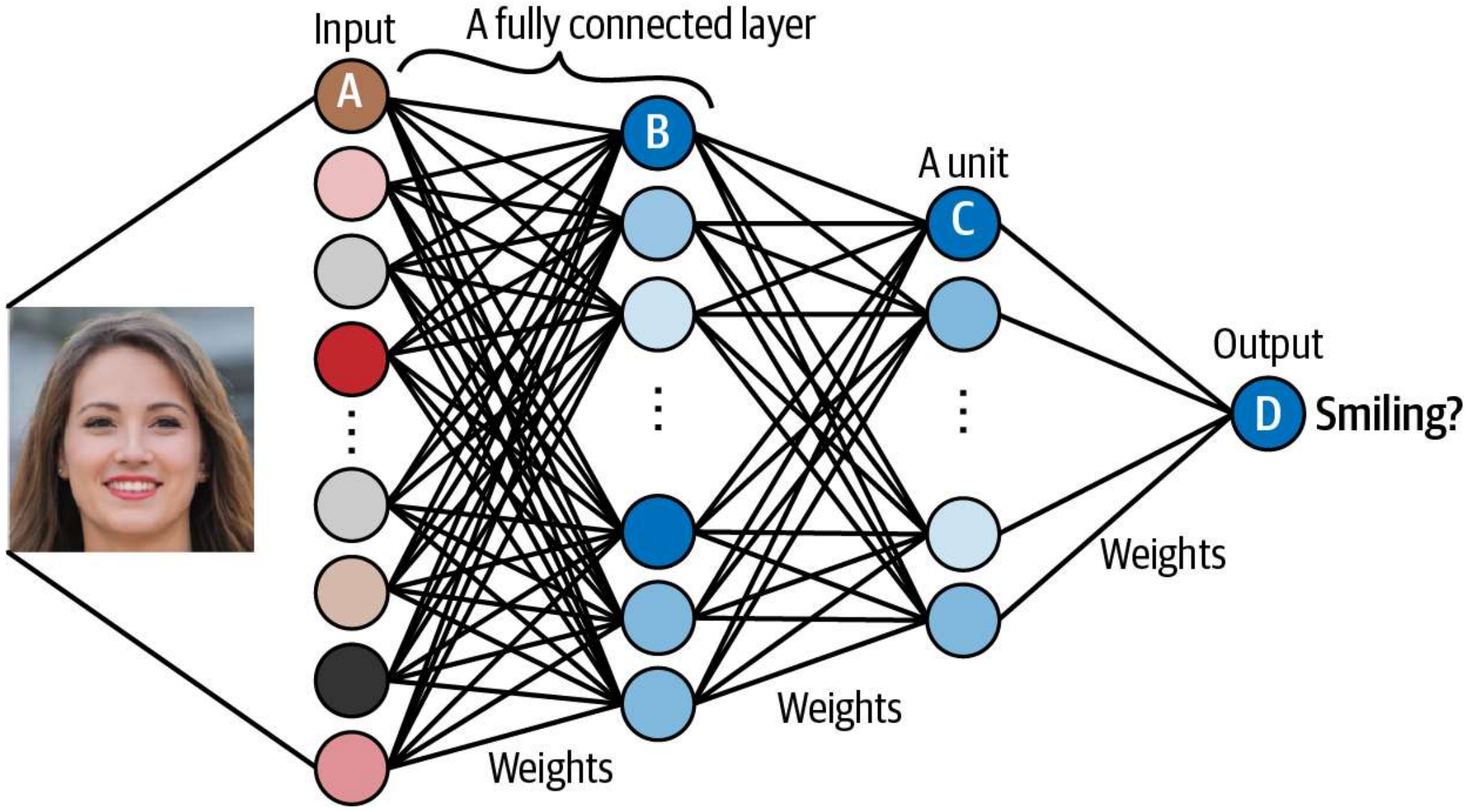
$$\hat{m}_{t+1} = \frac{m_{t+1}}{1 + \beta_1^{t+1}} \qquad \hat{v}_{t+1} = \frac{v_{t+1}}{1 + \beta_2^{t+1}}$$

Formula de actualización de pesos:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon} + \lambda \theta_t \right)$$

λ: Factor de decaimiento de pesos (típicamente 10^{-2}, 10^{-3})

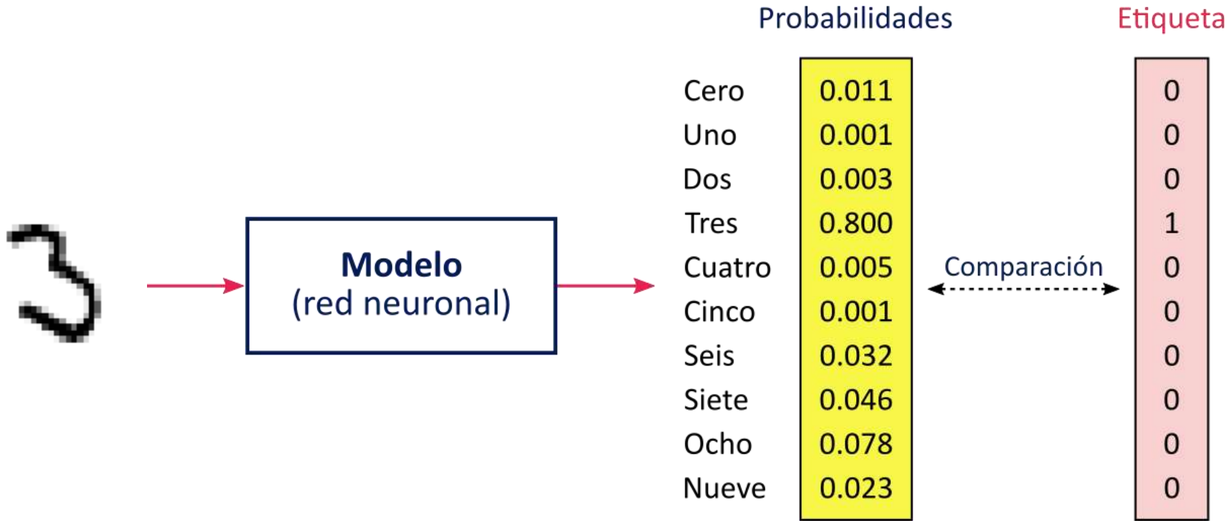
Multilayer *perceptron*



Cross-*entropy loss*

Cross-entropy

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i$$



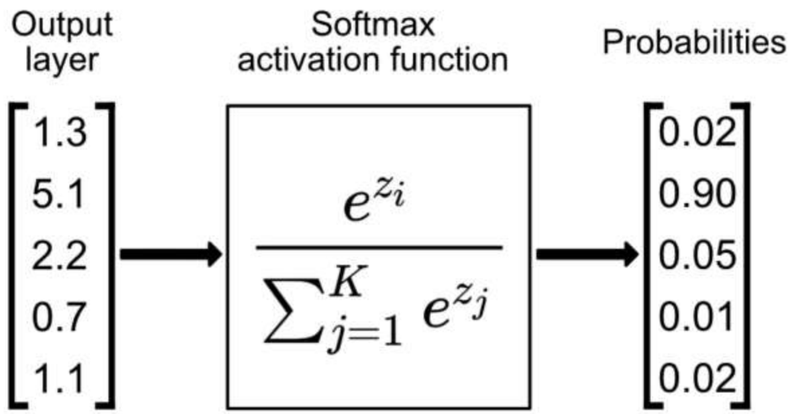
Binary cross-entropy

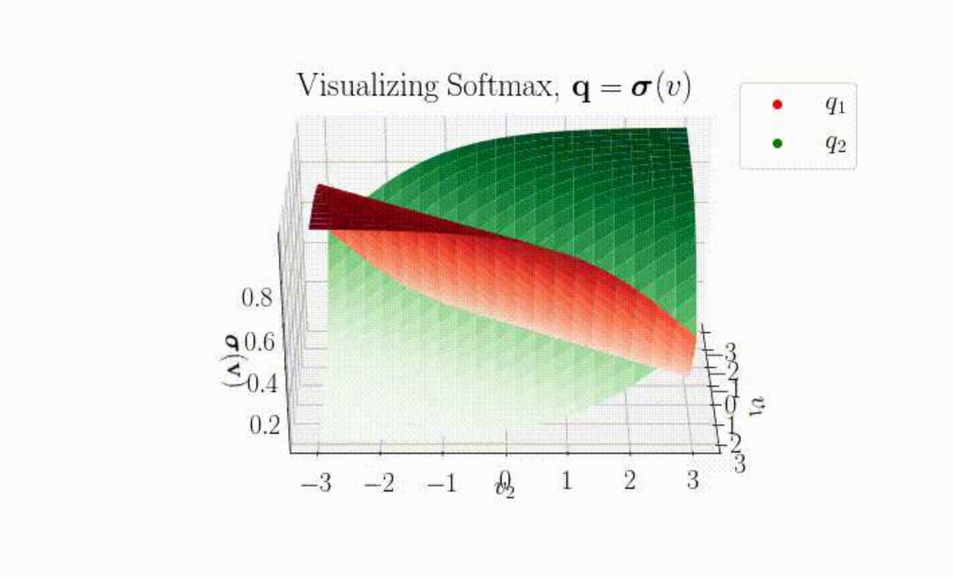
$$\mathcal{L}(y, \hat{y}) = -[y_0 \log \hat{y}_0 + y_1 \log \hat{y}_1 \ ]$$

- $y_0 = 1 - y$ y $y_1 = y$, donde y es la etiqueta verdadera que puede ser 0 o 1.
- $\hat{y}_0 = 1 - \hat{y}$ y $\hat{y}_1 = \hat{y}$, donde $\hat{y}$ es la probabilidad predicha de que la clase sea 1.

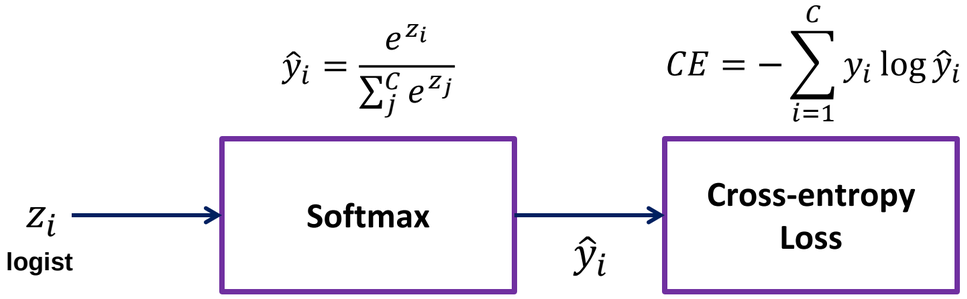
$$\mathcal{L}(y, \hat{y}) = -[(1 - y) \log(1 - \hat{y}) + y \log \hat{y} \ ]$$

Softmax


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

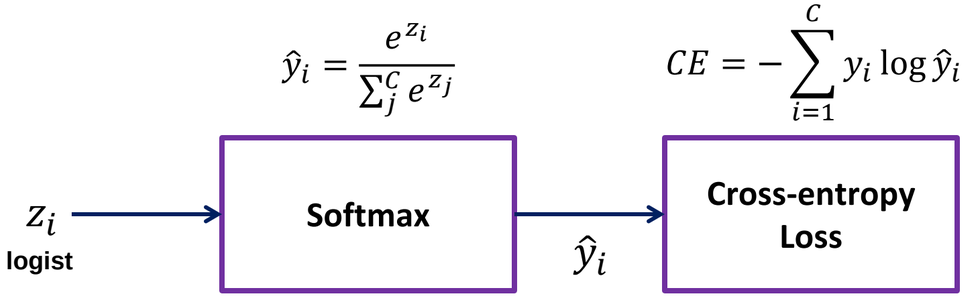

Visualizing Softmax, $\mathbf{q} = \boldsymbol{\sigma}(v)$
q_1
q_2
0.8
0.6
0.4
0.2
$\sigma(v)$
−3
−2
−1
0
1
2
3
v_2
v_1

Categorical cross-*entropy loss*



Reemplazando: $\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log \frac{e^{z_i}}{\sum_j^C e^{z_j}}$

Categorical cross-*entropy loss*



Reemplazando: $$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log \frac{e^{z_i}}{\sum_j^C e^{z_j}}$$

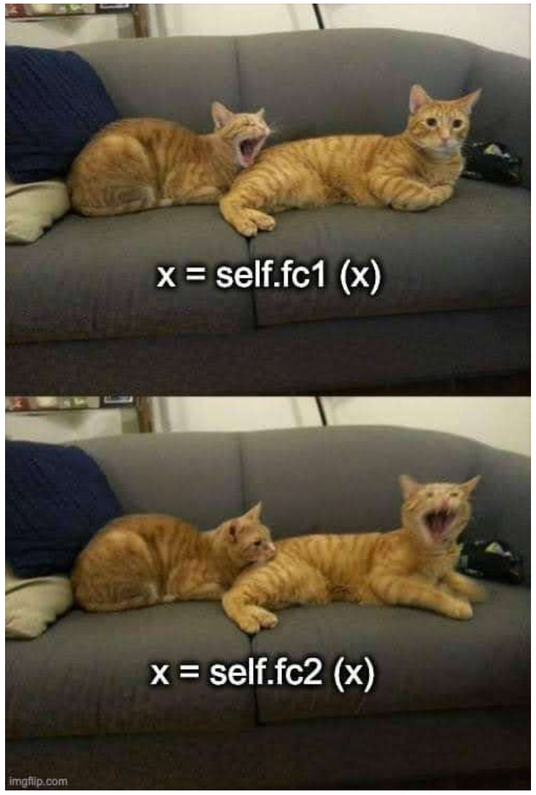
Si la etiqueta verdadera corresponde a la clase i, entonces $y_i = 1$ para esa clase y 0 para las demás. Por lo tanto, podemos simplificar la suma:

$$\mathcal{L}(y, \hat{y}) = -\log \frac{e^{z_y}}{\sum_j^C e^{z_j}} = -\left(z_y - \log \sum_j^C e^{z_j}\right) = \left(\log \sum_j^C e^{z_j}\right) - z_y$$

donde z_y es el logit correspondiente a la clase verdadera.

nn.CrossEntropyLoss

Pytorch *code!*



nn.Linear

in_features (*int*) – size of each input sample
out_features (*int*) – size of each output sample
bias (*bool*) – If set to False, the layer will not learn an additive bias. Default: True

nn. Sequential

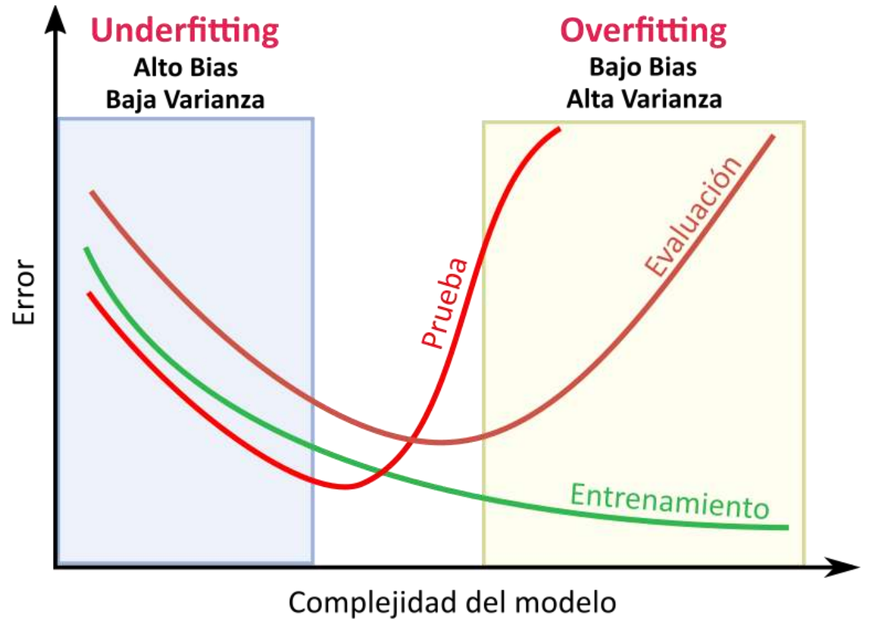
```
model = nn.Sequential(
          nn.Conv2d(1,20,5),
          nn.ReLU(),
          nn.Conv2d(20,64,5),
          nn.ReLU()
        )
```

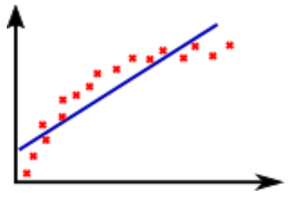
```
model = nn.Sequential(OrderedDict([
          ('conv1', nn.Conv2d(1,20,5)),
          ('relu1', nn.ReLU()),
          ('conv2', nn.Conv2d(20,64,5)),
          ('relu2', nn.ReLU())
        ]))
```

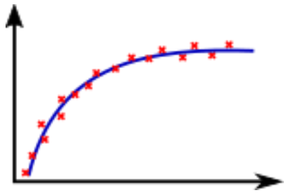
torch.optim.Adam

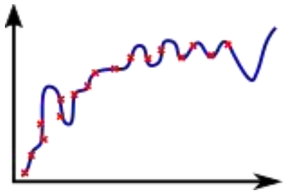
params (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
lr (*float*, *Tensor*, *optional*) – learning rate (default: 1e-3).
betas (*Tuple[float, float], optional*) – coefficients used for computing running averages of gradient and its square (default: (0.9, 0.999))
eps (*float*, *optional*) – term added to the denominator to improve numerical stability (default: 1e-8)

Bias–variance *tradeoff*

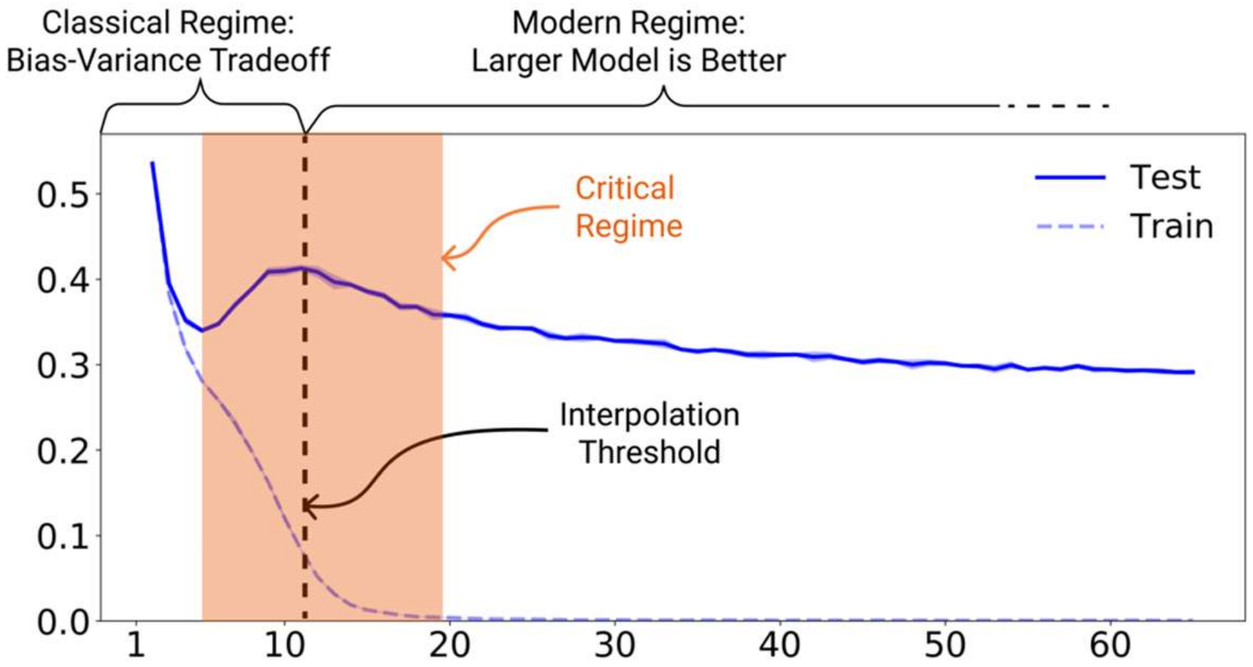








Double *descent*

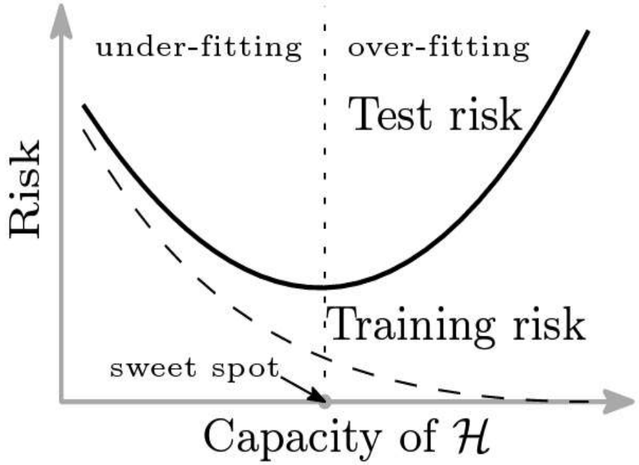


ResNet18 width parameter

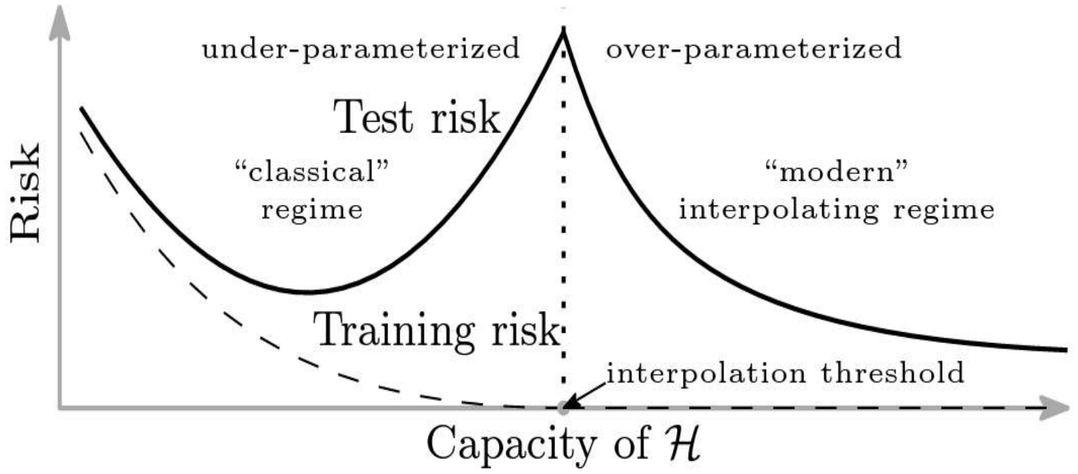
Preetum Nakkiran et al. (2021) "Deep double descent: Where bigger models and more data hurt". Journal of Statistical Mechanics: Theory and Experiment, 2021(12), 124003.

Double *descent*

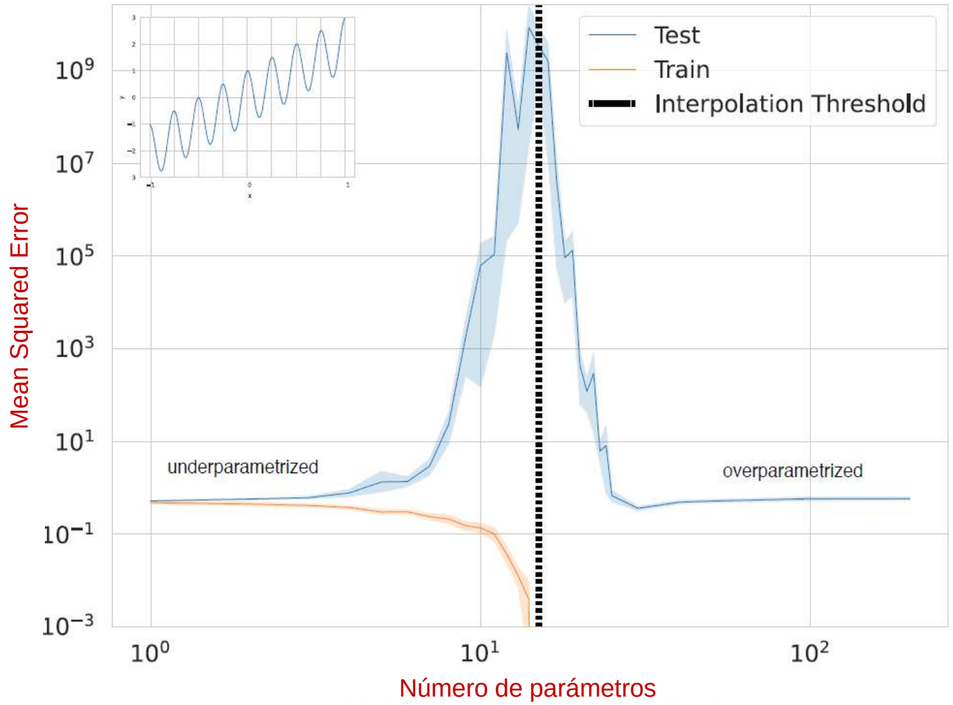
U-shaped risk curve



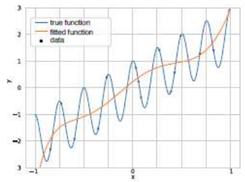
Double descent risk curve

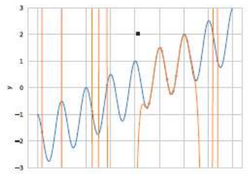


Double *descent*

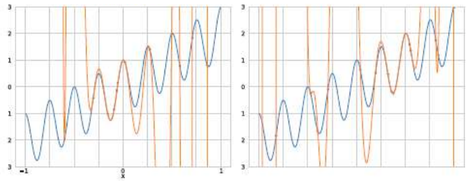


Underparameterized

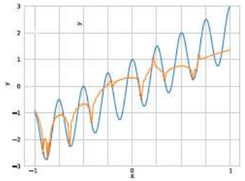




Interpolation threshold

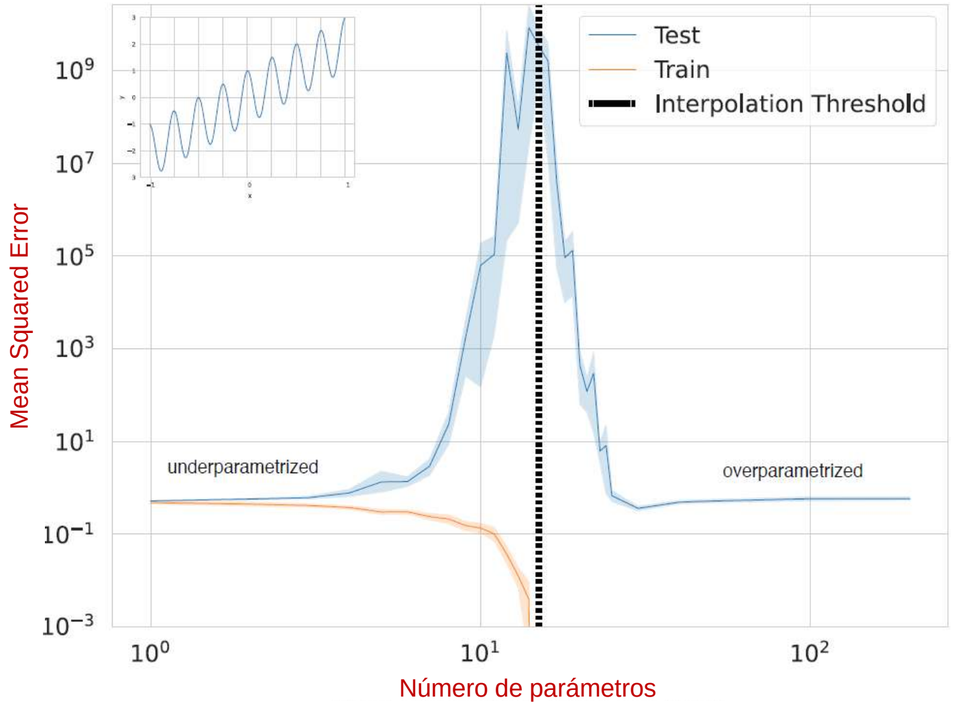


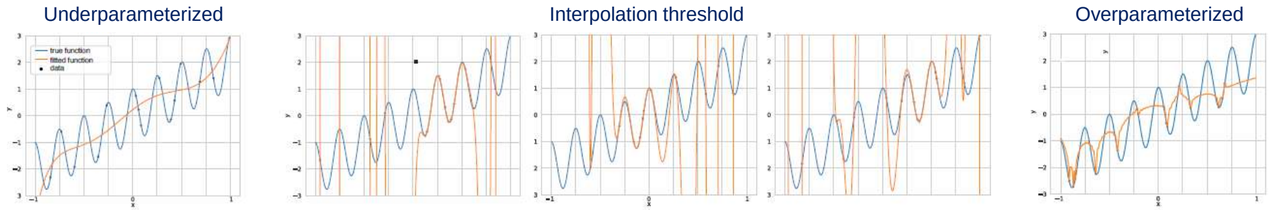
Overparameterized



Rylan Schaeffer et al. (2023) "Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle". arXiv preprint arXiv:2303.14151.

Double *descent*





Effective Model Complexity

The Effective Model Complexity (EMC) of a training procedure T, with respect to distribution $\mathcal{D}$ and parameter $\epsilon > 0$, is defined as:

$$\mathrm{EMC}_{\mathcal{D},\epsilon}(T) := \max\{n \mid \mathbb{E}_{S\sim\mathcal{D}^n}[\mathrm{Error}_S(T(S))] \leq \epsilon\}$$

where $\mathrm{Error}_S(M)$ is the mean error of model M on train samples S.

Under-paremeterized regime

If $\mathrm{Error}_{\mathcal{D},\epsilon}(T)$ is sufficiently smaller than n, any perturbation of T that increases its effective complexity will decrease the test error.

Over-parameterized regime

If $\mathrm{Error}_{\mathcal{D},\epsilon}(T)$ is sufficiently larger than n, any perturbation of T that increases its effective complexity will decrease the test error.

Critically parameterized regime

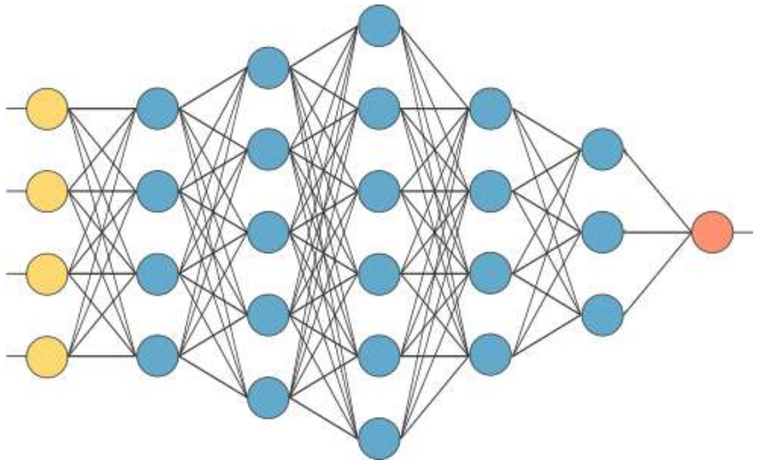
If $\mathrm{Error}_{\mathcal{D},\epsilon}(T) \approx n$ then a perturbation of T that increases its effective complexity might decrease or increase the test error.

Rylan Schaeffer et al. (2023) "Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle". arXiv preprint arXiv:2303.14151.

Vanishing/exploding *gradient*

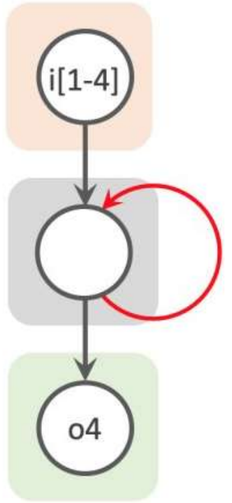
Vanishing gradient

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$$



Exploding gradient

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$$



Weight *initialization*

Si los pesos iniciales son muy grandes, los gradientes pueden crecer exponencialmente.

Una solución en controlar la variación en magnitud entre la entrada y la salida de cada capa de la red.

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization

La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$

donde su varianza es:

$$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$ donde su varianza es: $$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Dado que w_i y x_i son independientes $\text{var}(w_i x_i) = \text{var}(w_i) \cdot \text{var}(x_i)$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$ donde su varianza es: $$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Dado que w_i y x_i son independientes $\text{var}(w_i x_i) = \text{var}(w_i) \cdot \text{var}(x_i)$

Suponiendo que la entrada x_i está normalizada, es decir tiene varianza 1, tenemos: $$\text{var}(y) = \sum_{i=1}^{n} \text{var}(w_i)$$

Si todos los pesos w_i tienen la misma distribución con varianza σ^2, entonces: $$\text{var}(y) = n \cdot \sigma^2$$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$ donde su varianza es: $$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Dado que w_i y x_i son independientes $\text{var}(w_i x_i) = \text{var}(w_i) \cdot \text{var}(x_i)$

Suponiendo que la entrada x_i está normalizada, es decir tiene varianza 1, tenemos: $$\text{var}(y) = \sum_{i=1}^{n} \text{var}(w_i)$$

Si todos los pesos w_i tienen la misma distribución con varianza σ^2, entonces: $$\text{var}(y) = n \cdot \sigma^2$$

Varianza de la salida después de la activación ReLU

Si asumimos que, y sigue una distribución simétrica alrededor de cero, entonces: $$\text{var}(\text{ReLU}(y)) \approx \frac{1}{2}\text{var}(y) = \frac{1}{2} n \cdot \sigma^2$$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$ donde su varianza es: $$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Dado que w_i y x_i son independientes $\text{var}(w_i x_i) = \text{var}(w_i) \cdot \text{var}(x_i)$

Suponiendo que la entrada x_i está normalizada, es decir tiene varianza 1, tenemos: $$\text{var}(y) = \sum_{i=1}^{n} \text{var}(w_i)$$

Si todos los pesos w_i tienen la misma distribución con varianza σ^2, entonces: $$\text{var}(y) = n \cdot \sigma^2$$

Varianza de la salida después de la activación ReLU

Si asumimos que, y sigue una distribución simétrica alrededor de cero, entonces: $$\text{var}(\text{ReLU}(y)) \approx \frac{1}{2}\text{var}(y) = \frac{1}{2} n \cdot \sigma^2$$

Para evitar que la varianza de la activación disminuya a medida que avanzamos en las capas de la red, queremos que la varianza de la salida después de la activación sea la misma que la varianza de la entrada antes de la activación:

$$\frac{1}{2} n \cdot \sigma^2 = \text{var}(x) = 1 \quad \Rightarrow \quad \sigma^2 = \frac{2}{n}$$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

Weight *initialization*

Kaiming/He Initialization La salida y antes de aplicar la función de activación

$$y = \sum_{i=1}^{n} w_i x_i$$ donde su varianza es: $$\text{var}(y) = \text{var}\left(\sum_{i=1}^{n} w_i x_i\right) = \sum_{i=1}^{n} \text{var}(w_i x_i)$$

Dado que w_i y x_i son independientes $\text{var}(w_i x_i) = \text{var}(w_i) \cdot \text{var}(x_i)$

Suponiendo que la entrada x_i está normalizada, es decir tiene varianza 1, tenemos: $$\text{var}(y) = \sum_{i=1}^{n} \text{var}(w_i)$$

Si todos los pesos w_i tienen la misma distribución con varianza σ^2, entonces: $$\text{var}(y) = n \cdot \sigma^2$$

Varianza de la salida después de la activación ReLU

Si asumimos que, y sigue una distribución simétrica alrededor de cero, entonces: $$\text{var}(\text{ReLU}(y)) \approx \frac{1}{2}\text{var}(y) = \frac{1}{2} n \cdot \sigma^2$$

Para evitar que la varianza de la activación disminuya a medida que avanzamos en las capas de la red, queremos que la varianza de la salida después de la activación sea la misma que la varianza de la entrada antes de la activación:

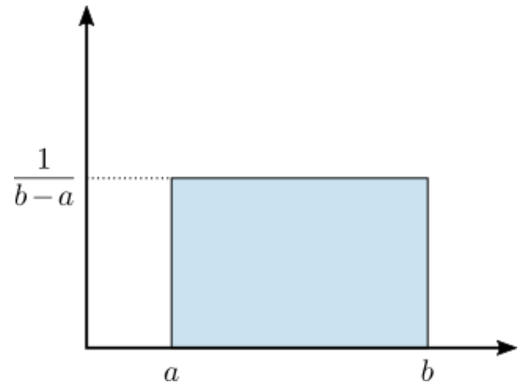
$$\frac{1}{2} n \cdot \sigma^2 = \text{var}(x) = 1 \Rightarrow \sigma^2 = \frac{2}{n}$$

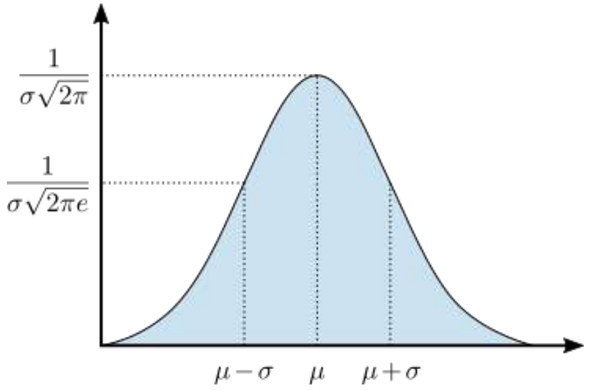
Finalmente, hacemos: $$w_i \sim \mathcal{N}\left(0, \frac{2}{n}\right)$$

Kaiming He et al. (2015) "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification". Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

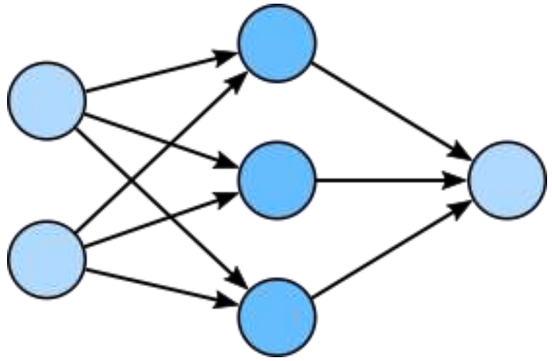
Weight *initialization*

	LeCun initialization	Xavier initialization	Kaiming initialization
Distribución Uniforme	$U\left(-\frac{3}{\sqrt{F_{in}}}, \frac{3}{\sqrt{F_{in}}}\right)$	$U\left(-\frac{6}{\sqrt{F_{in}+F_{out}}}, \frac{6}{\sqrt{F_{in}+F_{out}}}\right)$	$U\left(-\frac{6}{\sqrt{F_{in}}}, \frac{6}{\sqrt{F_{in}}}\right)$
Distribución Normal	$\mathcal{N}\left(0, \frac{1}{F_{in}}\right)$	$\mathcal{N}\left(0, \frac{2}{F_{in}+F_{out}}\right)$	$\mathcal{N}\left(0, \frac{2}{F_{in}}\right)$

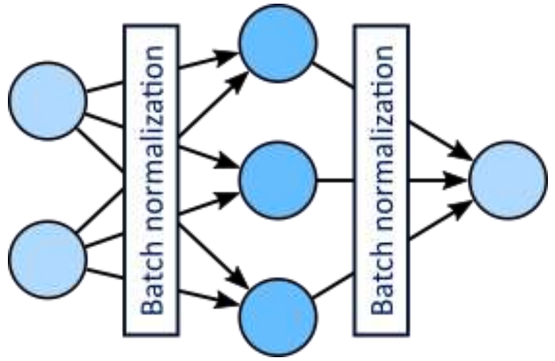




Batch *normalization*



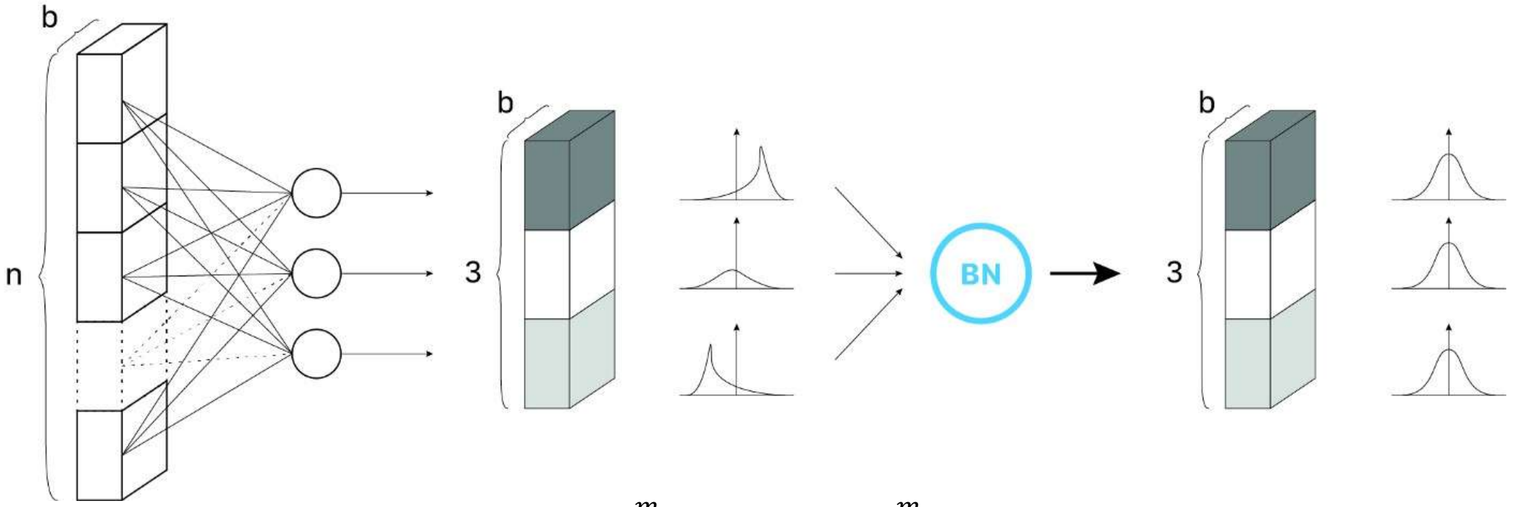
- Alta interdependencia entre distribuciones.
- Entrenamiento **lento** e **inestable**.



- Interdependencia mitigada entre distribuciones.
- Entrenamiento **rápido** y **estable**.

Sergey Ioffe and Christian Szegedy (2015) "Batch normalization: Accelerating deep network training by reducing internal covariate shift". Proceedings of The 32nd International Conference on Machine Learning (pp. 448-456).

Batch *normalization*



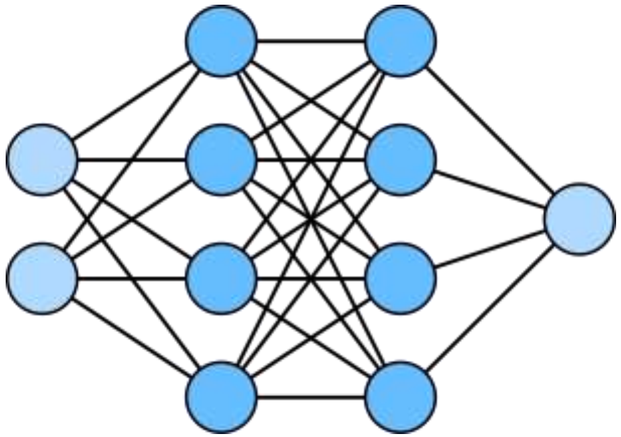
Normalización

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \mu_B)^2 \qquad \mathrm{BN}_{\gamma,\beta}(z_i) = \boldsymbol{\gamma}\hat{z}_i + \boldsymbol{\beta}$$

$$\hat{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Sergey Ioffe and Christian Szegedy (2015) "Batch normalization: Accelerating deep network training by reducing internal covariate shift". Proceedings of The 32nd International Conference on Machine Learning (pp. 448-456).

Dropout



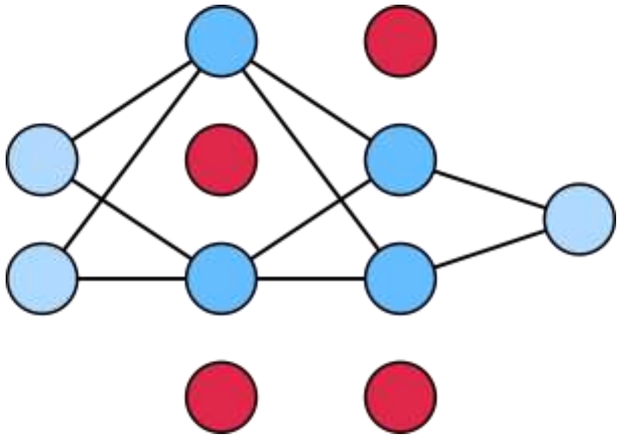
Entrenamiento:

$$z = \sum_{i=1}^{m} w_i x_i$$

Solo es aplicado a las capas *Fully connected*.

Li Wan et al. (2013) "Regularization of Neural Networks using DropConnect". International conference on machine learning. PMLR, 2013. p. 1058-1066.

Dropout



Prob: 0.5 Prob: 0.5

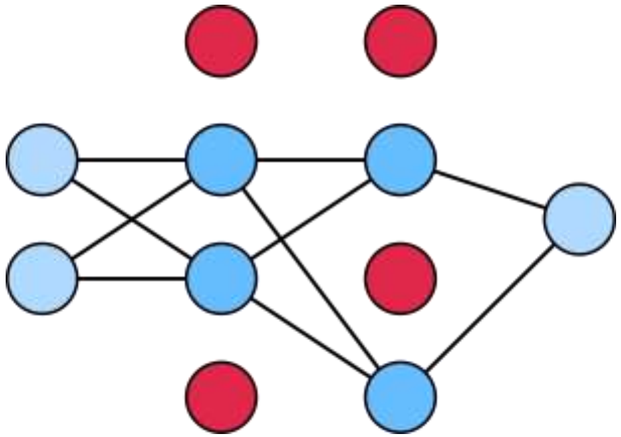
Solo es aplicado a las capas *Fully connected.*

Entrenamiento:

$$z = \frac{1}{1-p}\sum_{i=1}^{m} p_i w_i x_i \qquad \text{donde:} \quad p_i \sim \text{Bernoulli}(p)$$

Li Wan et al. (2013) "Regularization of Neural Networks using DropConnect". International conference on machine learning. PMLR, 2013. p. 1058-1066.

Dropout



Prob: 0.5 Prob: 0.5

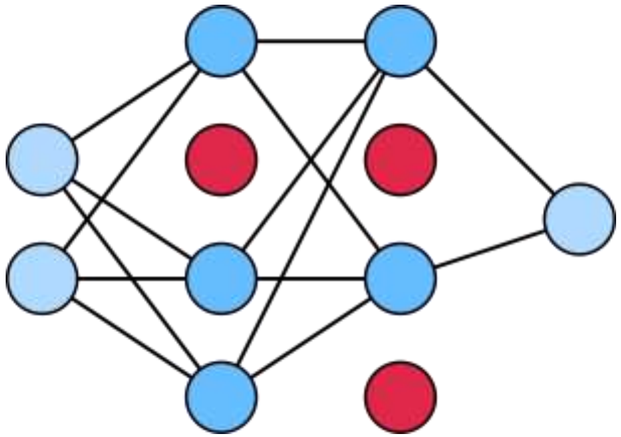
Solo es aplicado a las capas *Fully connected.*

Entrenamiento:

$$z = \frac{1}{1-p}\sum_{i=1}^{m} p_i w_i x_i \qquad \text{donde:} \qquad p_i \sim \text{Bernoulli}(p)$$

Li Wan et al. (2013) "Regularization of Neural Networks using DropConnect". International conference on machine learning. PMLR, 2013. p. 1058-1066.

Dropout



Prob: 0.5 Prob: 0.5

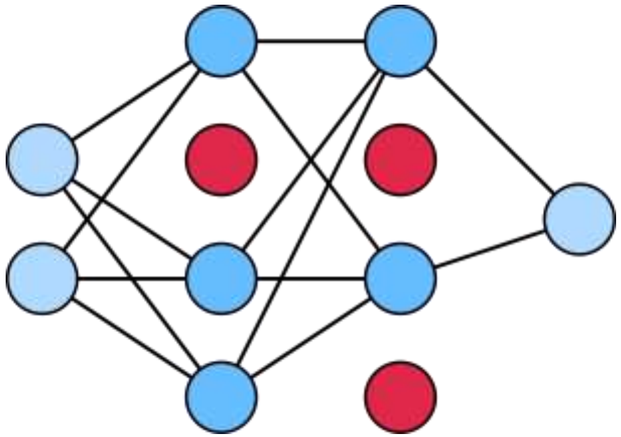
Solo es aplicado a las capas *Fully connected.*

Entrenamiento:

$$z = \frac{1}{1-p}\sum_{i=1}^{m} p_i w_i x_i \qquad \text{donde:} \qquad p_i \sim \text{Bernoulli}(p)$$

Li Wan et al. (2013) "Regularization of Neural Networks using DropConnect". International conference on machine learning. PMLR, 2013. p. 1058-1066.

Dropout



Prob: 0.5 Prob: 0.5

Solo es aplicado a las capas *Fully connected.*

Entrenamiento:

$$z = \frac{1}{1-p}\sum_{i=1}^{m} p_i w_i x_i \qquad \text{donde:} \quad p_i \sim \text{Bernoulli}(p)$$

Evaluación:

$$z = \sum_{i=1}^{m} w_i x_i$$

Li Wan et al. (2013) "Regularization of Neural Networks using DropConnect". International conference on machine learning. PMLR, 2013. p. 1058-1066.

Pytorch *code!*

nn.BatchNorm1d

num_features (*int*) – number of features or channels C of the input
eps (*float*) – a value added to the denominator for numerical stability. Default: 1e-5
momentum (*Optional[float]*) – the value used for the running_mean and running_var computation. Default: 0.1

nn.Dropout

p (*float*) – probability of an element to be zeroed. Default: 0.5
inplace (*bool*) – If set to True, will do this operation in-place. Default: False

nn.Module

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self) -> None:
        super().__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return F.relu(self.conv2(x))
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

GRACIAS

Victor Flores Benites

TRANSFORMATEC