Descenso por Gradiente (Estocástico)

Laboratorio de Datos, IC - FCEN - UBA - 1er. Cuatrimestre 2024

Regresión Lineal

Queremos explicar una variable en función de otras mediante un modelo lineal, por ejemplo:

$$Y = \beta_0 + \beta_1 X$$

Queremos explicar una variable en función de otras mediante un modelo lineal, por ejemplo:

$$Y = \beta_0 + \beta_1 X$$

Dados los datos $(x_1, y_1), \ldots, (x_n, y_n)$, nuestra función de pérdida es el Error Cuadrático Medio (MSE):

$$L(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Queremos explicar una variable en función de otras mediante un modelo lineal, por ejemplo:

$$Y = \beta_0 + \beta_1 X$$

Dados los datos $(x_1, y_1), \ldots, (x_n, y_n)$, nuestra función de pérdida es el Error Cuadrático Medio (MSE):

$$L(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Objetivo: hallar

$$\arg\min L(\beta_0, \beta_1)$$

(o sea, β_0 y β_1 que minimicen L)

Mencionamos que el problema se puede resolver hallando la solución de:

$$A^T A\beta = A^T Y$$

(de hecho, así lo hace `LinearRegression` de `scikit-learn`)

[1] $\mathcal{O}(C^2 N)$ donde C es la cantidad de *features* y N la cantidad de muestras

Mencionamos que el problema se puede resolver hallando la solución de:

$$A^T A\beta = A^T Y$$

(de hecho, así lo hace `LinearRegression` de `scikit-learn`)

Problema: resolver sistemas de ecuaciones *grandes* es computacionalmente costoso[1]

¿Qué pasa si tengo (literalmente) millones de datos?

[1] $\mathcal{O}(C^2 N)$ donde C es la cantidad de *features* y N la cantidad de muestras

Algoritmos de descenso

Hallar β es un problema de **optimización**: queremos hallar β que minimice

$$L(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_i(\beta))^2$$

Hallar β es un problema de **optimización**: queremos hallar β que minimice

$$L(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_i(\beta))^2$$

En nuestro ejemplo,

$$f_i(\beta) = \beta_0 + \beta_1 x_i$$

Los algoritmos de descenso son algoritmos iterativos: a partir de un punto inicial, en cada paso se *acercan* a un mínimo.

Se puede demostrar que si la función a optimizar cumple ciertas condiciones, en teoría convergen a un mínimo (local).

Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^{*} aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

  $d_k \leftarrow$ dirección del paso

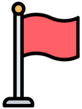
  $\eta_k \leftarrow$ longitud del paso

  $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k \, d_k$

  $k \leftarrow k + 1$

end

return $\beta^{(k)}$



Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

  $d_k \leftarrow$ dirección del paso

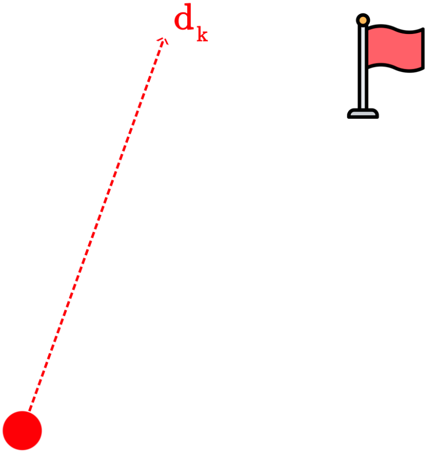
  $\eta_k \leftarrow$ longitud del paso

  $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k \, d_k$

  $k \leftarrow k + 1$

end

return $\beta^{(k)}$



Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

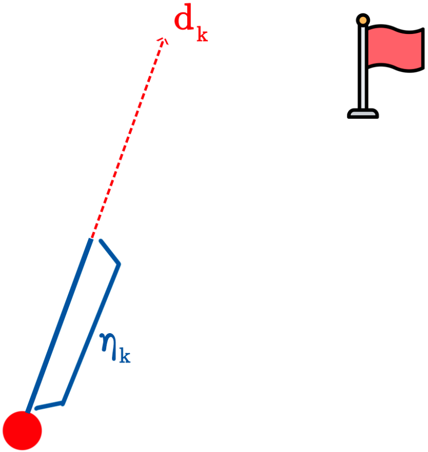
$k \leftarrow 0$

while ($\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max$) **do**

- $d_k \leftarrow$ dirección del paso
- $\eta_k \leftarrow$ longitud del paso
- $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k \, d_k$
- $k \leftarrow k + 1$

end

return $\beta^{(k)}$



Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

  $d_k \leftarrow$ dirección del paso

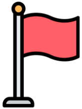
  $\eta_k \leftarrow$ longitud del paso

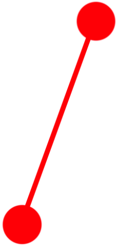
  $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k\, d_k$

  $k \leftarrow k + 1$

end

return $\beta^{(k)}$





Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

  $d_k \leftarrow$ dirección del paso

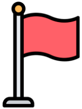
  $\eta_k \leftarrow$ longitud del paso

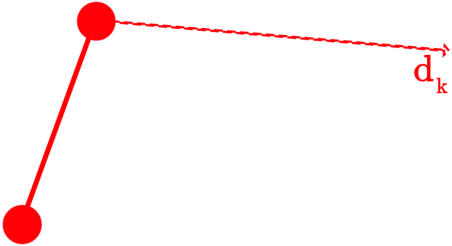
  $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k \, d_k$

  $k \leftarrow k + 1$

end

return $\beta^{(k)}$





Notación: $\beta^{(k)}$ es el valor del k-ésimo paso

Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

  $d_k \leftarrow$ dirección del paso

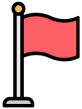
  $\eta_k \leftarrow$ longitud del paso

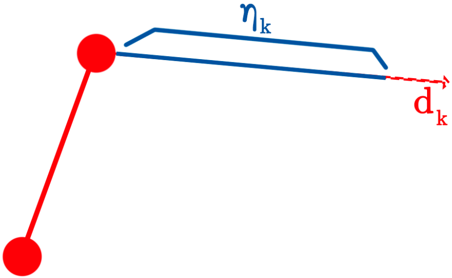
  $\beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k\, d_k$

  $k \leftarrow k + 1$

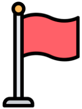
end

return $\beta^{(k)}$





Notación: $\beta^{(k)}$ es el valor del k-ésimo paso



Input: L, $\beta^{(0)}$, $\varepsilon > 0$, $it\_max > 0$

Output: β^* aproximación a un mínimo

$k \leftarrow 0$

while $(\nabla L(\beta^{(k)}) > \varepsilon$ ***and*** $k < it\_max)$ **do**

$\quad d_k \leftarrow$ dirección del paso

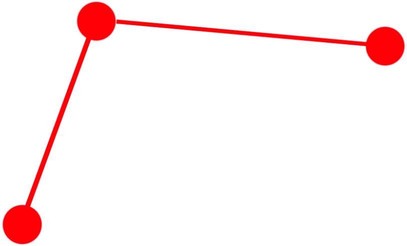
$\quad \eta_k \leftarrow$ longitud del paso

$\quad \beta^{(k+1)} \leftarrow \beta^{(k)} + \eta_k \, d_k$

$\quad k \leftarrow k + 1$

end

return $\beta^{(k)}$



En líneas generales, hay dos características que diferencian a los distintos algoritmos de descenso:

- la dirección del paso d_k
- la longitud del paso η_k

En líneas generales, hay dos características que diferencian a los distintos algoritmos de descenso:

- la dirección del paso d_k
- la longitud del paso η_k

La dirección del paso d_k debe ser una **dirección de descenso**, es decir, es suficiente que cumpla:

$$< d_k, \nabla L(\beta_k) > \leq 0$$

Descenso por Gradiente (GD)

$$d_k = -\nabla L(\beta^{(k)})$$

$$d_k = -\nabla L(\beta^{(k)})$$

Teorema: sea $f : \mathbb{R}^m \to \mathbb{R}$ tal que $f \in C^1$ y sea $x \in \mathbb{R}^m$, entonces $\nabla f(x)$ es la dirección de máximo crecimiento de f.

$$d_k = -\nabla L(\beta^{(k)})$$

Teorema: sea $f : \mathbb{R}^m \to \mathbb{R}$ tal que $f \in C^1$ y sea $x \in \mathbb{R}^m$, entonces $\nabla f(x)$ es la dirección de máximo crecimiento de f.
Corolario: $-\nabla f(x)$ es la dirección de máximo decrecimiento de f.

¿Y η_k?

¿Y η_k?

Y… es todo un tema (literalmente)

¿Y η_k?

Y… es todo un tema (literalmente)

Hay varias formas de determinar el η_k:

- dejarlo fijo: $\eta_k = \eta_0 \ \forall k$

¿Y η_k?

Y... es todo un tema (literalmente)

Hay varias formas de determinar el η_k:

- dejarlo fijo: $\eta_k = \eta_0 \; \forall k$
- hacerlo variar a lo largo de las iteraciones mediante alguna función. Por ejemplo:

$$\eta_k = \eta_0 e^{-d \cdot k} \quad d \text{ es el decaimiento (otro hiperparámetro)}$$

¿Y η_k?

Y... es todo un tema (literalmente)

Hay varias formas de determinar el η_k:

- dejarlo fijo: $\eta_k = \eta_0 \ \forall k$
- hacerlo variar a lo largo de las iteraciones mediante alguna función. Por ejemplo:

$$\eta_k = \eta_0 e^{-d \cdot k} \quad d \text{ es el decaimiento (otro hiperparámetro)}$$

- búsqueda lineal (Sección Áurea, Regla de Armijo, Regla de Wolfe, etc.)

Implementación

TensorFlow

Keras

Jerga tradicional

Jerga de Machine Learning

Jerga tradicional

- Iteración

Jerga de Machine Learning

- *Epoch* (época)

Jerga tradicional

- Iteración
- Longitud del paso

Jerga de Machine Learning

- *Epoch* (época)
- *Learning rate*

Jerga tradicional

Jerga de Machine Learning

- Iteración
- Longitud del paso
- *Intercept*

- *Epoch* (época)
- *Learning rate*
- *Bias*

Además, cambiamos la notación de β:

$$\beta = (\underbrace{\beta_0}_{\substack{b \\ \text{bias}}},\ \underbrace{\beta_1, \beta_2, \ldots, \beta_m}_{\substack{w=(w_0,w_1,\ldots,w_{m-1}) \\ \textit{weights}\text{ (pesos)}}})$$

Además, cambiamos la notación de β:

$$\beta = (\underbrace{\beta_0}_{\substack{b \\ \text{bias}}}, \underbrace{\beta_1, \beta_2, \ldots, \beta_m}_{\substack{w=(w_0, w_1, \ldots, w_{m-1}) \\ \textit{weights} \text{ (pesos)}}})$$

Ejemplos:

$$Y = \beta_0 + \beta_1 X \quad \rightarrow \quad Y = b + wX$$

Además, cambiamos la notación de β:

$$\beta = (\underbrace{\beta_0}_{\substack{b \\ \text{bias}}},\ \underbrace{\beta_1, \beta_2, \ldots, \beta_m}_{\substack{w=(w_0,w_1,\ldots,w_{m-1}) \\ \textit{weights}\text{ (pesos)}}})$$

Ejemplos:

$$\begin{aligned} Y = \beta_0 + \beta_1 X \quad &\rightarrow \quad Y = b + wX \\ Y = \beta_0 + \beta_1 X + \beta_2 X^2 \quad &\rightarrow \quad Y = b + w_0 X + w_1 X^2 \end{aligned}$$

Además, cambiamos la notación de β:

$$\beta = (\underbrace{\color{red}{\beta_0}}_{\substack{\color{red}{b}\\ \color{red}{\text{bias}}}}, \underbrace{\color{blue}{\beta_1, \beta_2, \ldots, \beta_m}}_{\substack{\color{blue}{w=(w_0, w_1, \ldots, w_{m-1})}\\ \color{blue}{\textit{weights}\text{ (pesos)}}}})$$

Ejemplos:

$$\begin{aligned} Y &= \beta_0 + \beta_1 X &\rightarrow\quad & Y = b + wX \\ Y &= \beta_0 + \beta_1 X + \beta_2 X^2 &\rightarrow\quad & Y = b + w_0 X + w_1 X^2 \\ Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 &\rightarrow\quad & Y = b + w_0 X_1 + w_1 X_2 + w_2 X_1 X_2 \end{aligned}$$

Es muy importante escalar los datos para garantizar el correcto funcionamiento del algoritmo.

Regresión no lineal

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$

OK

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK
- $Y = w e^X$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK
- $Y = we^X$ OK

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK
- $Y = w e^X$ OK
- $Y = w_0 e^{w_1 X}$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0X + w_1X^2$ OK
- $Y = b + w_0X_1 + w_1X_1^{w_2}$ NO
- $Y = b + w_0X_1 + w_1X_1X_2$ OK
- $Y = we^X$ OK
- $Y = w_0e^{w_1X}$ NO

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK
- $Y = w e^X$ OK
- $Y = w_0 e^{w_1 X}$ NO
- $Y = \dfrac{1}{1 + e^{-(b+wX)}}$

La resolución matricial para regresión sólo puede aplicarse cuando la función del modelo es lineal sobre b y w. ¿Cuáles de las siguientes funciones son lineales en b y en w?

- $Y = b + w_0 X + w_1 X^2$ OK
- $Y = b + w_0 X_1 + w_1 X_1^{w_2}$ NO
- $Y = b + w_0 X_1 + w_1 X_1 X_2$ OK
- $Y = we^X$ OK
- $Y = w_0 e^{w_1 X}$ NO
- $Y = \dfrac{1}{1 + e^{-(b+wX)}}$ NO

Descenso por gradiente no tiene esta limitación. Como:

$$L(b,w)=\frac{1}{n}\sum_{i=1}^{n}\underbrace{(y_i-f_i(b,w))^2}_{g_i(b,w)}=\frac{1}{n}\sum_{i=1}^{n}g_i(b,w)$$

vale que:

$$\nabla L(b,w)=\frac{1}{n}\sum_{i=1}^{n}\nabla g_i(b,w)$$

Entonces solamente necesitamos que las f_i sean C^1

Aplicación: Regresión Logística

Tenemos que clasificar pingüinos en dos especies (Gentoo o Chinstrap) a partir de su peso.

Consideramos el modelo:

$$Y = \frac{1}{1 + e^{-(b+wX)}}$$

Tenemos que clasificar pingüinos en dos especies (Gentoo o Chinstrap) a partir de su peso.

Consideramos el modelo:

$$Y = \frac{1}{1 + e^{-(b+wX)}}$$

La función

$$f_i(b, w) = \frac{1}{1 + e^{-(b+wx_i)}}$$

nos dará la *probabilidad* de que un pingüino que pesa x_i gramos sea de la especie Gentoo.

Función de error *Binary Cross-Entropy Loss* o *log loss* para Regresión Logística:

$$L(b,w) = -\frac{1}{n}\sum_{i=1}^{n} \Big(y_i \log(f_i(b,w)) + (1-y_i)\log(1-f_i(b,w)) \Big)$$

Función de error *Binary Cross-Entropy Loss* o *log loss* para Regresión Logística:

$$L(b,w) = -\frac{1}{n}\sum_{i=1}^{n}\Big(y_i \log(f_i(b,w)) + (1-y_i)\log(1-f_i(b,w))\Big)$$

Obs: Regresión Logística suele funcionar mejor cuando los datos están centrados además de estar escalados.

Resumiendo

	Método Matricial	Descenso por Gradiente
Dataset Pequeño	✓	✗
Dataset Grande	✗	✓
Solución	exacta	aproximada
Funciones	$f_i(w, b)$ lineal	$f_i(w, b) \in C^1$

Descenso por Gradiente

Lo bueno:

- (relativamente) sencillo de implementar
- más eficiente que la resolución matricial para datasets grandes
- no nos limita a modelos lineales

Lo malo:

- menos eficiente que la resolución matricial para datasets pequeños
- el resultado depende del punto inicial
- presencia hiperparámetros
- convergencia lenta cerca de un mínimo local

Descenso por Gradiente

Lo bueno:

- (relativamente) sencillo de implementar
- más eficiente que la resolución matricial para datasets grandes
- no nos limita a modelos lineales

Lo malo:

- menos eficiente que la resolución matricial para datasets pequeños
- el resultado depende del punto inicial
- presencia hiperparámetros
- convergencia lenta cerca de un mínimo local

Agregar aleatoriedad de manera inteligente suele ayudar a que los algoritmos de descenso converjan más rápido.

Descenso por Gradiente Estocástico (SGD)

$$L(b,w) = \frac{1}{n}\sum_{i=1}^{n} g_i(b,w)$$

Idea del GD:

Para cada época k:

1. da un paso en dirección $-\nabla L = -\sum_{i=1}^{n} \nabla g_i$ de longitud η_k

$$L(b, w) = \frac{1}{n} \sum_{i=1}^{n} g_i(b, w)$$

Idea del GD:

Para cada época k:

1. da un paso en dirección $-\nabla L = -\sum_{i=1}^{n} \nabla g_i$ de longitud η_k

Idea del SGD:

Para cada época k:

1. para cada dato de entrenamiento x_i:
   1.1 da un paso en dirección $-\nabla g_i$ de longitud η_k
2. mezclá el conjunto de entrenamiento

Idea del SGD con mini-batch:

Para cada época k:

1. separá los datos de entrenamiento en conjuntos (*batches*)
2. para cada *batch* B:
   2.1 da un paso en dirección $-\frac{1}{|B|}\sum_{i:x_i\in B}\nabla g_i$ de longitud η_k
3. mezclá el conjunto de entrenamiento

Extras

- **Learning rate schedule:** se define una función para que el learning rate varíe a lo largo de las iteraciones. Algunas comunes son:
  1. $\eta_k = \eta_0 e^{-d \cdot k}$ $\quad d$ es el decaimiento (otro hiperparámetro)
  2. $\eta_k = \dfrac{\eta_{k-1}}{1 + d \cdot k}$
  3. $\eta_k = \eta_0 \cdot d^{\left\lfloor \frac{1+k}{r} \right\rfloor}$ (cada r épocas, el learning rate es multiplicado por d)

- **Learning rate schedule:** se define una función para que el learning rate varíe a lo largo de las iteraciones. Algunas comunes son:
    1. $\eta_k = \eta_0 e^{-d \cdot k}$ $\quad d$ es el decaimiento (otro hiperparámetro)
    2. $\eta_k = \dfrac{\eta_{k-1}}{1 + d \cdot k}$
    3. $\eta_k = \eta_0 \cdot d^{\left\lfloor \frac{1+k}{r} \right\rfloor}$ (cada r épocas, el learning rate es multiplicado por d)
- **Early stopping:** si pasadas p épocas la función de pérdida no mejora, el entrenamiento se detiene.


HIPERPARAMETROS
LIGHTYEAR
HIPERPARAMETROS POR TODOS LADOS
imgflip.com