

Introducción a la Inteligencia Artificial

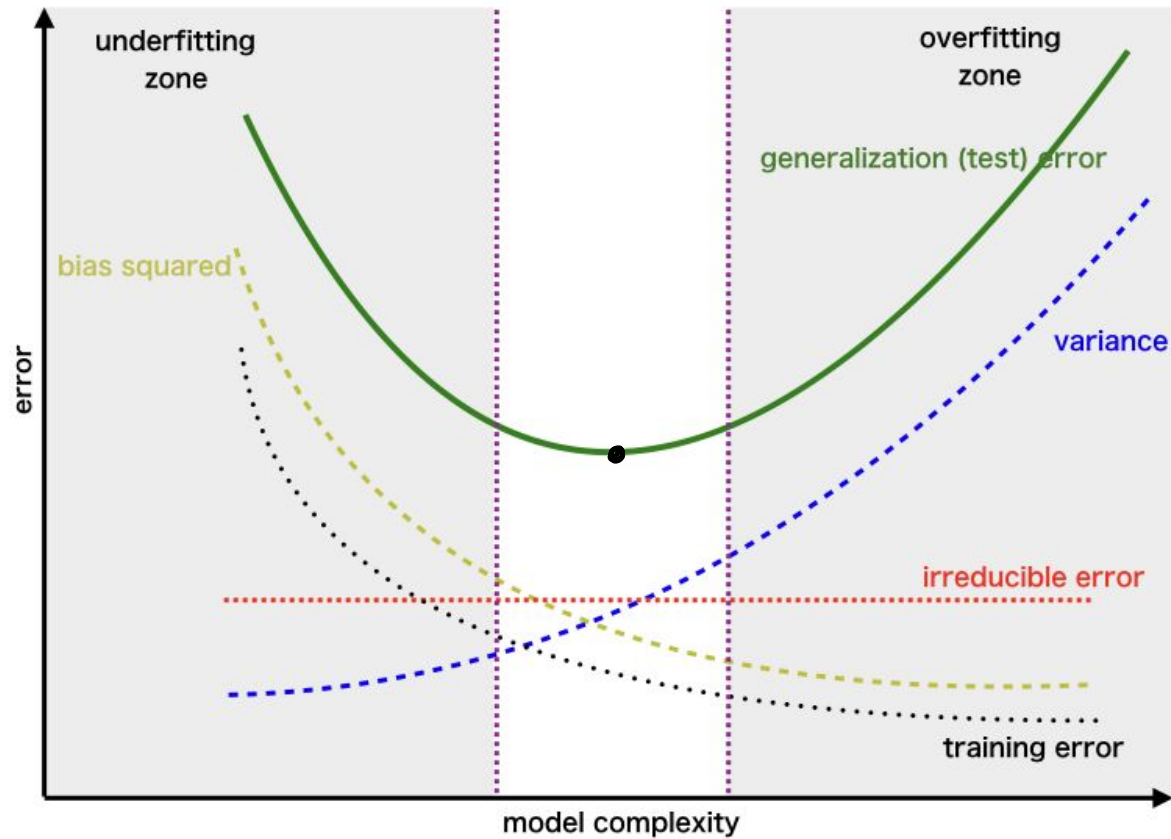
Clase 4



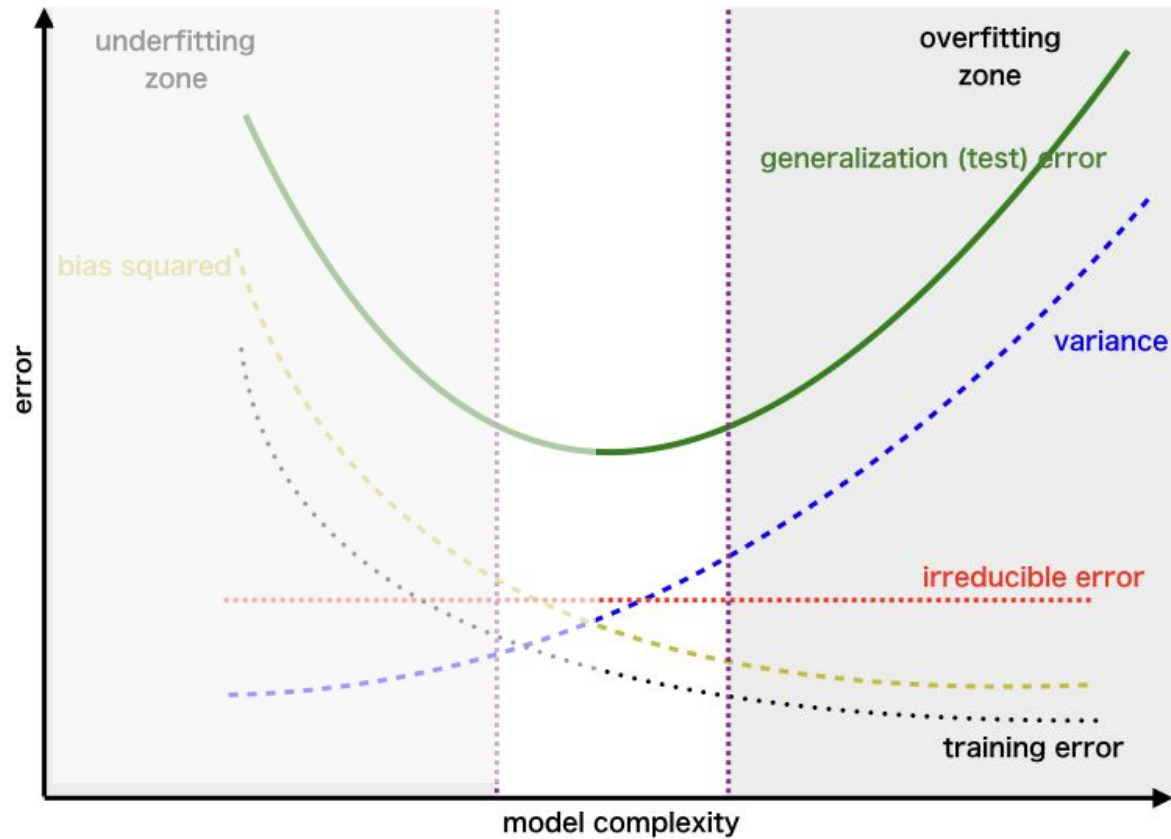
Clase 4

D. Riesgo Empírico

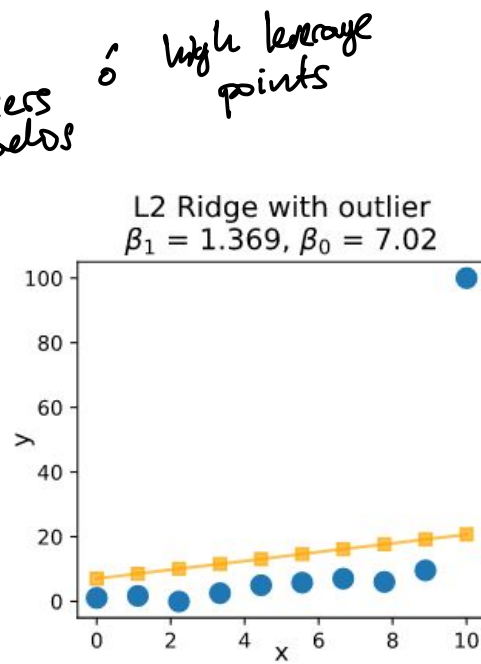
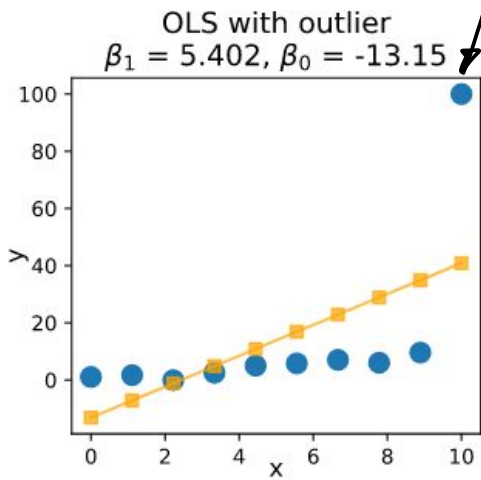
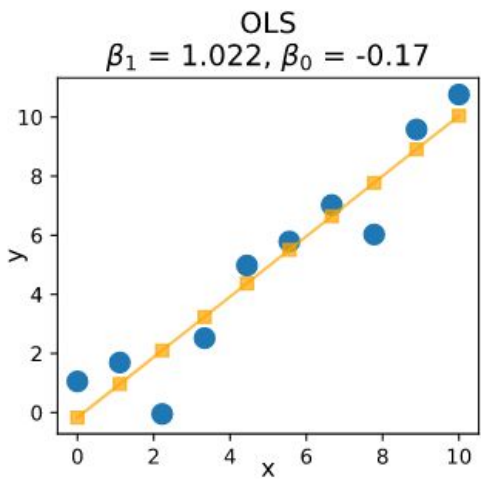
1. Regularización
 - a. Caso general
 - b. Ridge
 - c. Lasso
2. Gradient descent
 - a. GD
 - b. GD Estocástico
 - c. GD Mini-Batch
3. Entrenamiento de modelos
 - a. Selección de modelos
 - b. Cross-Validation



Regularización

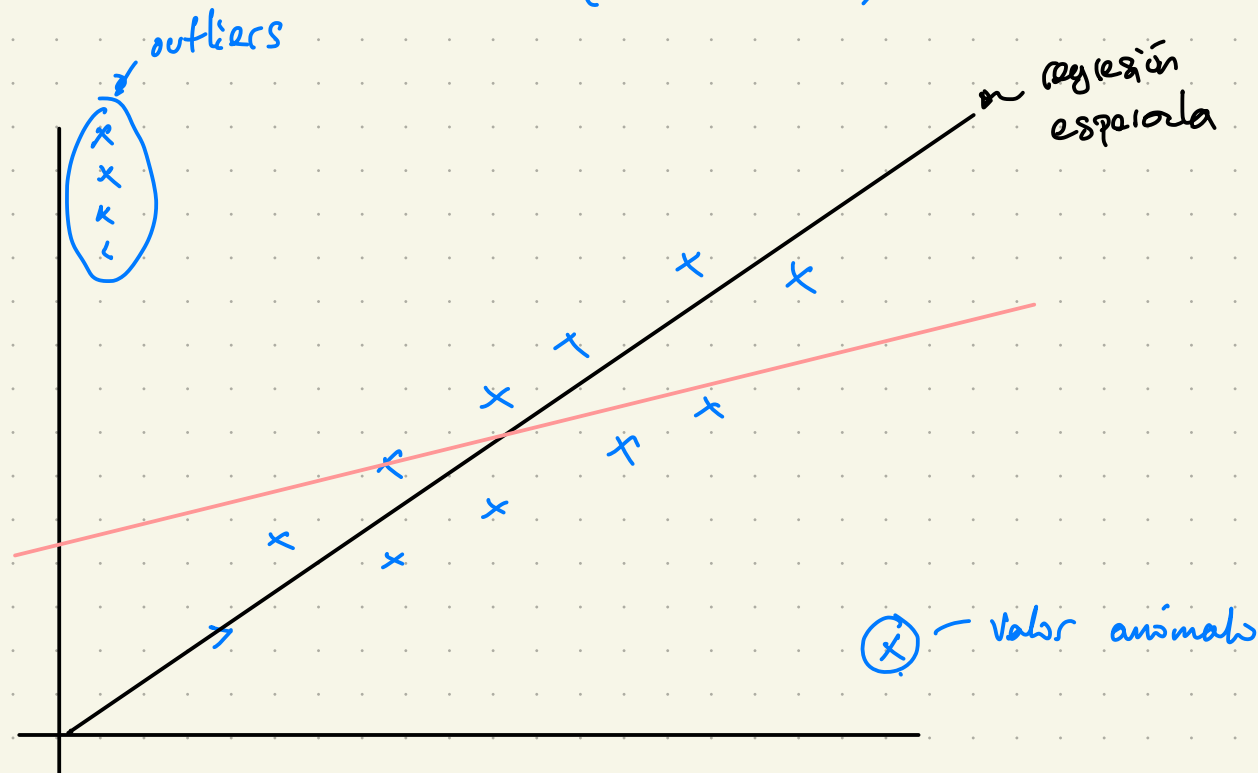


Regularización - Motivación



Riesgo Empírico

$$R(f) = \mathbb{E} \left(\underbrace{L(f, \hat{f})}_{\substack{\text{función de} \\ \text{pérdida} \\ \text{(loss function)}}} \right) \quad \leadsto \quad L(f, \hat{f}) = (f - \hat{f})^2 \quad \text{fn. de pérdida } L_2$$



¿Cómo podemos mejorar mi reg. lineal?

1. Cambiar L

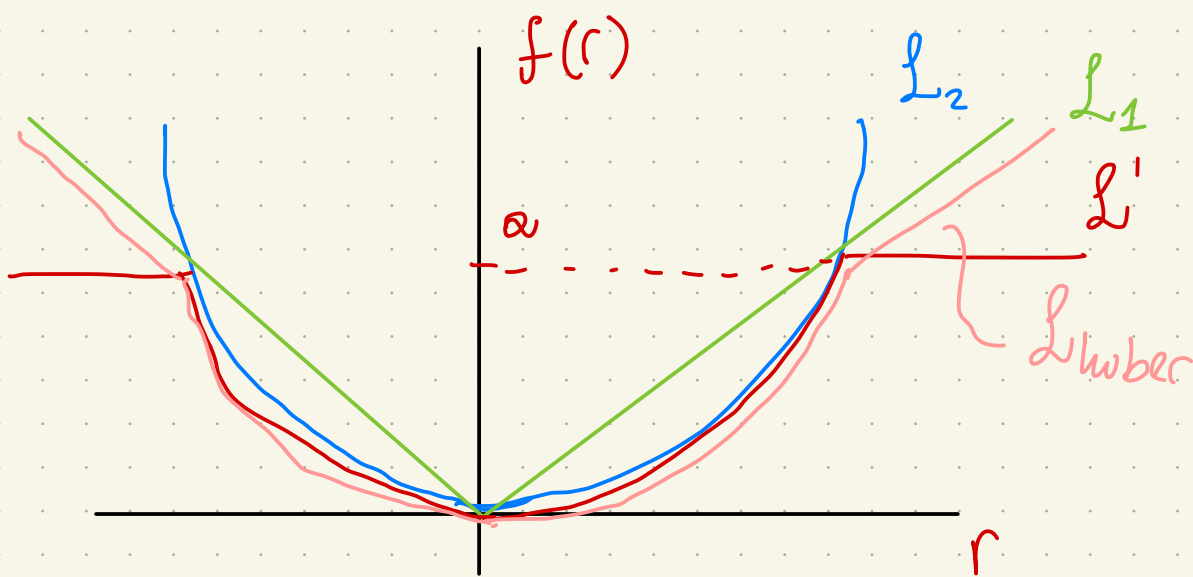
- $L_1 = |f - \hat{f}|$

- $L_2 = (f - \hat{f})^2$

- $L_{II} = \mathbb{I}_{|f - \hat{f}| \leq c}$

- L_{Huber} } Estimadores robustos de la pérdida (multiparámetros)
- L_{Tukey} }

$$L' = \begin{cases} (f - \hat{f})^2 & \text{si } |f - \hat{f}| \leq c \\ a & \text{o.w.} \end{cases}$$



$$L_{\text{huber}} = \begin{cases} \frac{1}{2} r^2 & |r| \leq \delta \\ \delta (|r| - \frac{1}{2}\delta) & \text{o.w.} \end{cases}$$

$$L_{\text{Tukey}} = \begin{cases} \frac{c^2}{\sigma} \left(1 - \left[1 - \left(\frac{|r|}{c} \right)^2 \right]^3 \right) & \text{para } |r| \leq c \\ \frac{c^2}{\sigma} & \text{o.w.} \end{cases}$$

'huber regressor'

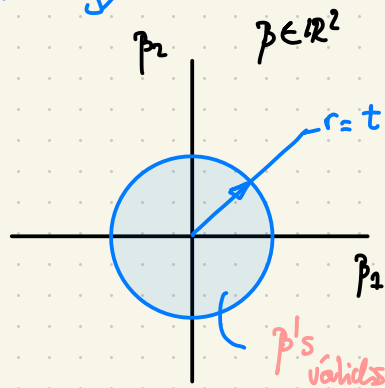
nosotros siempre buscamos minimizar R de la forma más sencilla, a veces cambiar L no es barato o peor no es útil entonces probamos **regularizar**

2. Regularización.

Con reg. buscamos min R al mismo tiempo que restringimos (limitamos) el comp. de los parámetros (**parameter shrinking**).

partimos de

$$\begin{cases} \hat{y} = \beta_0 + \sum_i \beta_i x_i \\ \hat{\beta} = \arg \min_{\beta} \left(\sum_i y_i - \sum_j \beta_j x_{ij} \right)^2 \\ \frac{1}{2} \sum_j \beta_j^2 \leq t \quad (\| \beta \|^2 \leq t) \end{cases}$$



$$q=2 \rightarrow \hat{\beta} = \arg \min_{\beta} \left(\sum_i y_i - \sum_j \beta_j x_{ij} \right)^2 - \lambda \sum_j \beta_j^2$$

Parámetro de complejidad

↑ Término de regularización
(**Weight decay**)

Espacio de parámetros

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2$$

Observado - Predicción

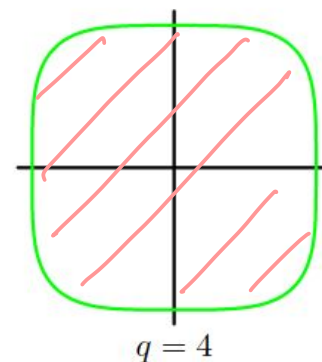
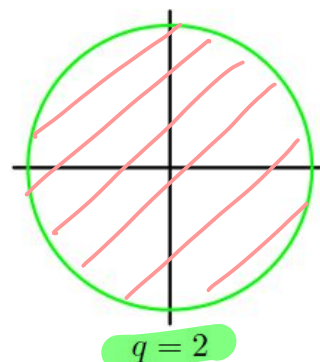
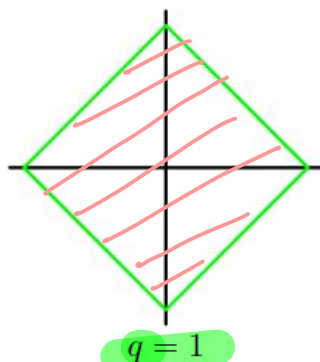
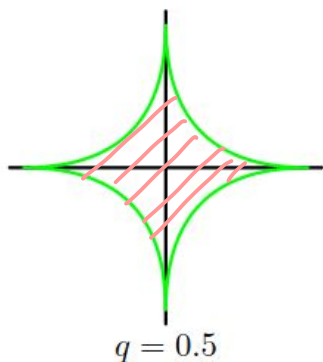


\mathbf{w} está “libre”

$$\frac{1}{2} \sum_{n=1}^N \{t_n - \mathbf{w}^T \phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2} \sum_{j=1}^M |w_j|^q$$

Término de
regularización
"weight decay"

→ w afecta la pérdida



▣ valores validos
de $\|w\|_q$

Lasso

Ridge

$$w = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$$

Maximum A Posteriori como regularización

Distribución “a priori” de los parámetros \longrightarrow Observar data \longrightarrow Actualizar distribución (Posterior)

$$p(w) \sim D(\theta)$$

$$(\mathcal{X}, \mathcal{Y})$$

$$p(w|\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}|\mathcal{X}, w)p(w)}{p(\mathcal{Y}|\mathcal{X})}$$

$$\phi(w)$$

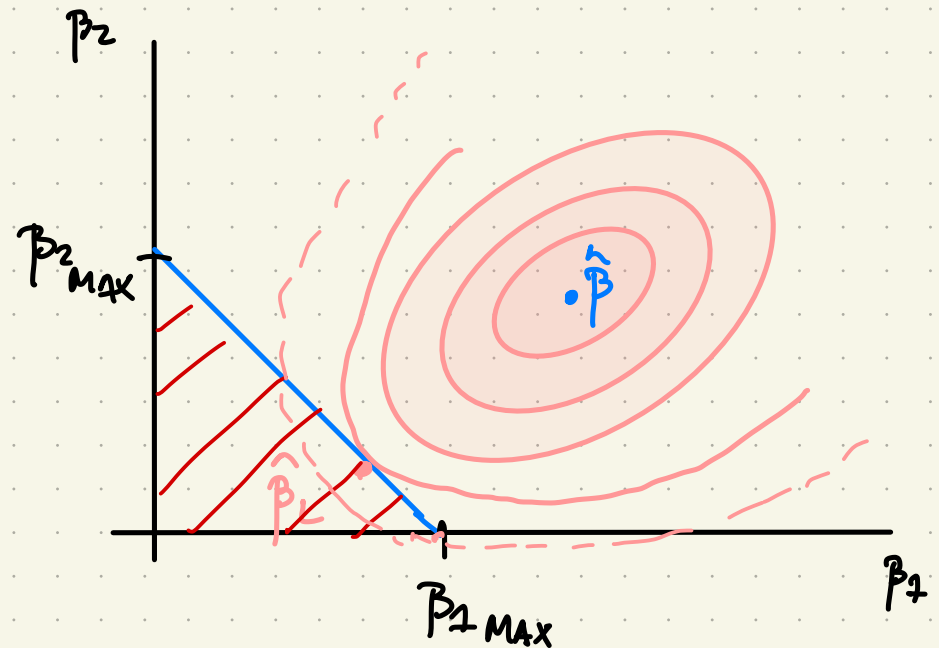
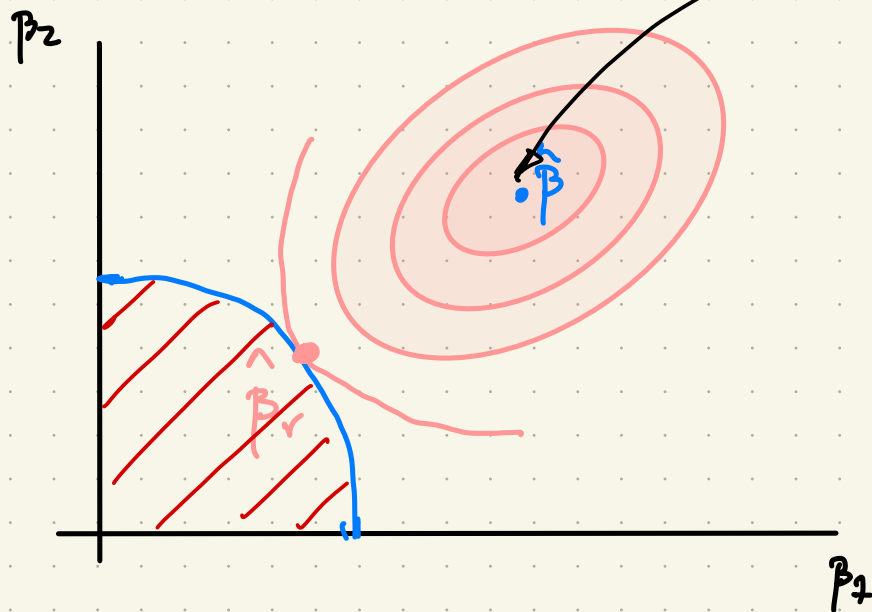
$$\phi(\beta)$$


$$w_{map} = (\Phi^T \Phi + \frac{\sigma^2}{b^2} I)^{-1} \Phi^T y$$

Gaussian prior con varianza b^2

Representación gráfica:

$\hat{\beta}$: Es el mejor $\hat{\beta}$ posible



 β 's válidos

Ridge
↓

porque enrobustece
mi sist.

$$- \lambda \sum_j |\beta_j|^2$$

Lasso
↓

porque hace un
proceso de selección
de variables

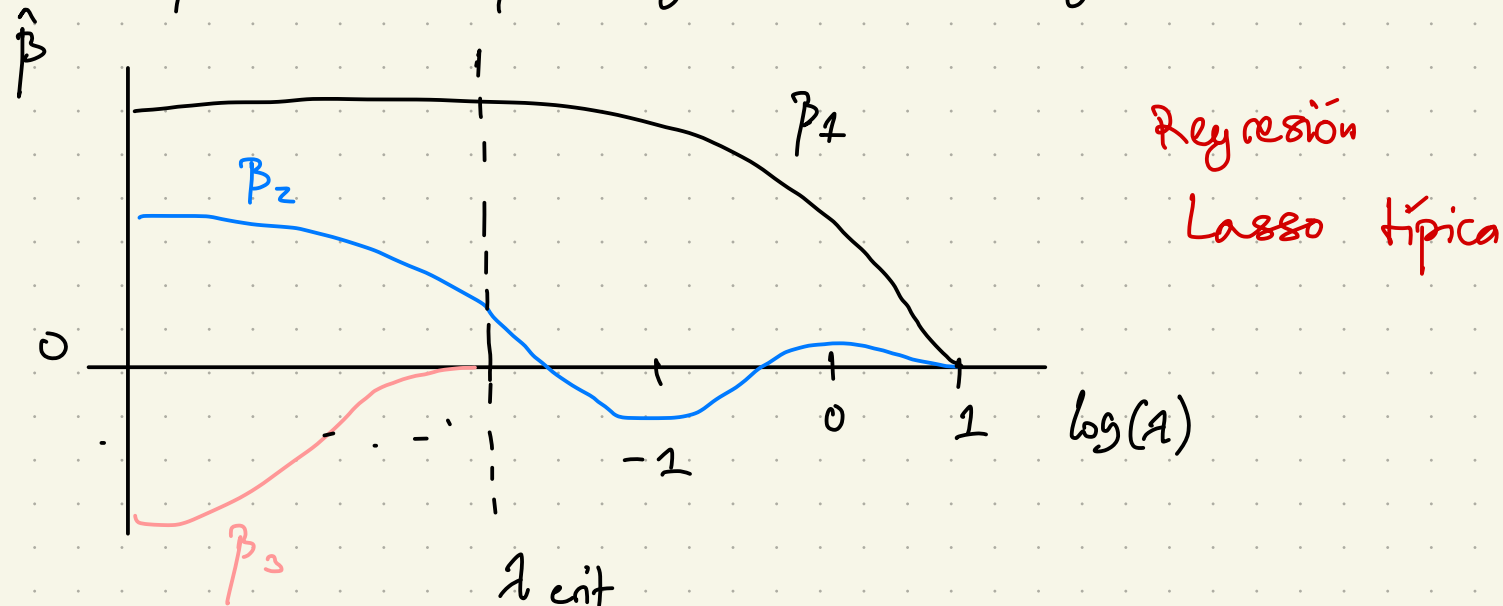
Recordemos que $\hat{\beta}$ en gal tiene curvas de nivel elípticas (por construcción del estimador).

Pero, regularizar viene a costo de alejarnos del óptimo.

¿Cómo funciona?

1. Elegir q (lasso: 1, Ridge: 2)
2. Voy a elegir un vector de λ 's "apropiados" (λ no q penalidad)
3. optimizo para λ_i no $RSS(\lambda; q) = \|y - X\beta\|^q + \lambda \|\beta\|_q$
4. Calcular las métricas (Error de representación, bondad, algún criterio externo).

5. Comparamos y elegimos el mejor:



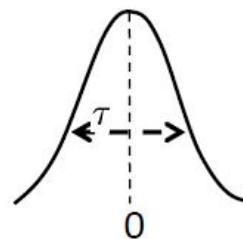
Maximum A Posteriori como regularización - Ridge (L2)

$$\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \underbrace{\log p(\{Y_i\}_{i=1}^n | \beta, \sigma^2, \{X_i\}_{i=1}^n)}_{\text{Conditional log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

I) Gaussian Prior

$$\beta \sim \mathcal{N}(0, \tau^2 \mathbf{I})$$

$$p(\beta) \propto e^{-\beta^T \beta / 2\tau^2}$$



$$\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \|\beta\|_2^2 \quad \text{Ridge Regression}$$

\downarrow
constant(σ^2, τ^2)

$$\hat{\beta}_{\text{MAP}} = (\mathbf{A}^T \mathbf{A} + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{Y}$$

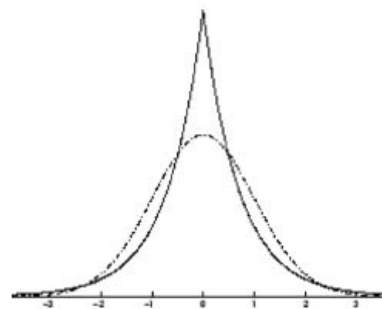
Maximum A Posteriori como regularización - LASSO (L1)

$$\hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \underbrace{\log p(\{Y_i\}_{i=1}^n | \beta, \sigma^2, \{X_i\}_{i=1}^n)}_{\text{Conditional log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

II) Laplace Prior

$$\beta_i \stackrel{iid}{\sim} \text{Laplace}(0, t)$$

$$p(\beta_i) \propto e^{-|\beta_i|/t}$$

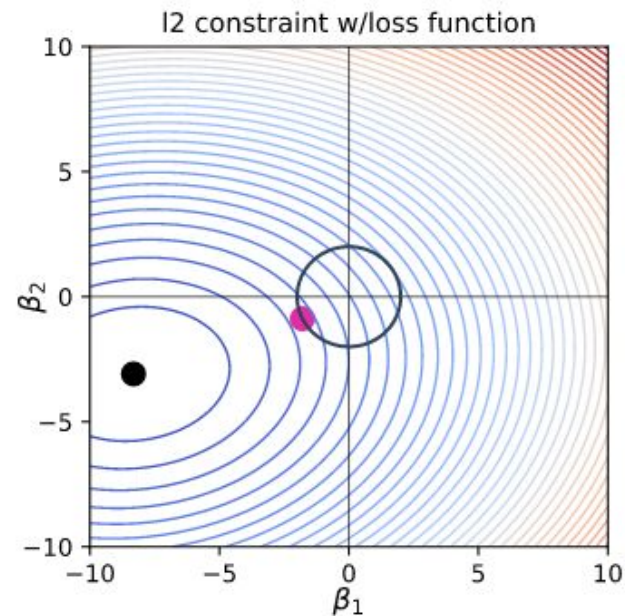
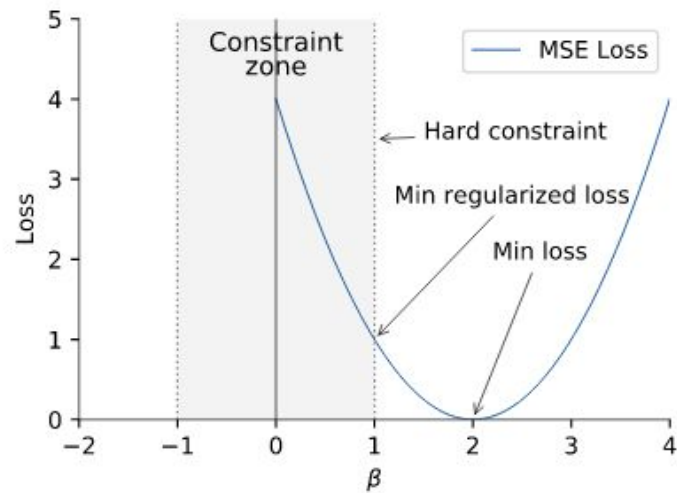


$$\hat{\beta}_{\text{MAP}} = \arg \min_{\beta} \sum_{i=1}^n (Y_i - X_i \beta)^2 + \lambda \|\beta\|_1$$

\downarrow
constant(σ^2, t)

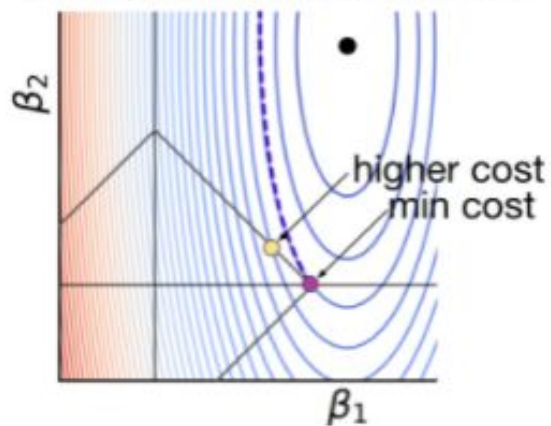
Lasso

Regularización

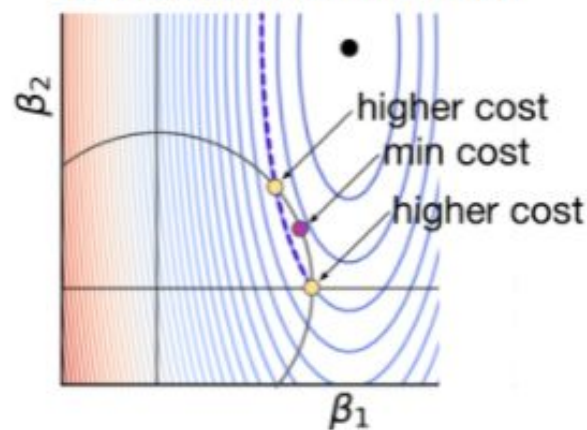


Regularización

(a) L1 Constraint Diamond



(b) L2 Constraint Circle

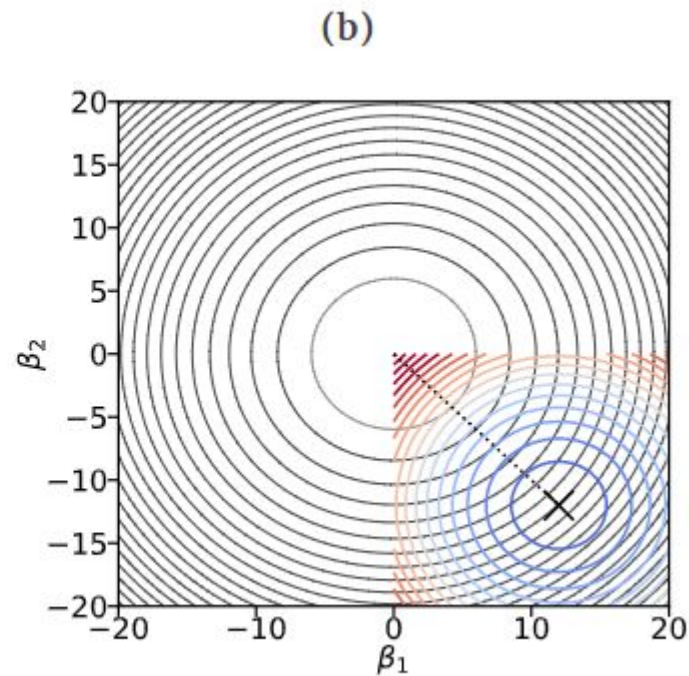
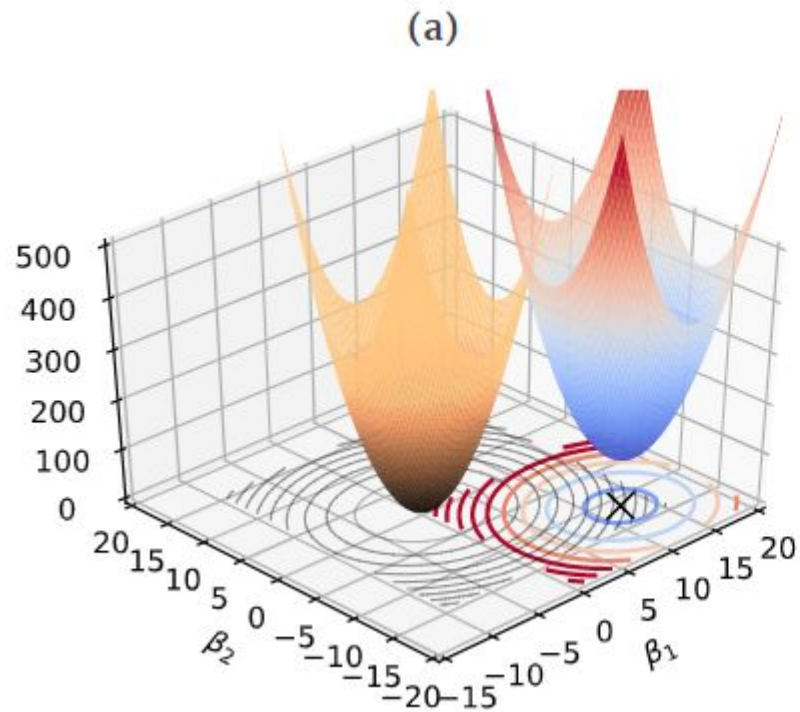


ElasticNet *λ término de complejidad*
¿Qué β se reduce más?

$$(\alpha\lambda\|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2)$$

α parametro de elasticnet

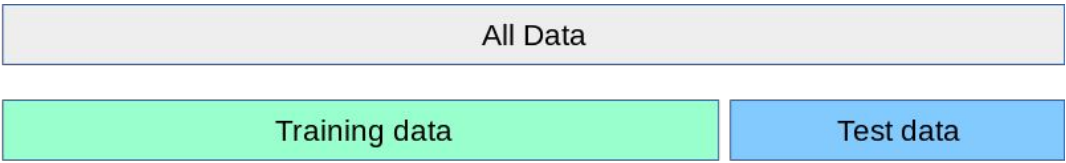
Regularización



Entrenamiento de modelos - Cross-Validation

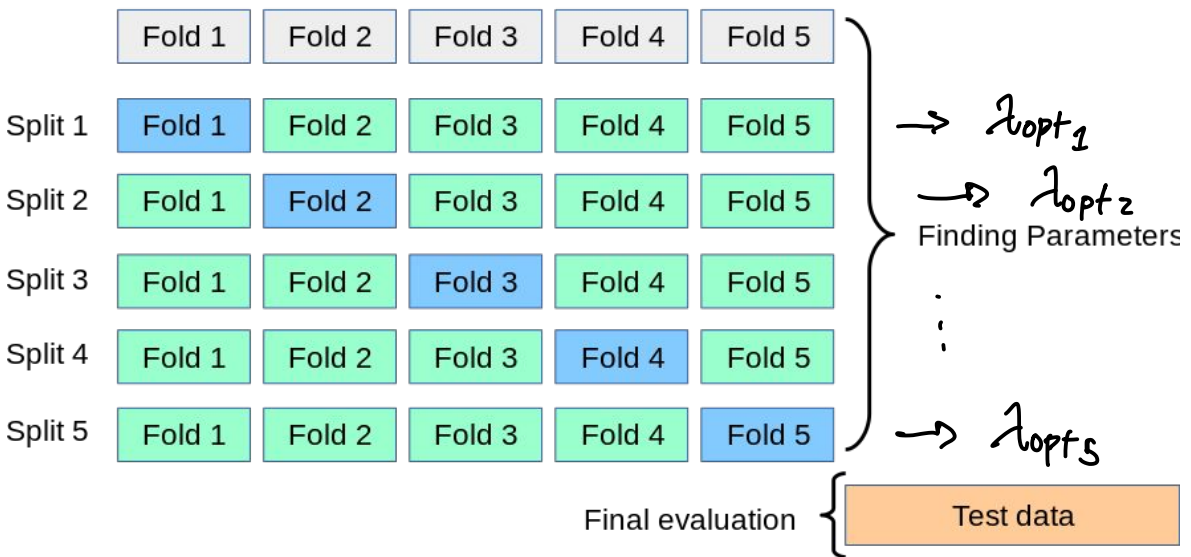
Cross-Validation

parameters n_folds folds en gral son subdivisiones random del dataset



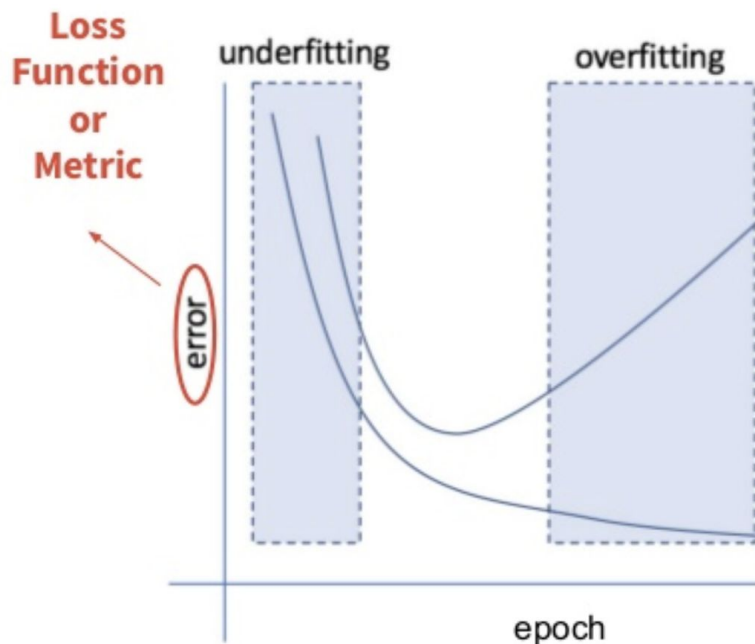
lasso
 $\lambda \in [-10, 5]$

X_t, y_t
 $X_{t_1} \quad X_{t_2} \quad X_{t_3} \quad X_{t_4} \quad X_{t_5}$
 $X_{t'}$



λ_{opt}

Entrenamiento numérico del modelo seleccionado - Obtención de parámetros

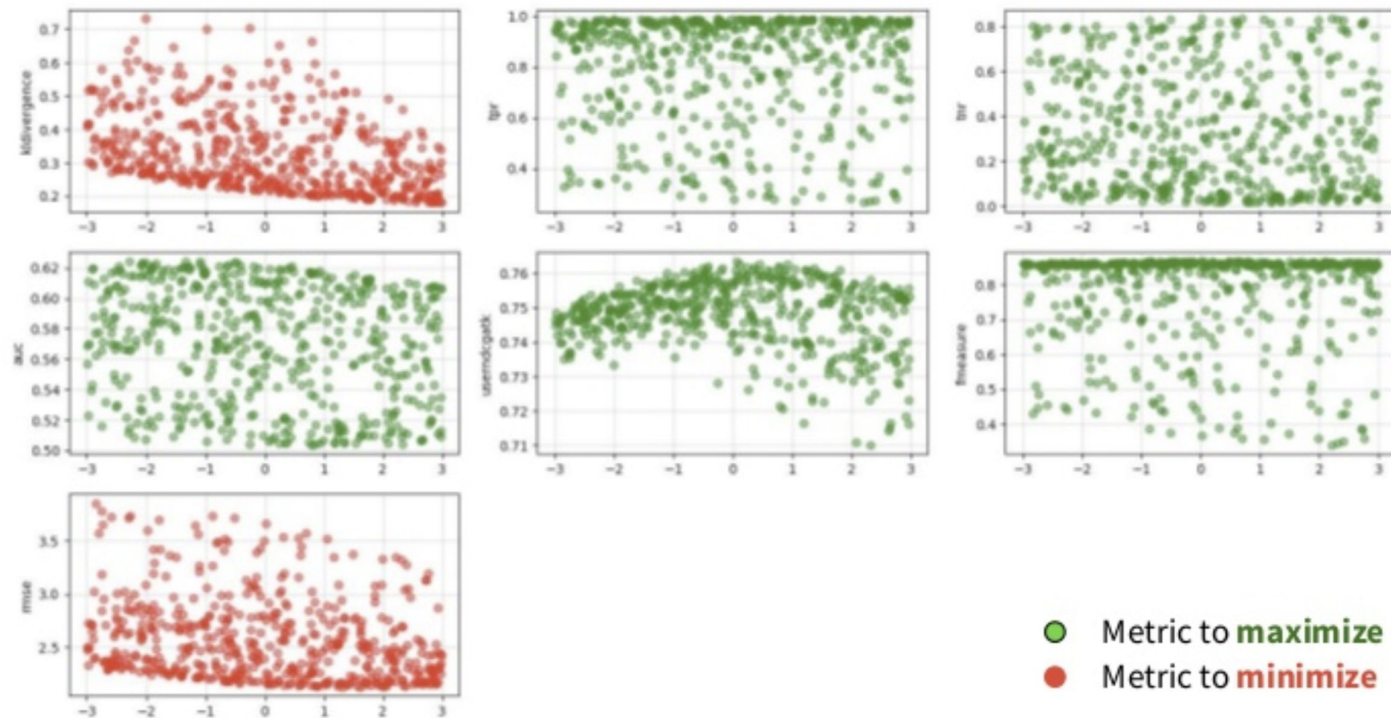


Mini-Batch Gradient Descent

for epoch in n_epochs:

- shuffle the batches
- **for batch in n_batches:**
 - compute the predictions for **the batch**
 - compute the error for the batch
 - compute the gradient for **the batch**
 - update the parameters of the model
- plot error vs epoch

Selección de los hiper parámetros



Grid Search

Random Search

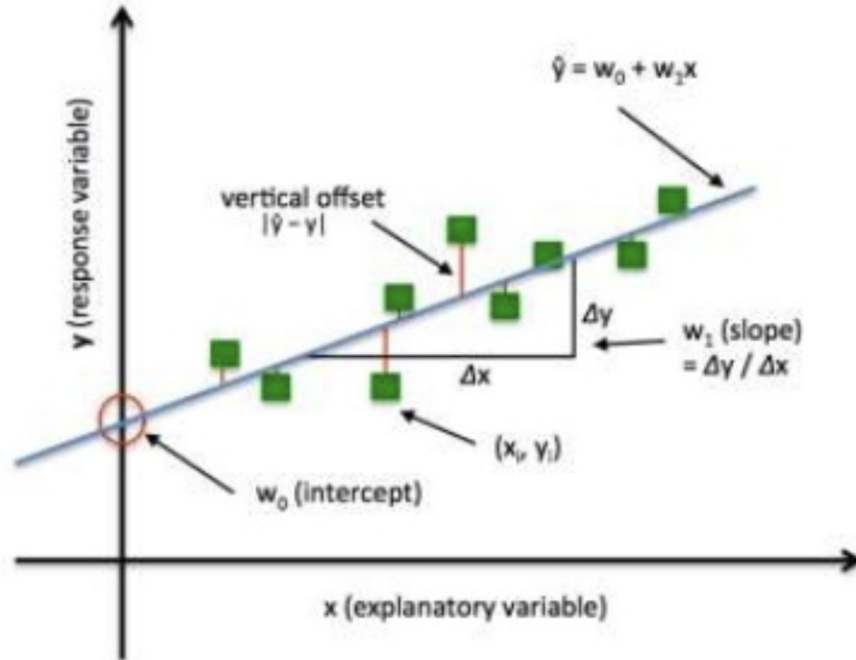
Gradiente Descendente

Implementación de Gradiente Descendente

Solucion analitica

$$\min_W \|Y - XW\|_2^2$$

$$W = (X^T X)^{-1} X^T Y$$

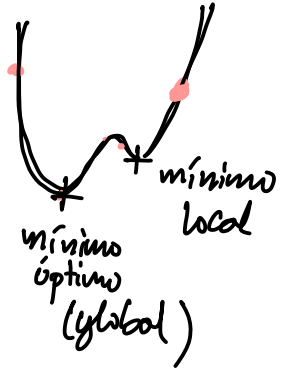


Implementación de Gradiente Descendente

GD (α, w_0)

Solución numérica

$$\min_W \|Y - XW\|_2^2 \Rightarrow \min_W \sum_i (y_i - X_i \cdot W)^2$$



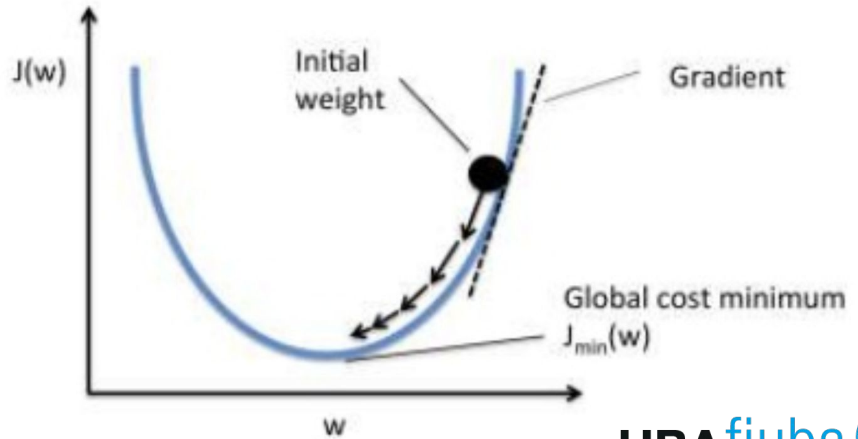
cond inicial

w_0

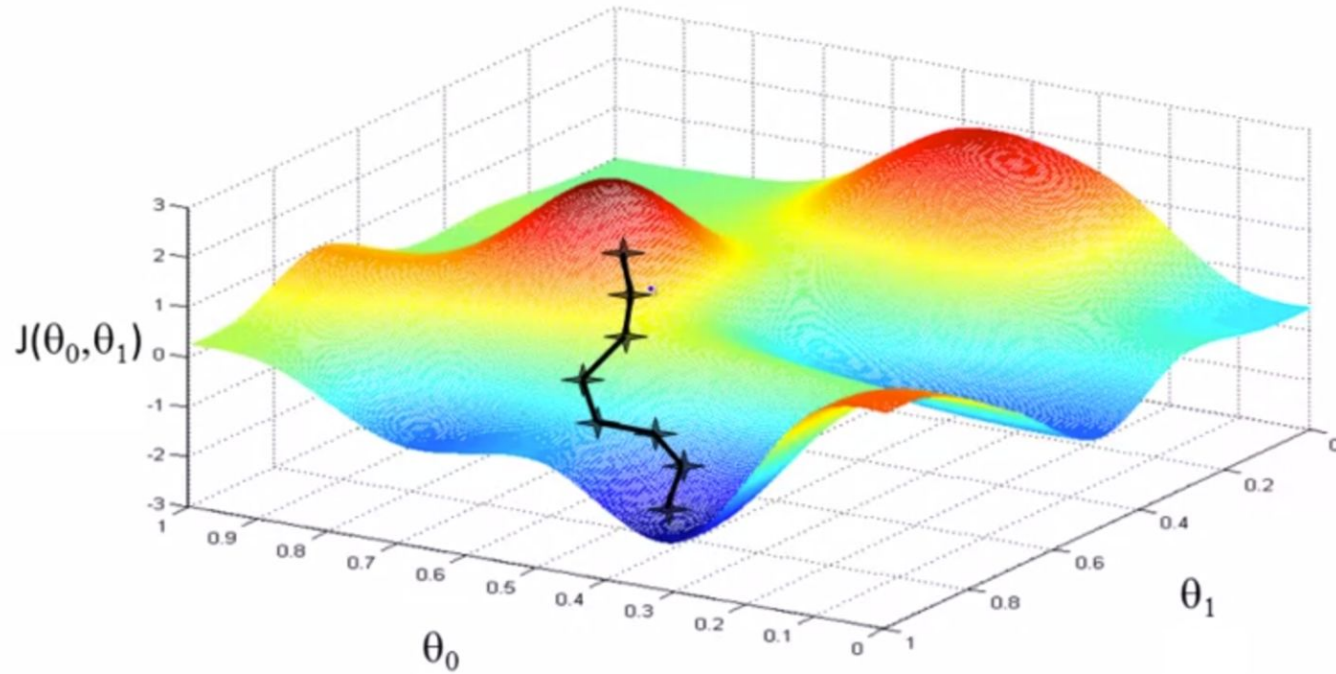
$$W \leftarrow W - \underbrace{\alpha \nabla \left(\sum_i (y_i - X_i \cdot W)^2 \right)}_{\text{"evolución"}}$$

learning rate

$\hookrightarrow \alpha \in \mathbb{R}^+$

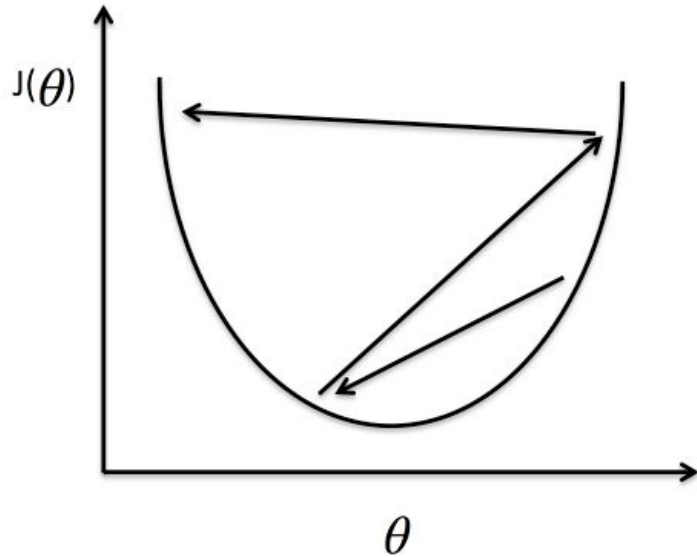


Gradiente Descendente

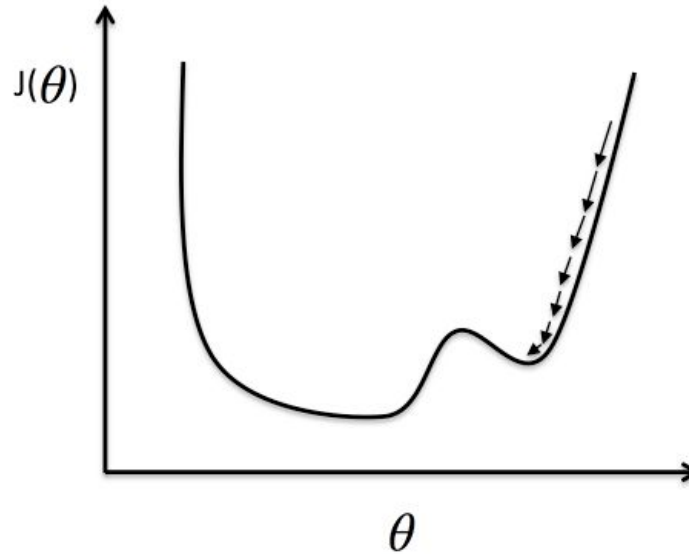


Andrew Ng

Gradiente Descendente



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

Implementación de Gradiente Descendente

Solución numérica

$$\begin{aligned}\nabla_w J(w) &= \nabla_w \left(\sum_i (y_i - X_i W)^2 \right) \\ &= \sum_i \left(\nabla_w (y_i - X_i W)^2 \right) \\ &= \sum_i \left(\nabla_w (y_i - (x_{i1}w_1 + x_{i2}w_2 + \cdots + x_{im}w_m))^2 \right) \\ &= \sum_i \left(-2(y_i - \hat{y}_i)x_{ij} \right) \quad \forall j \in (1 \cdots m)\end{aligned}$$

Implementación de Gradiente Descendente

Solución numérica

$$\nabla \left(\sum_{\text{all samples}} (y_i - f_W(X_i))^2 \right)$$

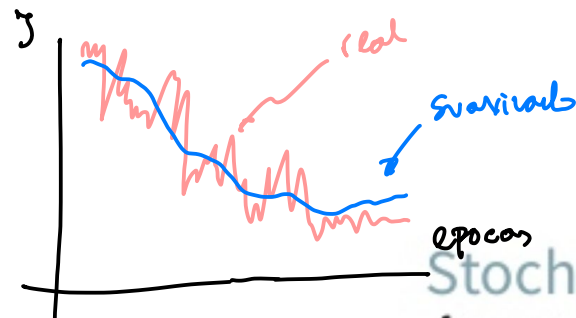
Gradient Descent algorithm

for epoch in n_epochs: *Alternativa while e < tol || n_epoch < n_epochs:*

- compute the predictions for **all the samples**
- compute the error between truth and predictions
- compute the gradient using **all the samples**
- update the parameters of the model

Implementación de Gradiente Descendente Estocástico

Solución numérica



$$\nabla ((y_i - f_W(X_i))^2)$$

$i \subset k$
es un subset
es la cant. total de muestras

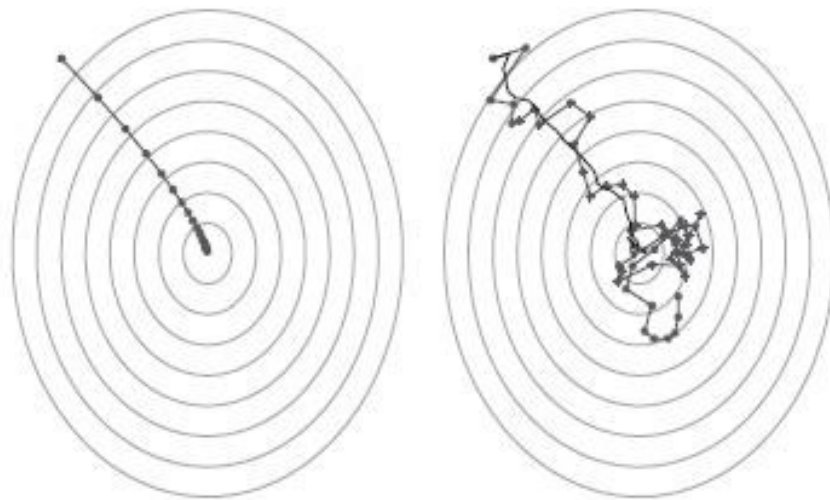
Stochastic Gradient Descent algorithm

for epoch in n_epochs:

- shuffle the samples
- **for sample in n_samples:**
 - compute the predictions for **the sample**
 - compute the error between truth and predictions
 - compute the gradient using **the sample**
 - update the parameters of the model

Implementación de Gradiente Descendente Estocástico

Solución numérica



Implementación de Gradiente Descendente Mini-Batch

Solución numérica

$$\nabla \left(\sum_{\text{batch samples}} (y_i - f_W(X_i))^2 \right)$$

Diagram illustrating the Mini-Batch Gradient Descent process. A box contains five batches: B_1 , B_2 , B_3 , B_4 , and B_5 . Below the box, two rows of batches are shown, representing the order of batches used in two different epochs:

Epoch 1: B_5, B_3, B_1, B_4, B_2

Epoch 2: B_1, B_4, B_3, B_5, B_2

Mini-Batch Gradient Descent algorithm *n_epoch, n_batch, batch-size*

for epoch in n_epochs:

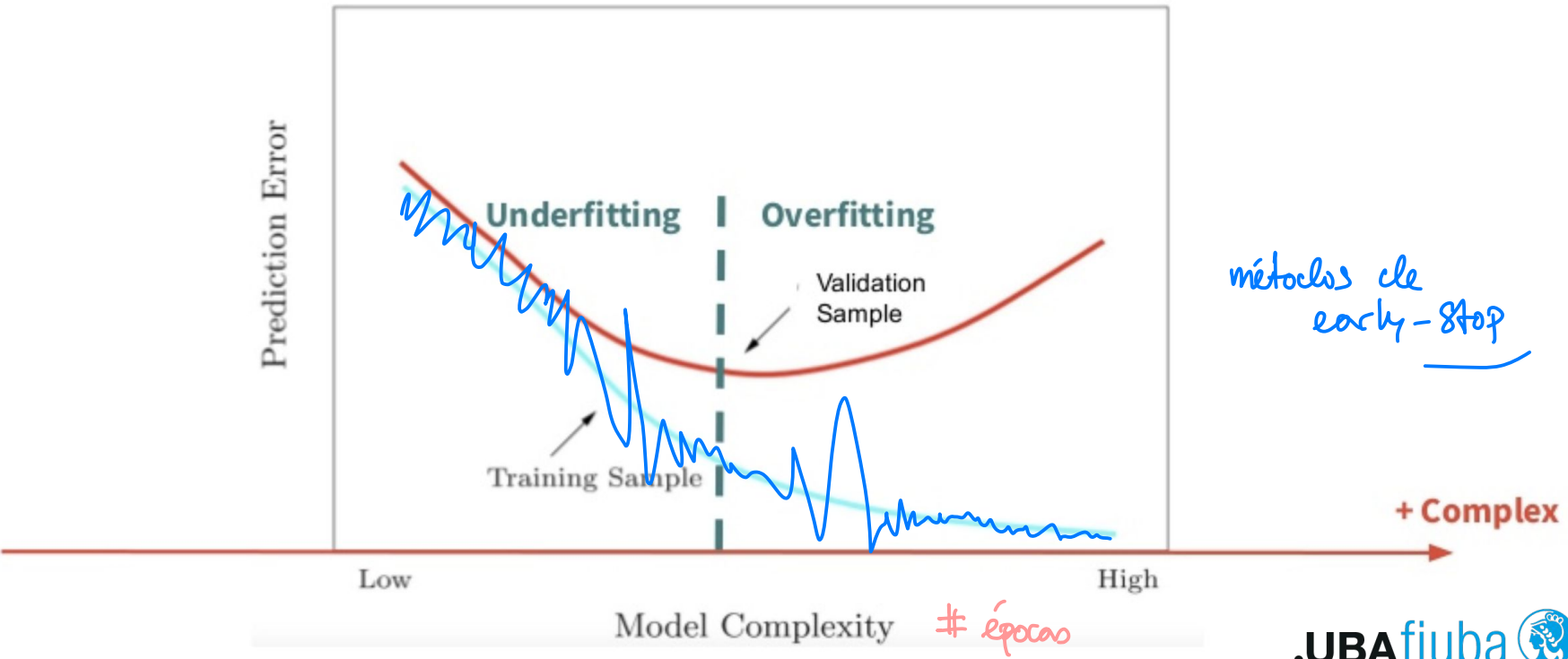
- shuffle the batches
- **for batch in n_batches:**
 - compute the predictions for **the batch**
 - compute the error for the batch
 - compute the gradient for **the batch**
 - update the parameters of the model

Comparativa de gradientes

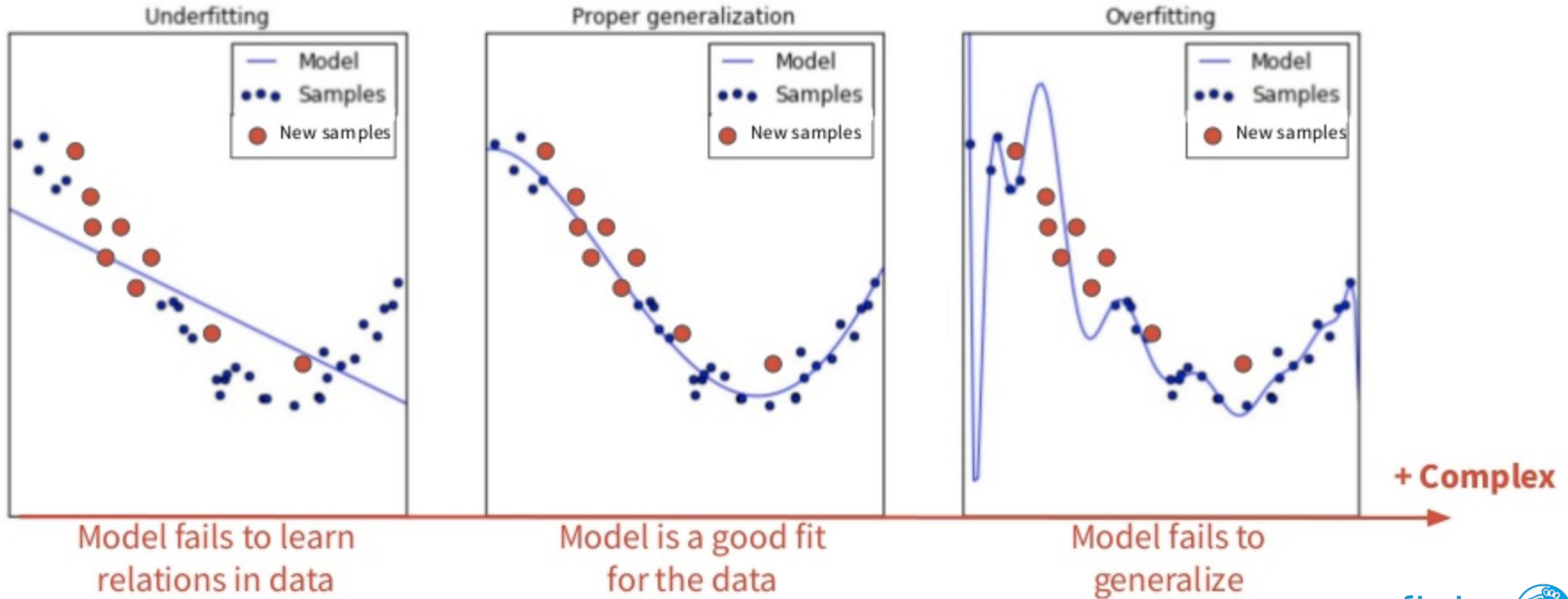
	Gradient Descent	Stochastic Gradient Descent	Mini-Batch Gradient Descent
Gradient	$\nabla \left(\sum_{\text{all samples}} (y_i - f_W(X_i))^2 \right)$	$\nabla \left((y_i - f_W(X_i))^2 \right)$	$\nabla \left(\sum_{\text{batch samples}} (y_i - f_W(X_i))^2 \right)$
Speed	Very Fast (vectorized)	Slow (compute sample by sample)	Fast (vectorized)
Memory	O(dataset)	O(1)	O(batch)
Convergence	Needs more epochs	Needs less epochs	Middle point between GD and SGD
Gradient Stability	Smooth updates in params	Noisy updates in params	Middle point between GD and SGD

Selección de modelos

Selección de modelos



Selección de modelos



Bibliografía

- The Elements of Statistical Learning | Trevor Hastie | Springer
- An Introduction to Statistical Learning | Gareth James | Springer
- Deep Learning | Ian Goodfellow | <https://www.deeplearningbook.org/>
- Mathematics for Machine Learning | Deisenroth, Faisal, Ong
- Artificial Intelligence, A Modern Approach | Stuart J. Russell, Peter Norvig
- A visual explanation for regularization of linear models - Terence Parr
- A Complete Tutorial on Ridge and Lasso Regression in Python - Aarshay Jain