Introducción a la Inteligencia Artificial

Clase 5

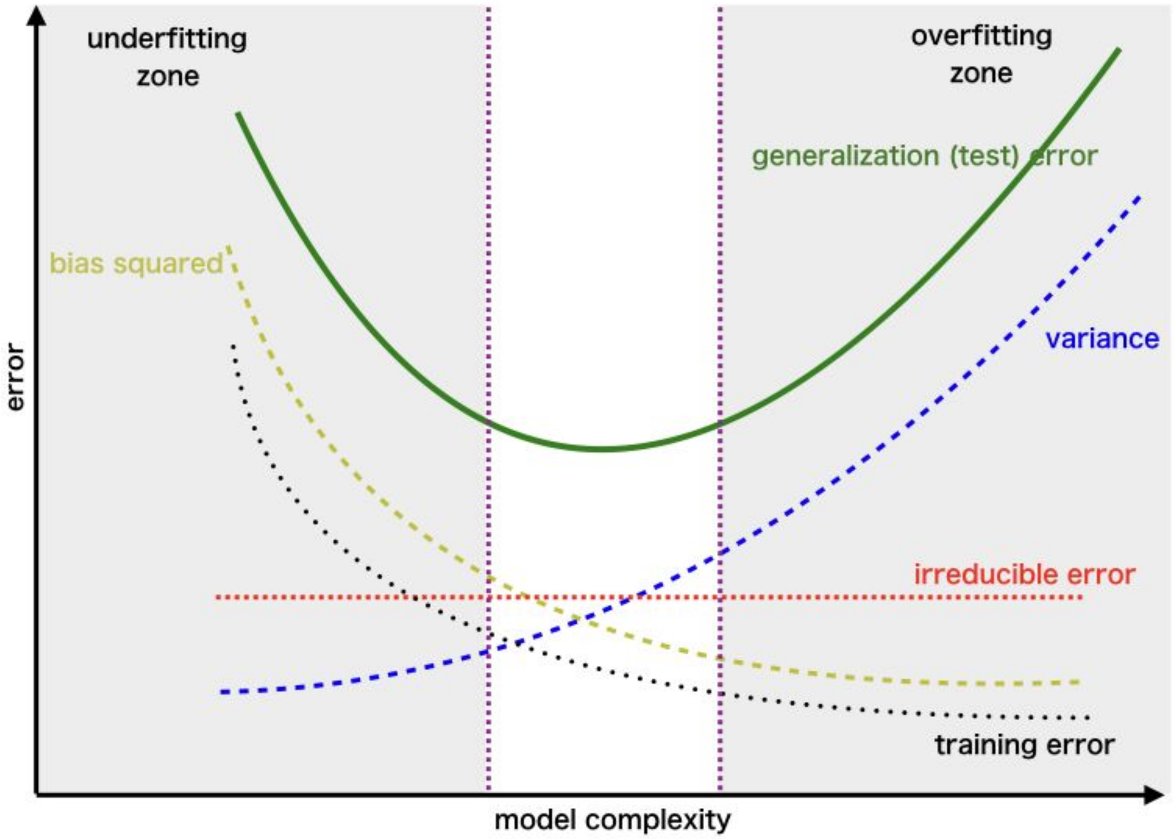
.UBA fiuba

FACULTAD DE INGENIERÍA

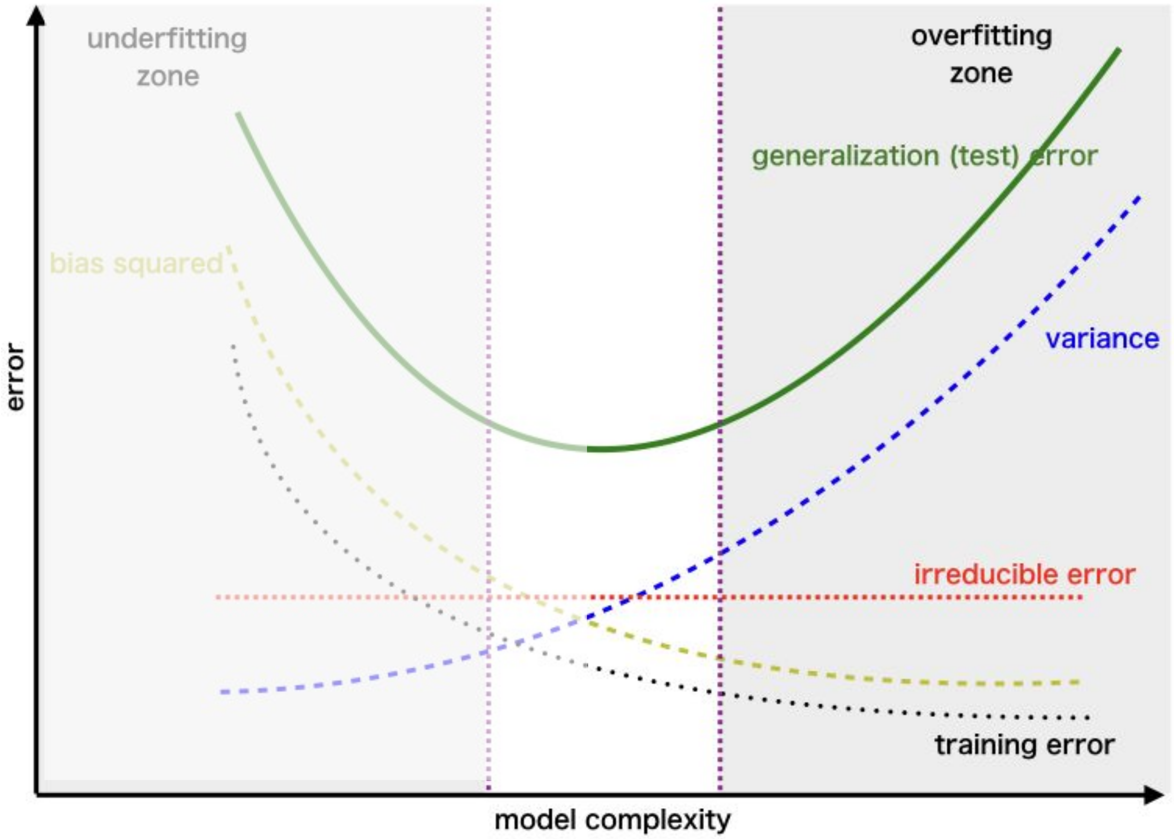
Índice

Clase 5

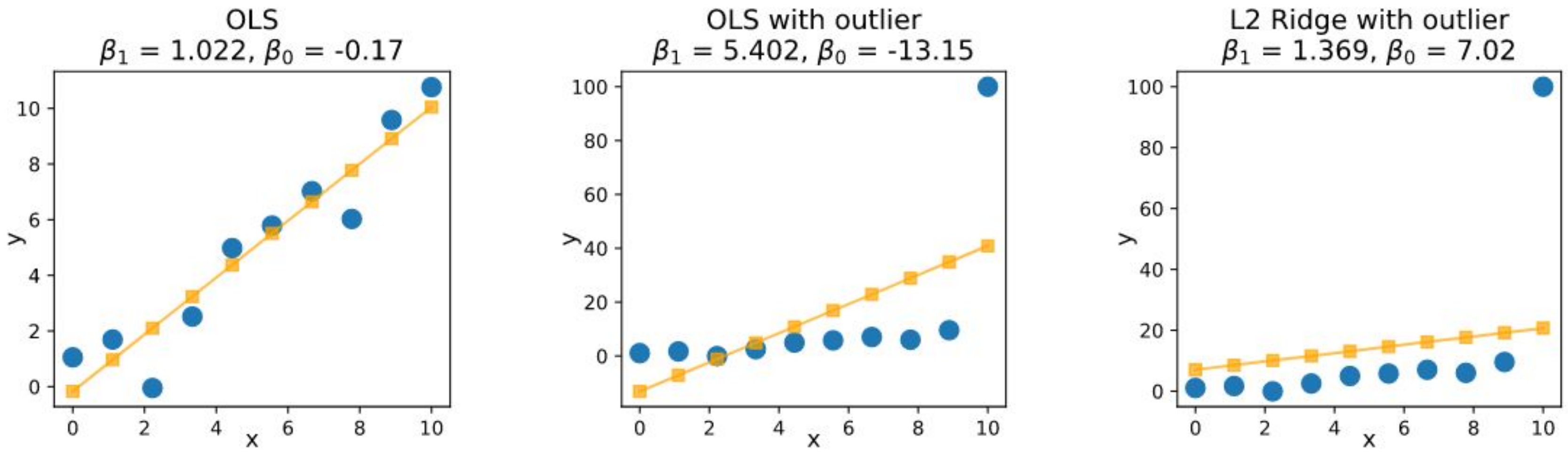
Regularización



Regularización



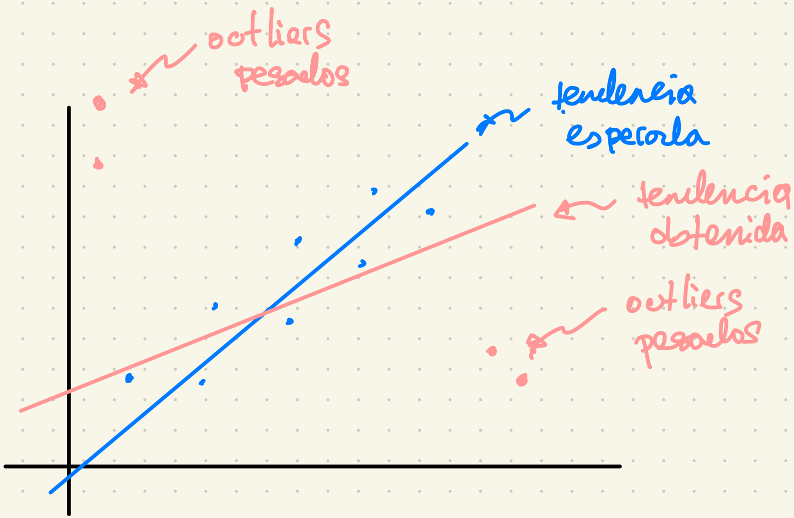
Regularización - Motivación



Riesgo Empírico

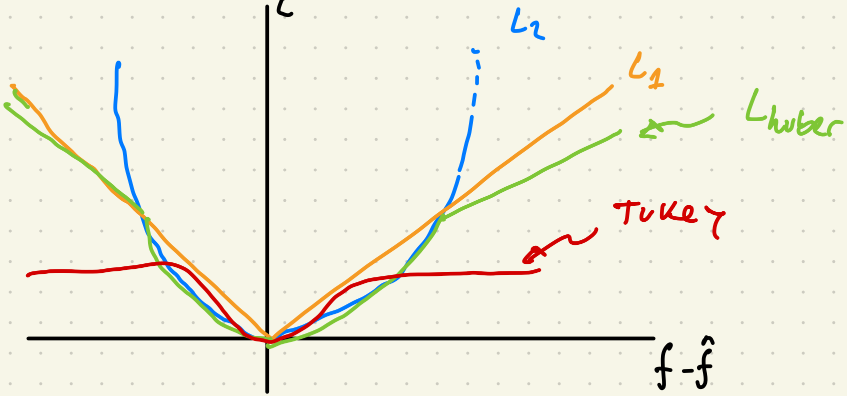
$$R(f) = \mathbb{E}(L(f,\hat{f})) \rightsquigarrow L(f,\hat{f}) = (f-\hat{f})^2 \to$$ pérdida cuadrática penalidad ℓ_2

hasta ahora usamos esto



¿cómo puedo mejorar mi tendencia obtenida?

+ Cambiar L:
  - $L_2 = (f-\hat{f})^2 \to \mathbb{E}(L_2) = ECM$
  - $L_1 = |f-\hat{f}| \to \mathbb{E}(L_1) = EAM$
  - $L_{\mathbb{I}} = \mathbb{I}_{|f-\hat{f}|>c} = \begin{cases} 0 & |f-\hat{f}|<c \\ 1 & ow \end{cases}$



Cuando queremos robustecer el Riesgo empírico vamos a elegir un L adecuado.

$\Rightarrow$ si utilizo modelos robustos $\Rightarrow$ en gral tengo más parámetros para optimizar

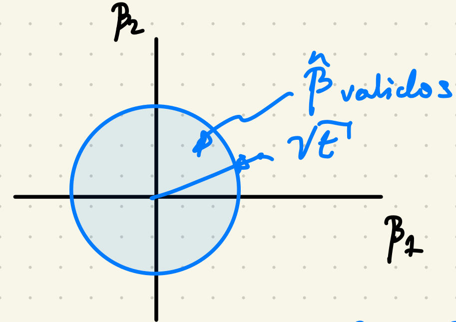
siempre buscamos minimizar R, si no se puede (o no se justifica) usar regresores robustos $\Rightarrow$ podemos regularizar.

Regularización:

Con reg. buscamos min. el riesgo emp R al mismo tiempo que restringimos (o limitamos) el comportamiento de los parámetros (parameter shrinkage). $\Rightarrow$ buscamos disminuir el error de representación

partimos de $\hat{y} = \beta_0 + \sum_i \beta_i X_i \Rightarrow \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_i \left(y_i - \sum_j \beta_j x_j\right)^2$

Vamos a condicionar β:



$$\begin{cases} \arg\min_{\beta} \sum_i (\cdots)^2 \\ \sum_j \beta_j^2 \le t \quad (\|\beta\| \le t) \end{cases}$$

con esto: $$\hat{\beta} = \arg\min_{\beta} \left(\sum_i y_i - \sum_j \beta_j x_j \right)^2 - \lambda \sum_j \beta_j^{2}$$

en gral es q.

término de regularización

parámetro de complejidad

termino de regularización = Weigth decay

Regularización

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

Observado - Predicción

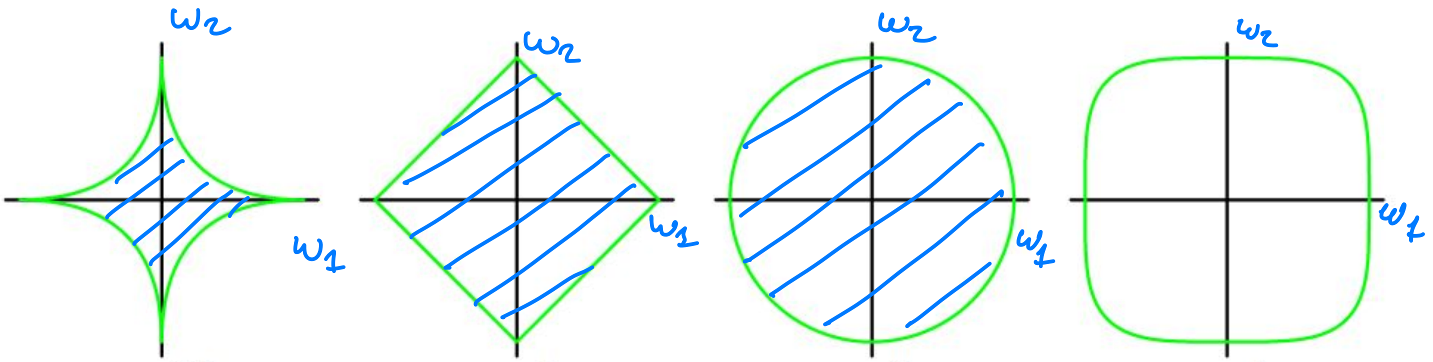
↓

w está “libre”

Regularización

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

Término de regularización "weight decay" → w afecta la pérdida



$q = 0.5$ $\quad$ $q = 1$ $\quad$ $q = 2$ $\quad$ $q = 4$

Lasso $\quad$ Ridge

$\hat{w}$ válidos

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

Regularización

Maximum A Posteriori como regularización

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

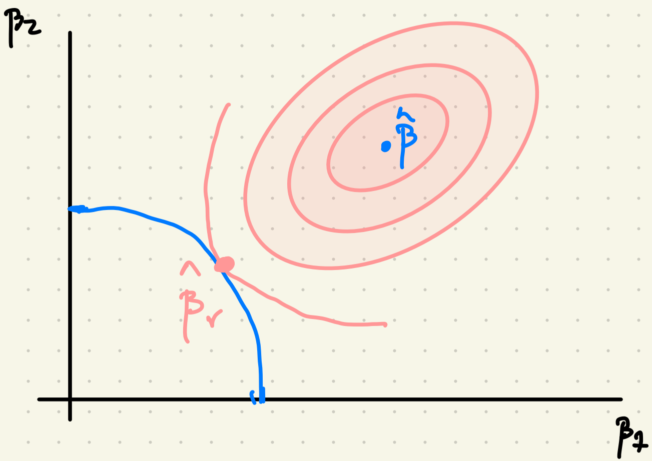
$$p(w) \sim D(\theta)$$

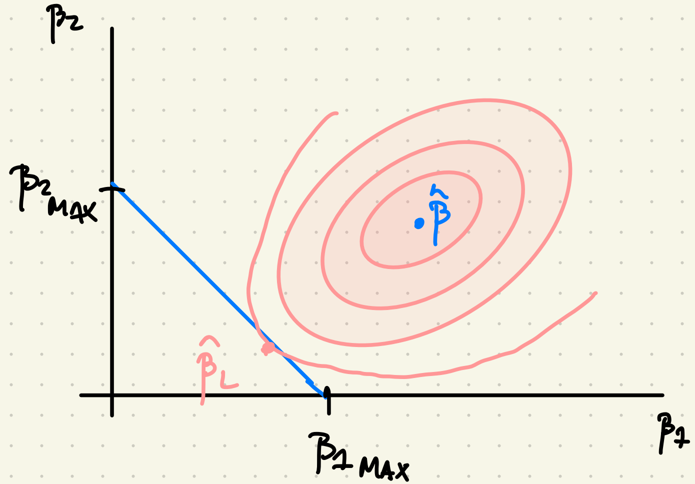
$$(\mathcal{X}, \mathcal{Y})$$

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

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

Gaussian prior con varianza b2





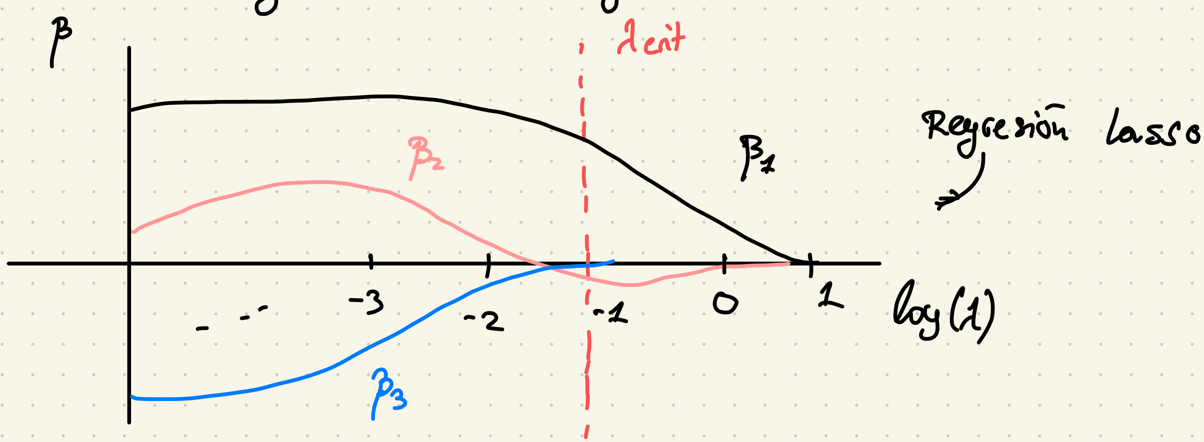
Ridge $\qquad - \lambda \sum_j |\beta_j|^q \qquad$ Lasso

Recordemos que $\hat{\beta}$ en gral tiene curvas de nivel elípticos, esto es por construcción del estimador.

El costo de regularizar es que nos alejamos del óptimo

¿Cómo funciona?

1. plantear (elegir) q (lasso 1, ridge 2)
2. elegimos un vector de λ's 'apropiado' ($\lambda \uparrow \Rightarrow \uparrow$ penalidad)
3. optimizar con un $\lambda_i \Rightarrow$ $RSS(\lambda; q) = \|y - X\beta\|^2 + \lambda \|\beta\|_q$
4. Calculamos las métricas (Error de representación, bondad de ajuste, etc...)
5. Comparamos y elegimos el mejor.

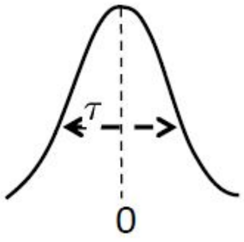


Maximum A Posteriori como regularización - Ridge (L2)

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

I) Gaussian Prior

$$\beta \sim \mathcal{N}(0, \tau^2 \mathbf{I}) \qquad p(\beta) \propto e^{-\beta^T \beta / 2\tau^2}$$



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \underbrace{\lambda}_{\text{constant}(\sigma^2, \tau^2)} \|\beta\|_2^2$$

Ridge Regression

$$\widehat{\beta}_{\text{MAP}} = (\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{A}^\top \boldsymbol{Y}$$

Machine Learning 10-315, Carnegie Mellon

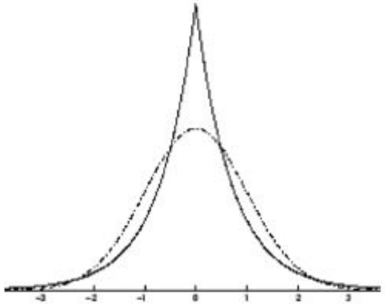
Regularización

Maximum A Posteriori como regularización - LASSO (L1)

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

II) Laplace Prior

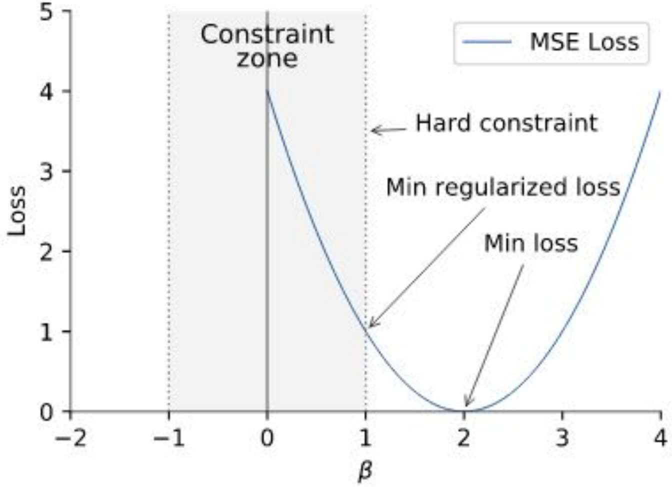
$$\beta_i \overset{iid}{\sim} \text{Laplace}(0, t) \qquad\qquad p(\beta_i) \propto e^{-|\beta_i|/t}$$

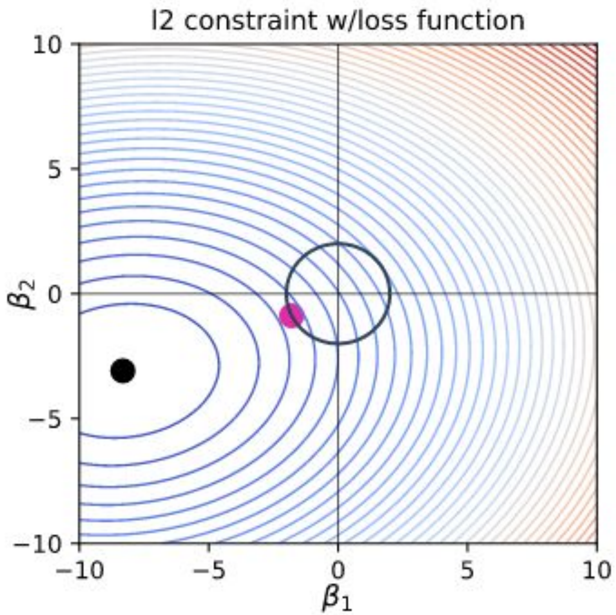


$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \underbrace{\lambda}_{\text{constant}(\sigma^2, t)} \|\beta\|_1$$

Lasso

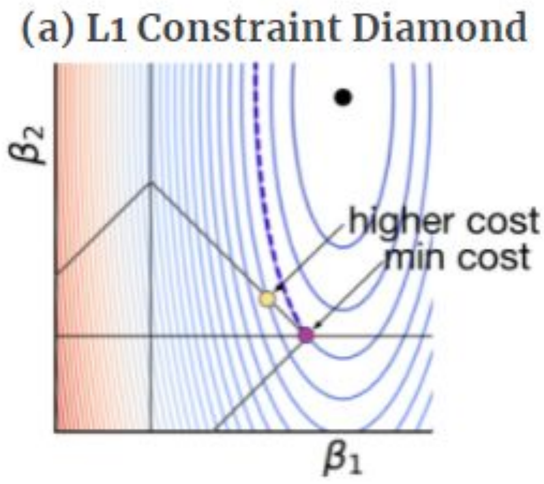
Regularización

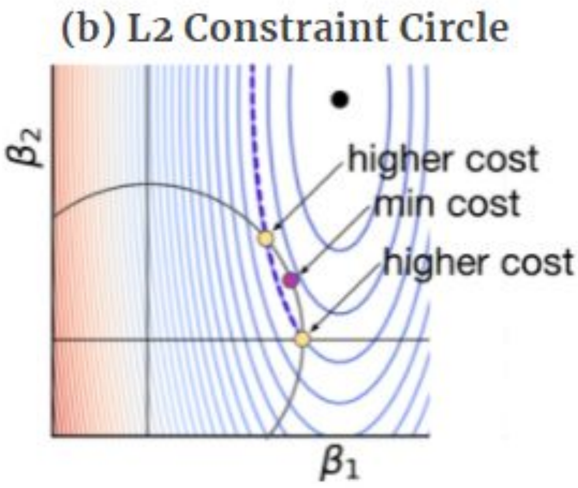




Ejemplo interactivo:
https://shinylive.io/py/app/#regularization

Regularización



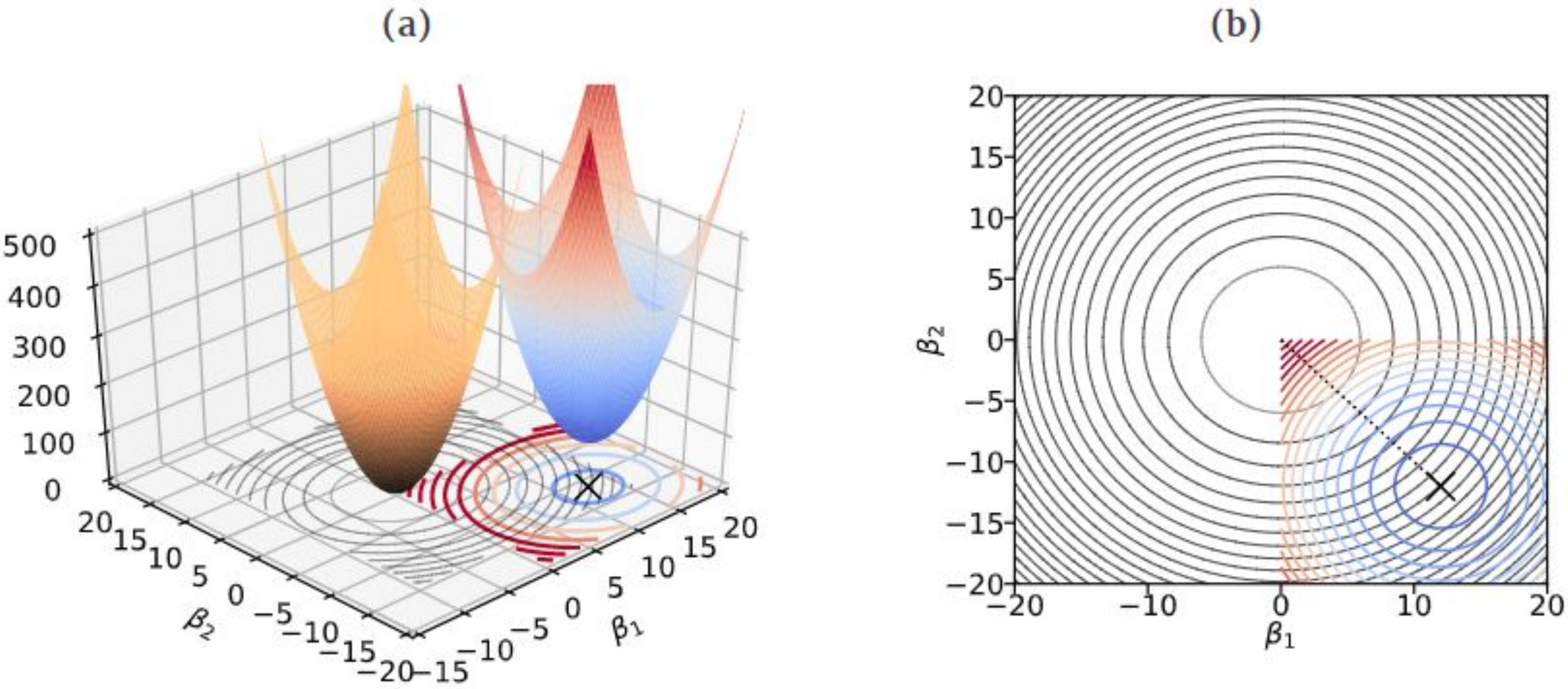


ElasticNet

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

¿Qué β se reduce más?

Regularización



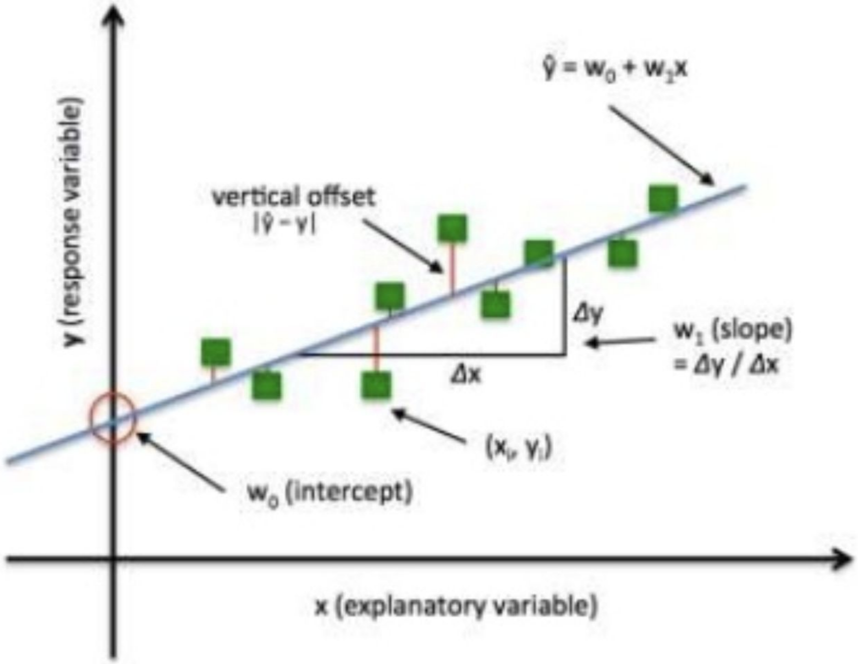
Gradiente Descendente

Implementación de Gradiente Descendente

Solucion analitica

$$\min_{W} \|Y - XW\|_2^2$$

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

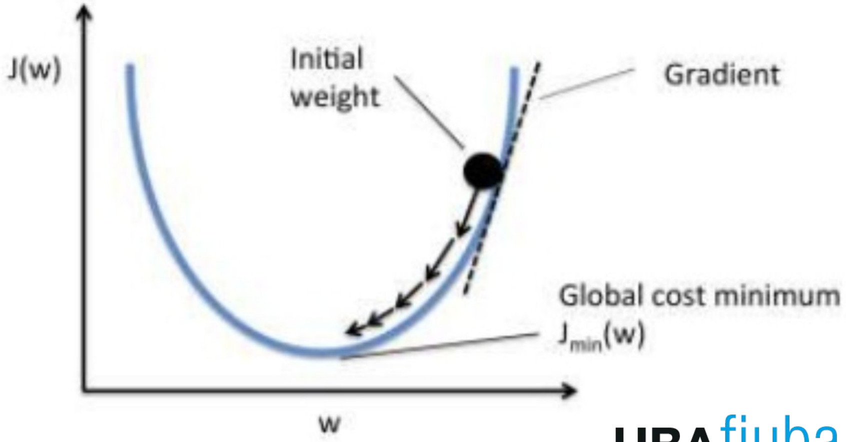


Implementación de Gradiente Descendente

Solución numérica

$$\min_{W} \|Y - XW\|_2^2 \implies \min_{W} \sum_i \left(y_i - X_i \cdot W\right)^2$$

$$W \longleftarrow W - \alpha \nabla \left(\sum_i (y_i - X_i \cdot W)^2 \right)$$

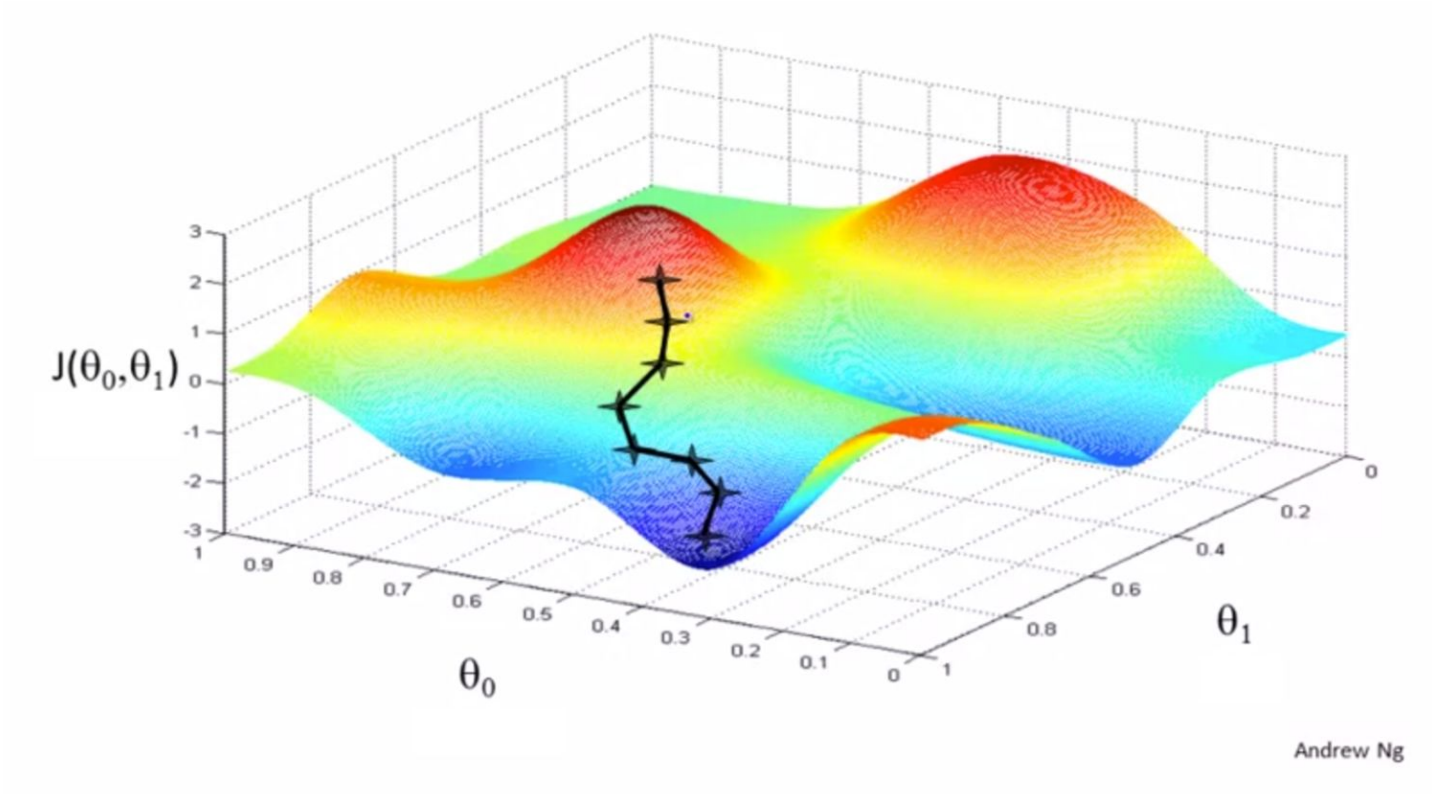


$$w_{t+1} = w_t - \alpha \nabla R$$

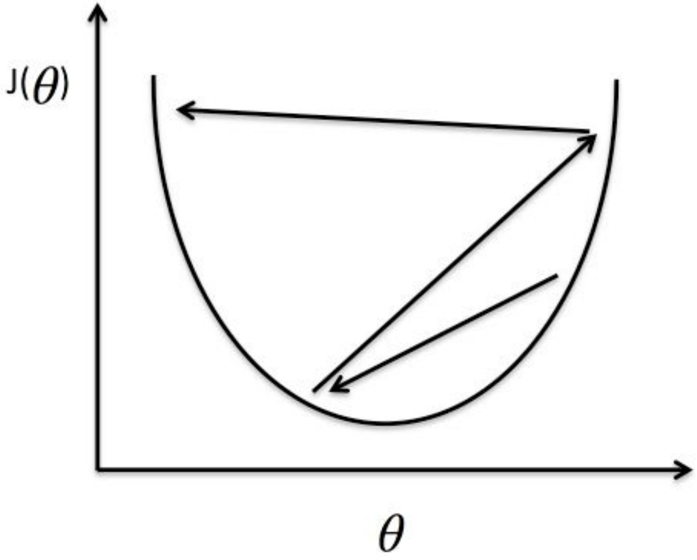
gradiente del funcional

learning rate

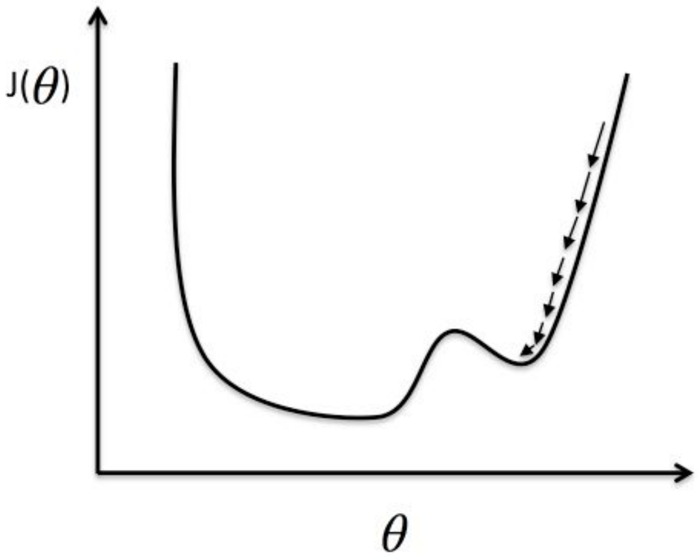
Gradiente Descendente



Gradiente Descendente



Large learning rate: Overshooting.



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

.UBAfiuba FACULTAD DE INGENIERÍA

Implementación de Gradiente Descendente

Solución numérica

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

en sklearn Linear_models. SGDRegressor

Implementación de Gradiente Descendente

$W_{t+1} = W_t - \alpha \nabla_w J(w)$

Solución numérica

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

Gradient Descent algorithm (dado $W_{inicial}$)

for epoch in 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

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

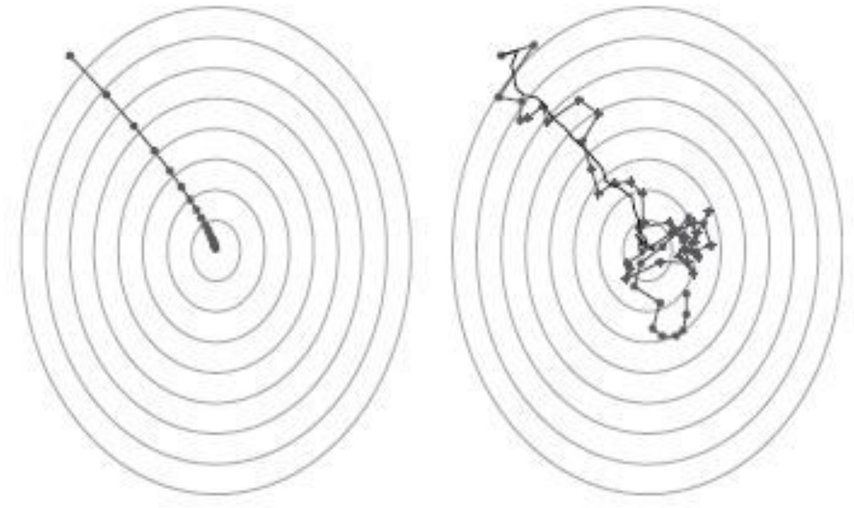
Stochastic Gradient Descent algorithm

for epoch in n_epochs: (samples = 10^6)

- shuffle the samples n_samples = 10^2
- 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

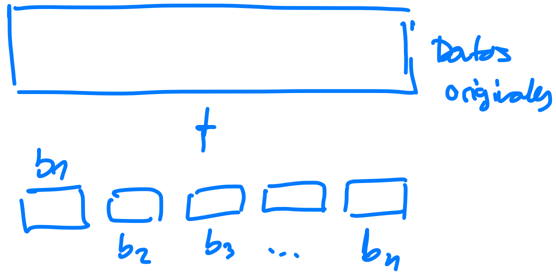
Implementación de Gradiente Descendente Estocástico

Solución numérica



Implementación de Gradiente Descendente Mini-Batch

Solución numérica



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

Mini-Batch Gradient Descent algorithm

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

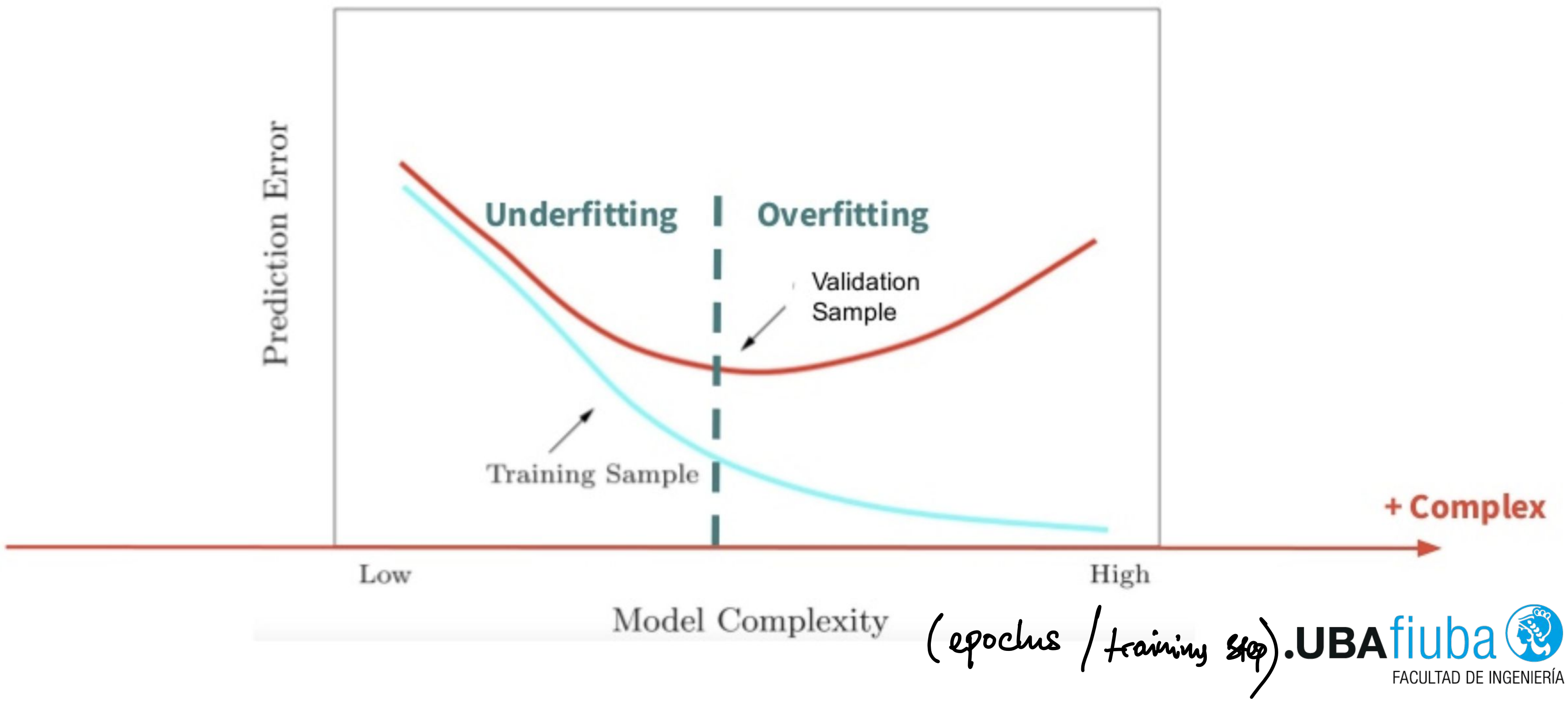
\sum_i \Big(-2 (y_i - \hat{y}_i) x_{ij} \Big)

Comparativa de gradientes

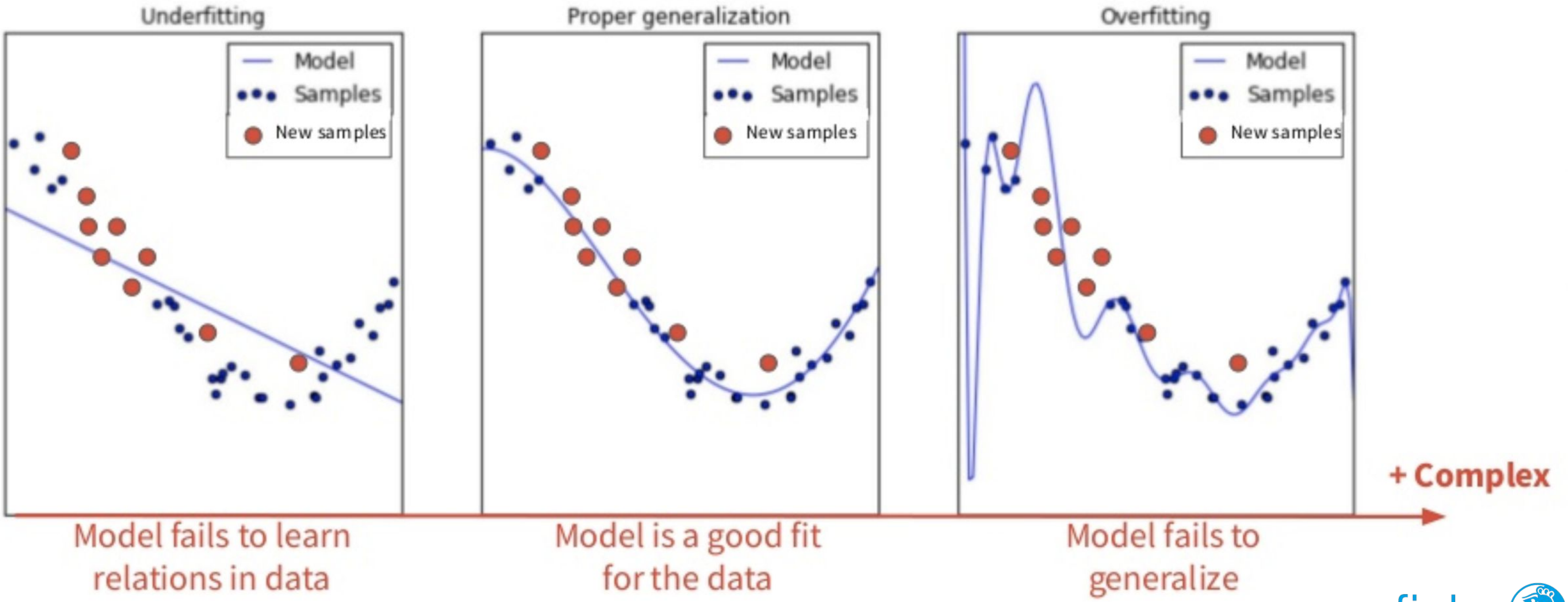
	Gradient Descent	Stochastic Gradient Descent	Mini-Batch Gradient Descent
Gradient	$\nabla\left(\sum_{\text{all samples}} (y_i - f_W(X_i))^2\right)$	$\nabla\left((y_i - f_W(X_i))^2\right)$	$\nabla\left(\sum_{\text{batch samples}} (y_i - f_W(X_i))^2\right)$
Speed	Very Fast (vectorized)	Slow (compute sample 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

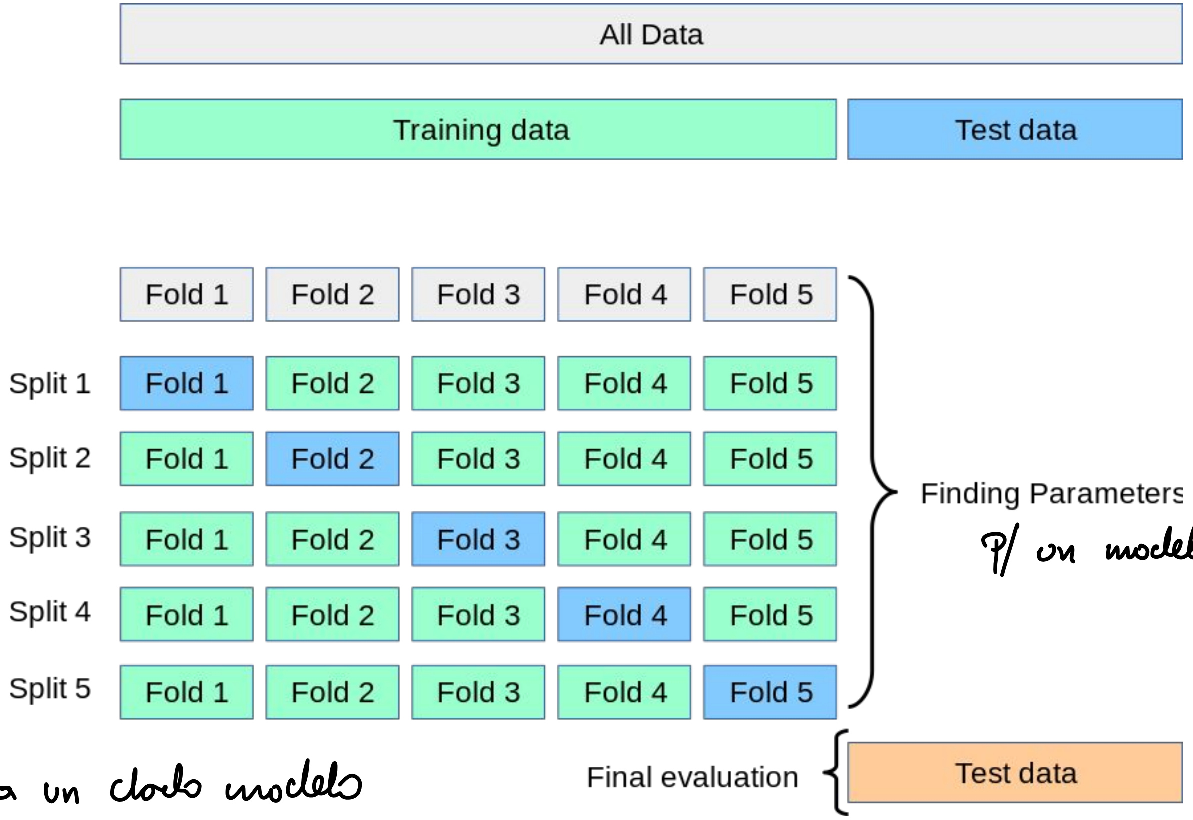


Selección de modelos



Entrenamiento de modelos - Cross-Validation

Cross-Validation



modelos = { ... }

SVM (K=1, dist=eu)

SVM (K=rbf, dist eu)

SVM (K=rbf, dis=M)

P/ un modelo

$Ac_{s1} \rightarrow$ Split 1

$Ac_{s2} \rightarrow$ Split 2

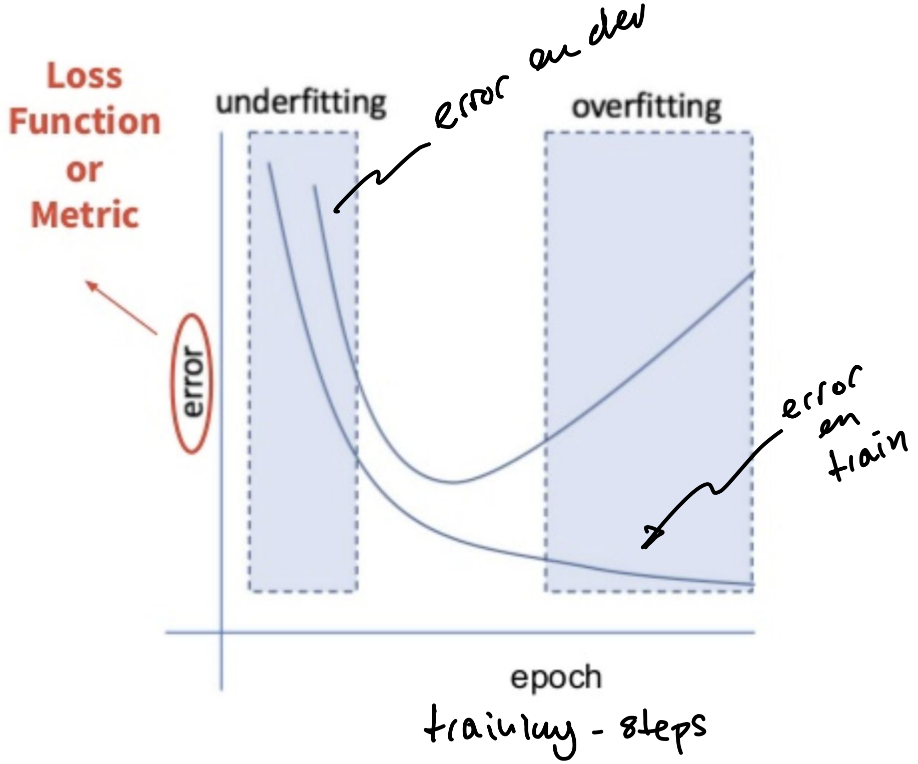
⋮

$Ac_{s5} \rightarrow$ Split 5

$\overline{Ac}$ para un dado modelo

.UBAfiuba FACULTAD DE INGENIERÍA

Entrenamiento numérico del modelo seleccionado - Obtención de parámetros



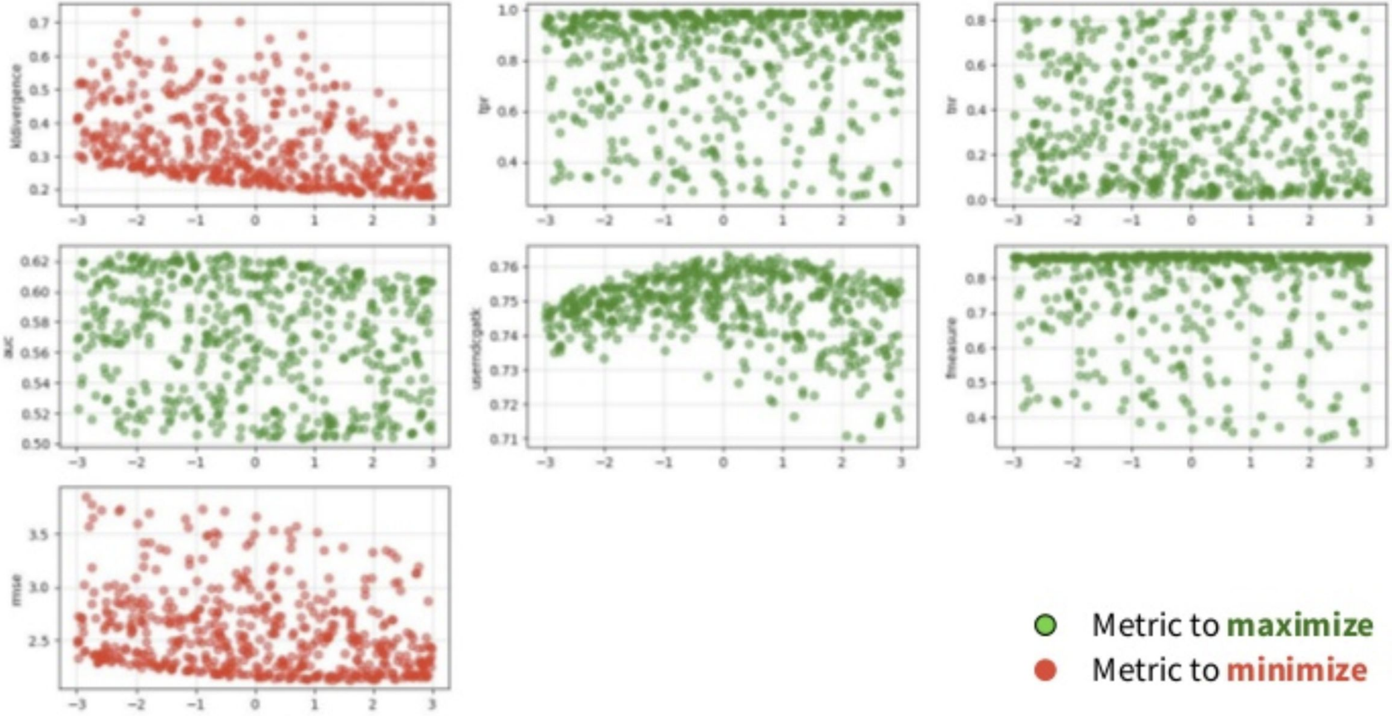
Dataset → train 80%
↳ Dev 5%
↳ test 15%

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

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
- Deep Learning con Pytorch, Chintala