Introducción a la Inteligencia Artificial
Clase 4

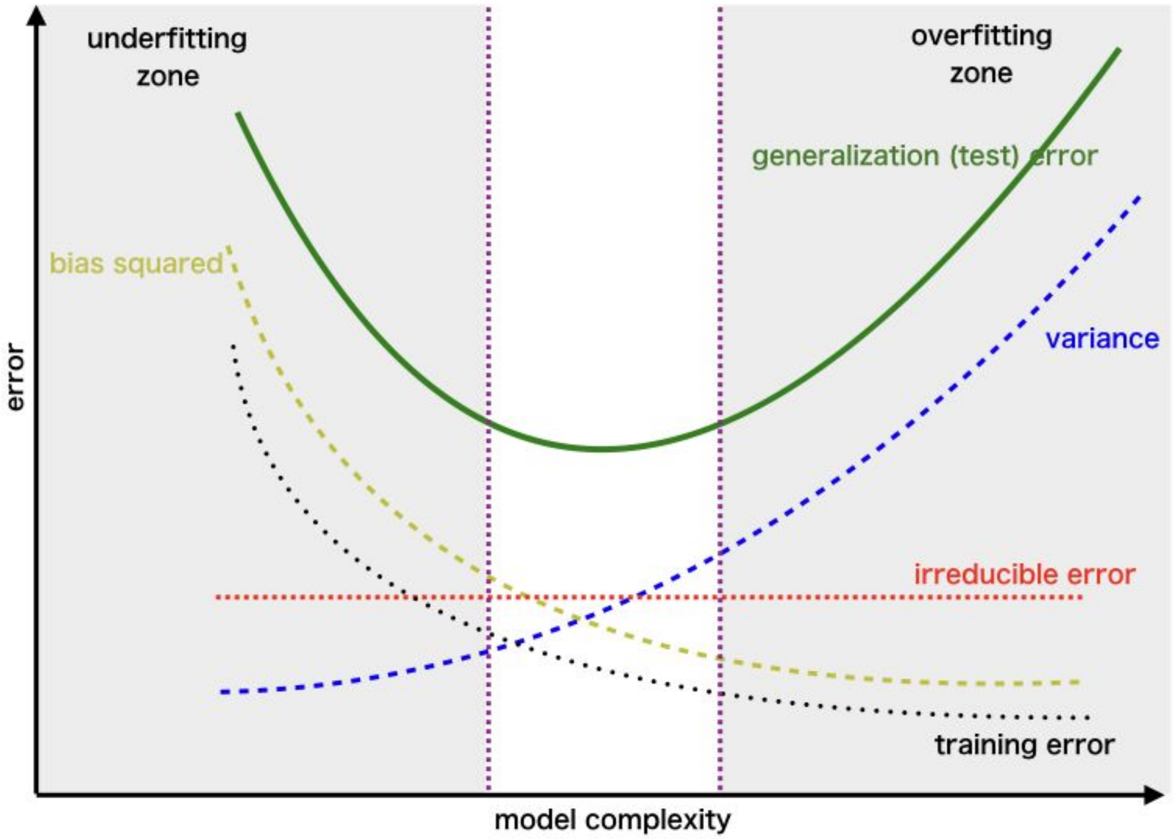
.UBA fiuba
FACULTAD DE INGENIERÍA

Índice

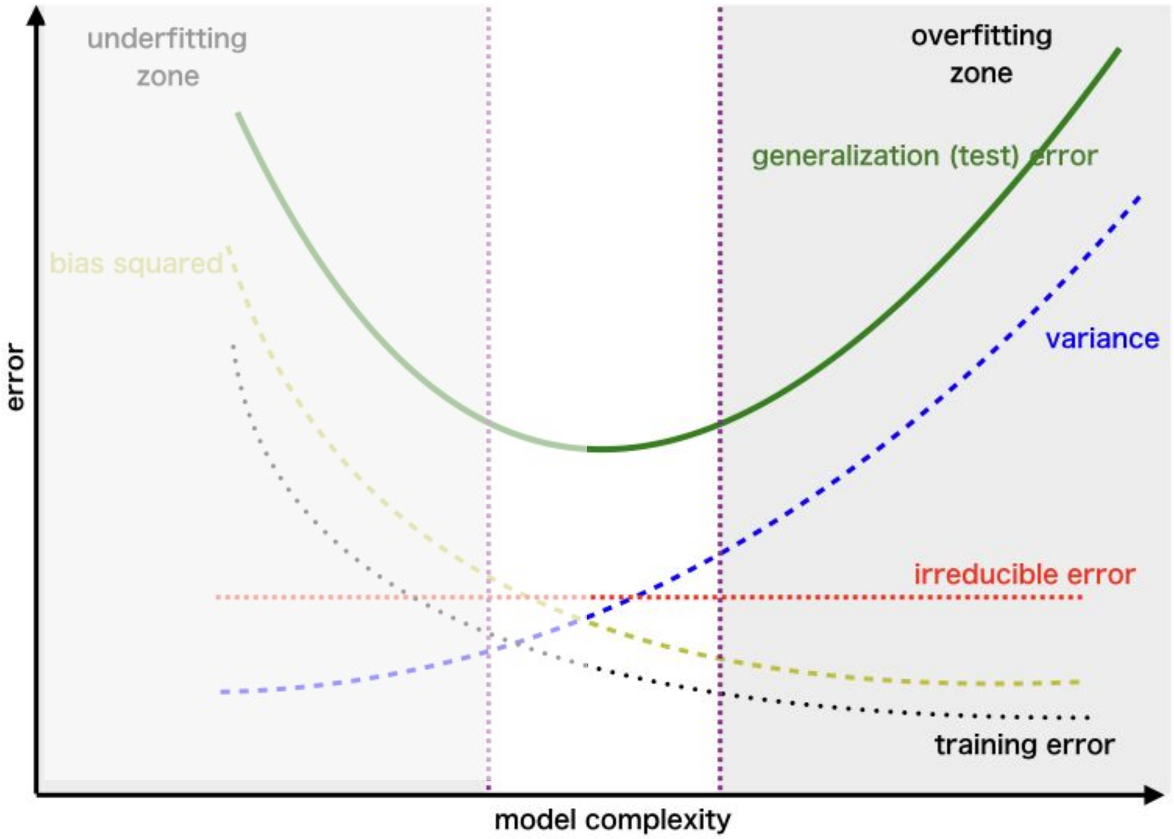
Clase ~~5~~ 4

0. 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

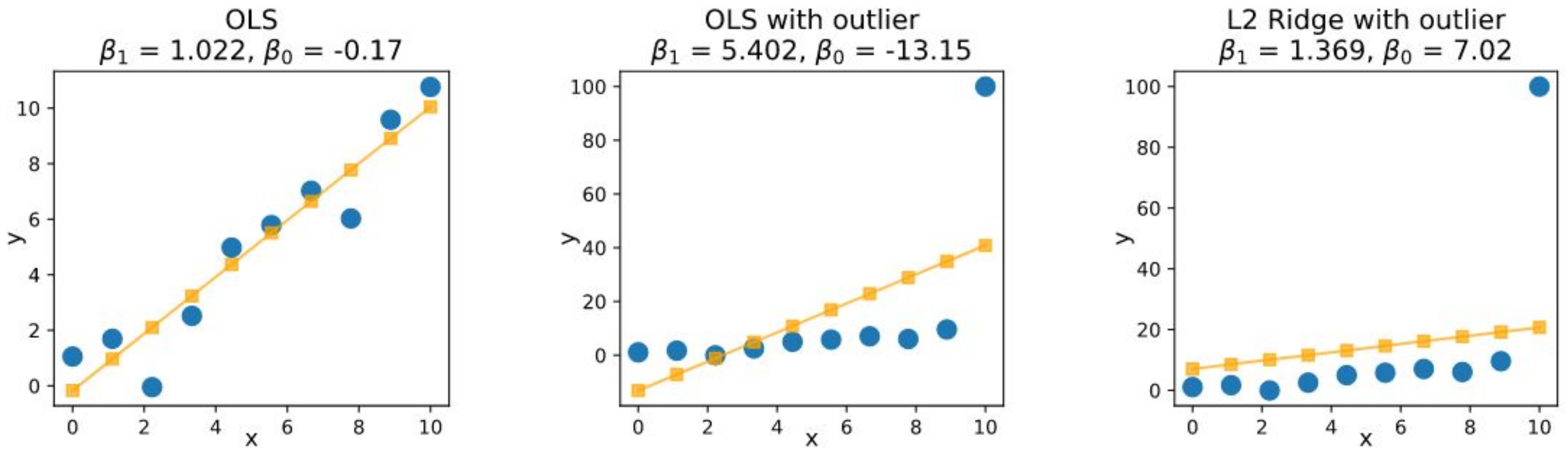


Regularización



.UBAfiuba FACULTAD DE INGENIERÍA

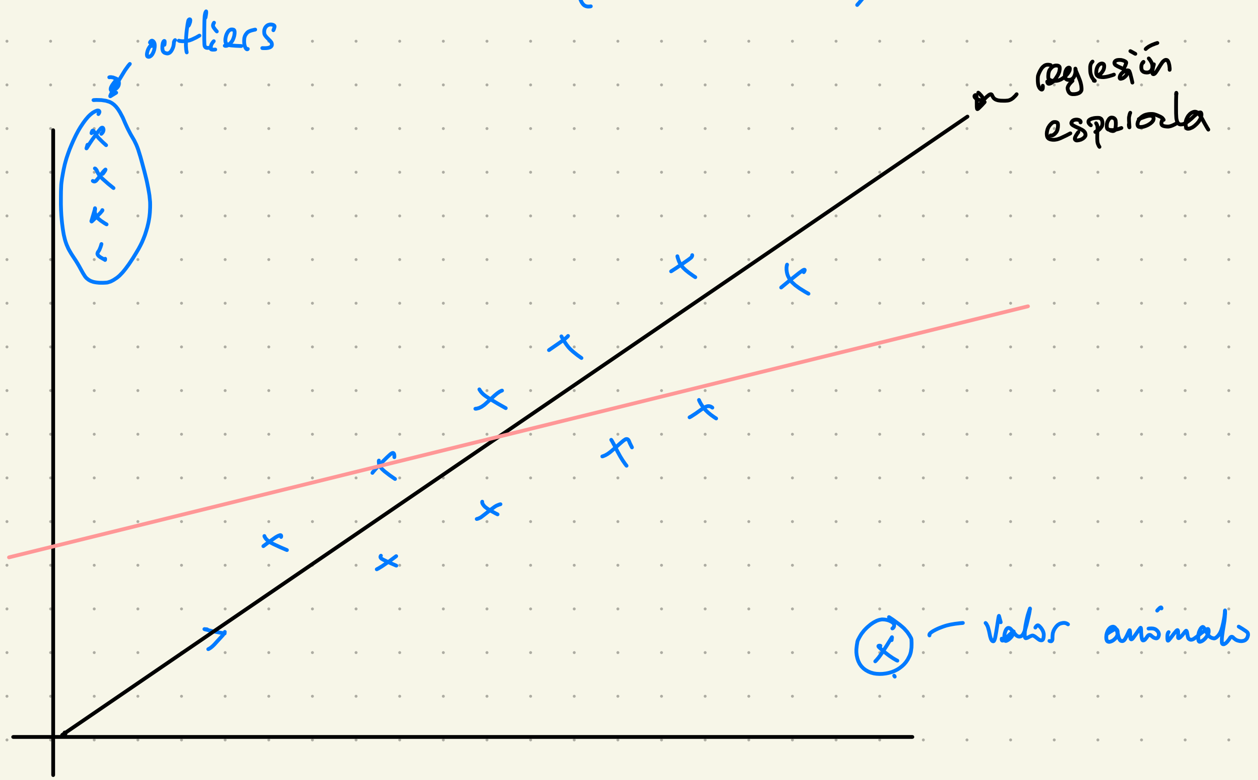
Regularización - Motivación



Riesgo Empírico

$$R(f) = \mathbb{E}\big(\underbrace{L(f, \hat{f})}_{\text{función de pérdida (loss function)}}\big) \rightsquigarrow L(f, \hat{f}) = (f - \hat{f})^2$$

fn. de pérdida L2



¿Cómo podemos mejorar mi reg. lineal?

1. Cambiar L
   - $L_1 = |f - \hat{f}|$
   - $L_2 = (f - \hat{f})^2$
   - $L_{\mathbb{I}} = \mathbb{I}_{|f - \hat{f}| \leq c}$
   - L_{huber}
   - L_{Tukey}

L_{huber}, L_{Tukey}: Estimadores robustos de la pérdida (multiparamétricos)

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

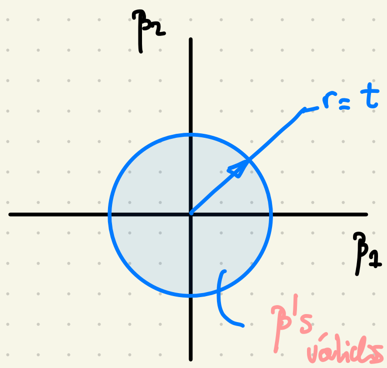
nosotros siempre busamos minimizar R de la forma más sencillo, a veces cambiar L no es barato o peor no es útil → probamos regularizar

2. Regularización.

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

partimos de

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



Espacio de parametros

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

parámetro de complejidad

↑ Termino de regularización (Weigth decay)

Regularización

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

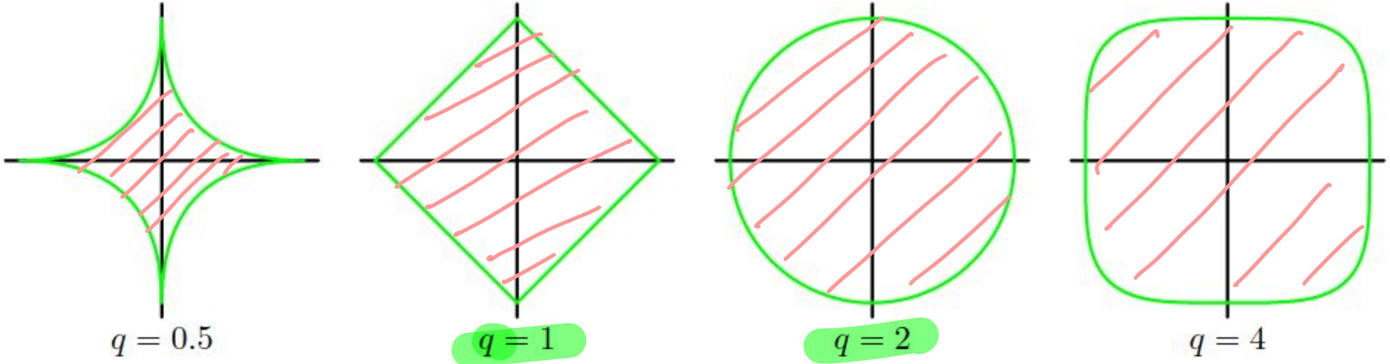
Observado - Predicción

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



Lasso

Ridge

▨ valores validos de $\|w\|_q$

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

Regularización

Maximum A Posteriori como regularización

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

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

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

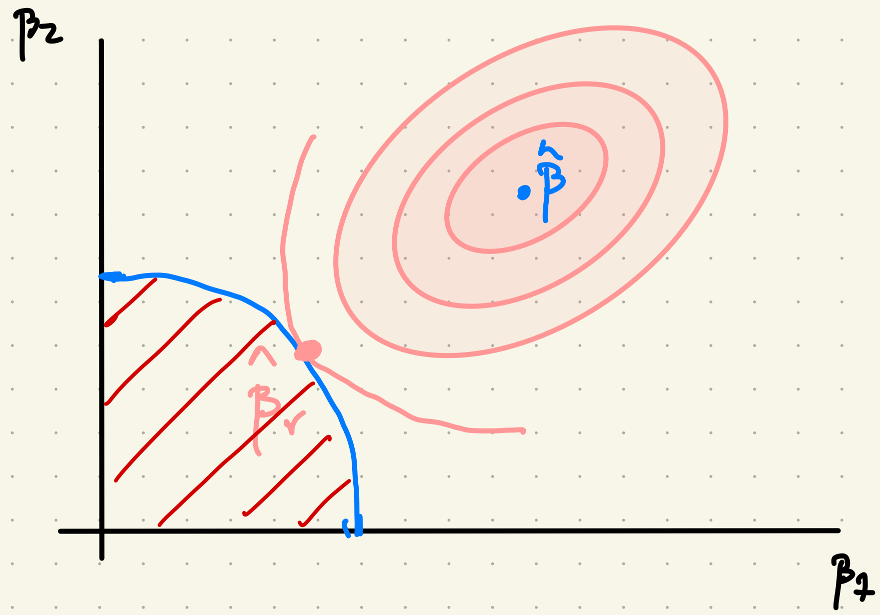
$$p(w|\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}|\mathcal{X}, w)p(w)}{p(\mathcal{Y}|\mathcal{X})}$$

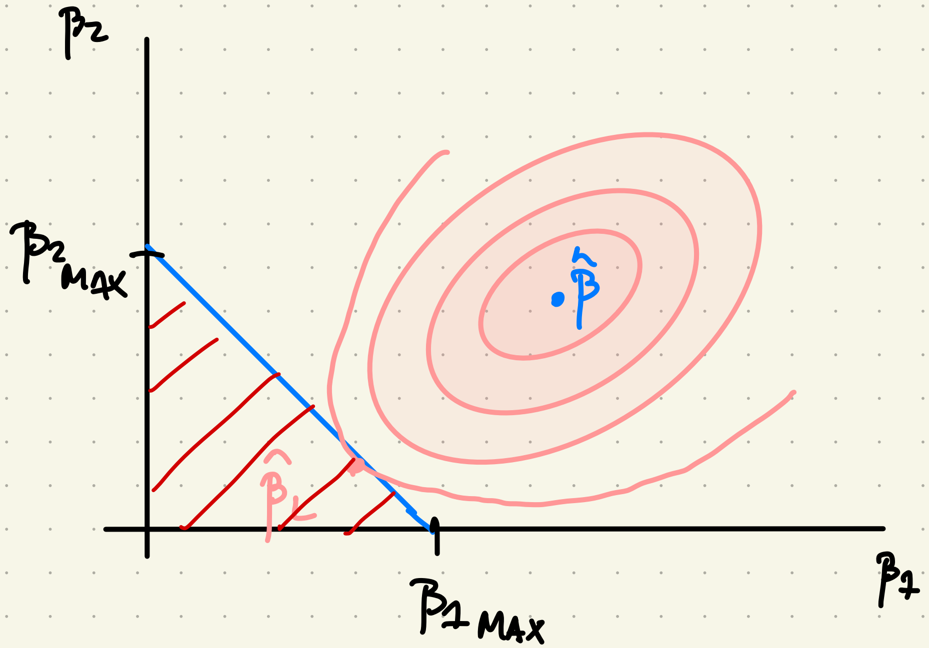
$$w_{map} = (\Phi^T \Phi + \frac{\sigma^2}{b^2} I)^{-1} \Phi^T y$$

Gaussian prior con varianza b2

Representación gráfica:

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





β's válidos

Ridge

porque enrobustece mi sist.

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

Lasso

porque hace un proceso de selección de variables

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

Pero, regularizar viene a costo de alejarnos del óptimo.

¿Cómo funciona?

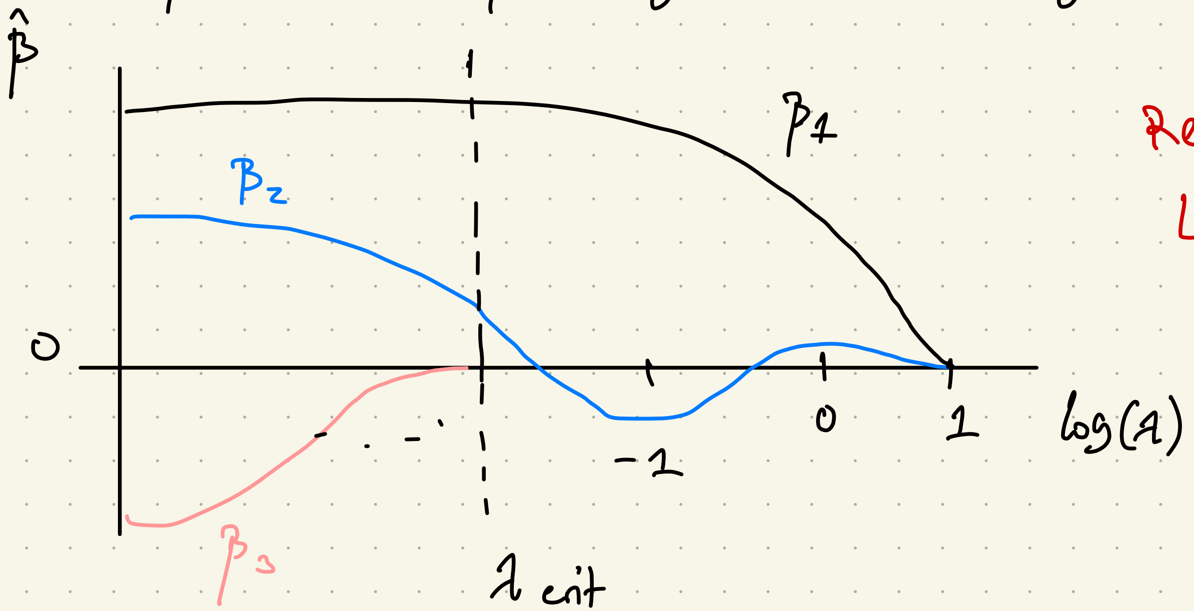
1. Elegir q (Lasso: 1, Ridge: 2)

2. Voy a elegir un vector de λ's "apropiado" ($\lambda \uparrow$ más $\uparrow$ penalidad)

3. optimizo para λ_i más $RSS(\lambda; q) = \| y - X\beta \|^q + \lambda \|\beta\|_q$

4. Calcular las métricas (Error de representación, bondad, algún criterio externo).

5. Comparamos y elegimos el mejor:



Regresión Lasso típica

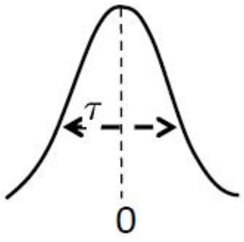
Regularización

Maximum A Posteriori como regularización - Ridge (L2)

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

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

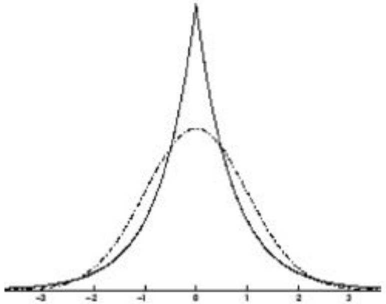
$$\widehat{\beta}_{\text{MAP}} = (\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{A}^\top \boldsymbol{Y}$$

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

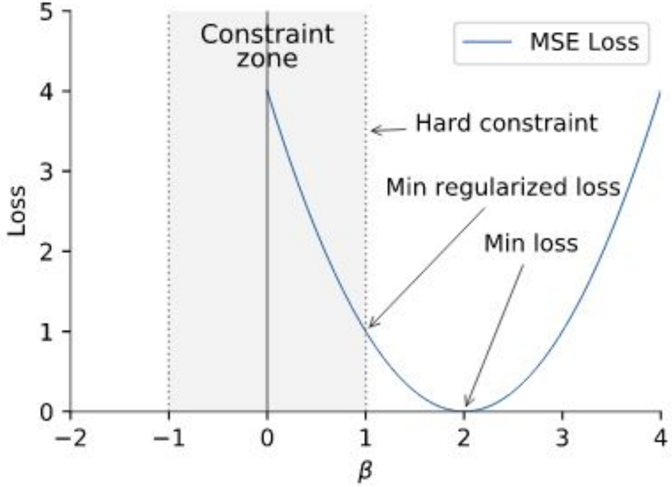
$$\beta_i \overset{iid}{\sim} \text{Laplace}(0, t) \qquad\qquad p(\beta_i) \propto e^{-|\beta_i|/t}$$

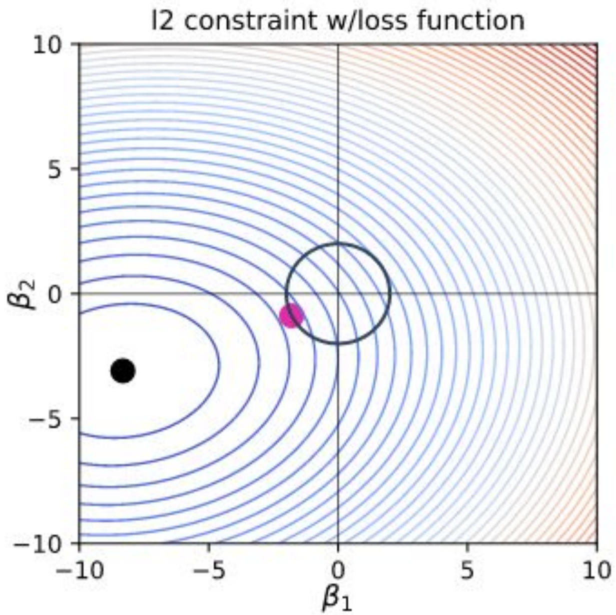


$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \underbrace{\lambda}_{\text{constant}(\sigma^2, t)} \|\beta\|_1 \qquad \text{Lasso}$$

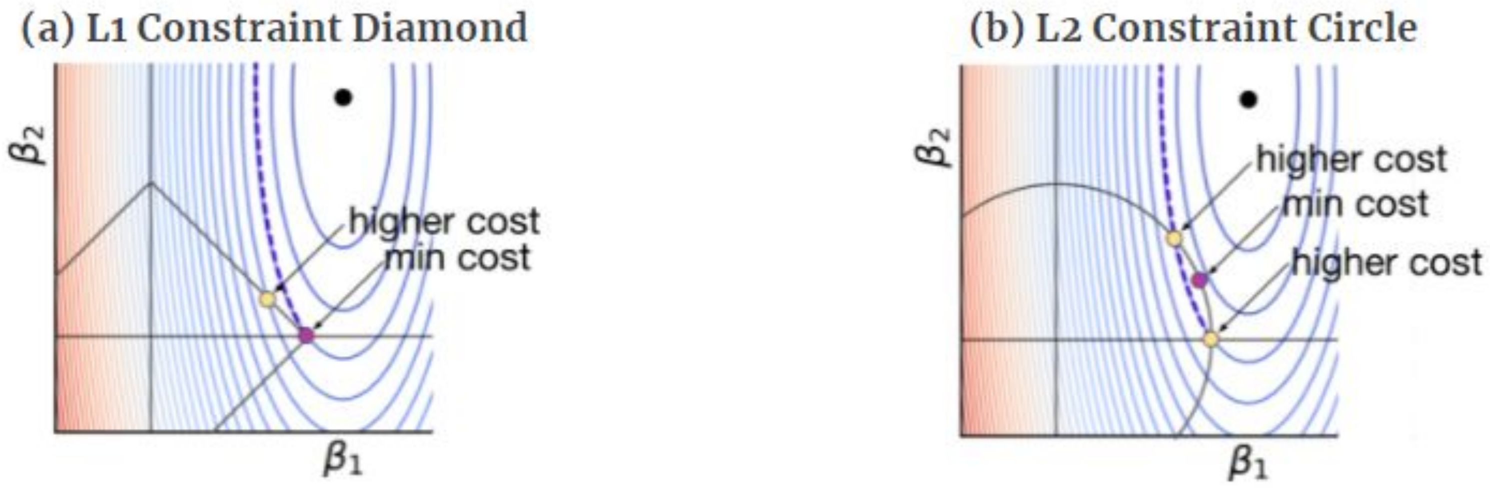
Machine Learning 10-315, Carnegie Mellon

Regularización





Regularización

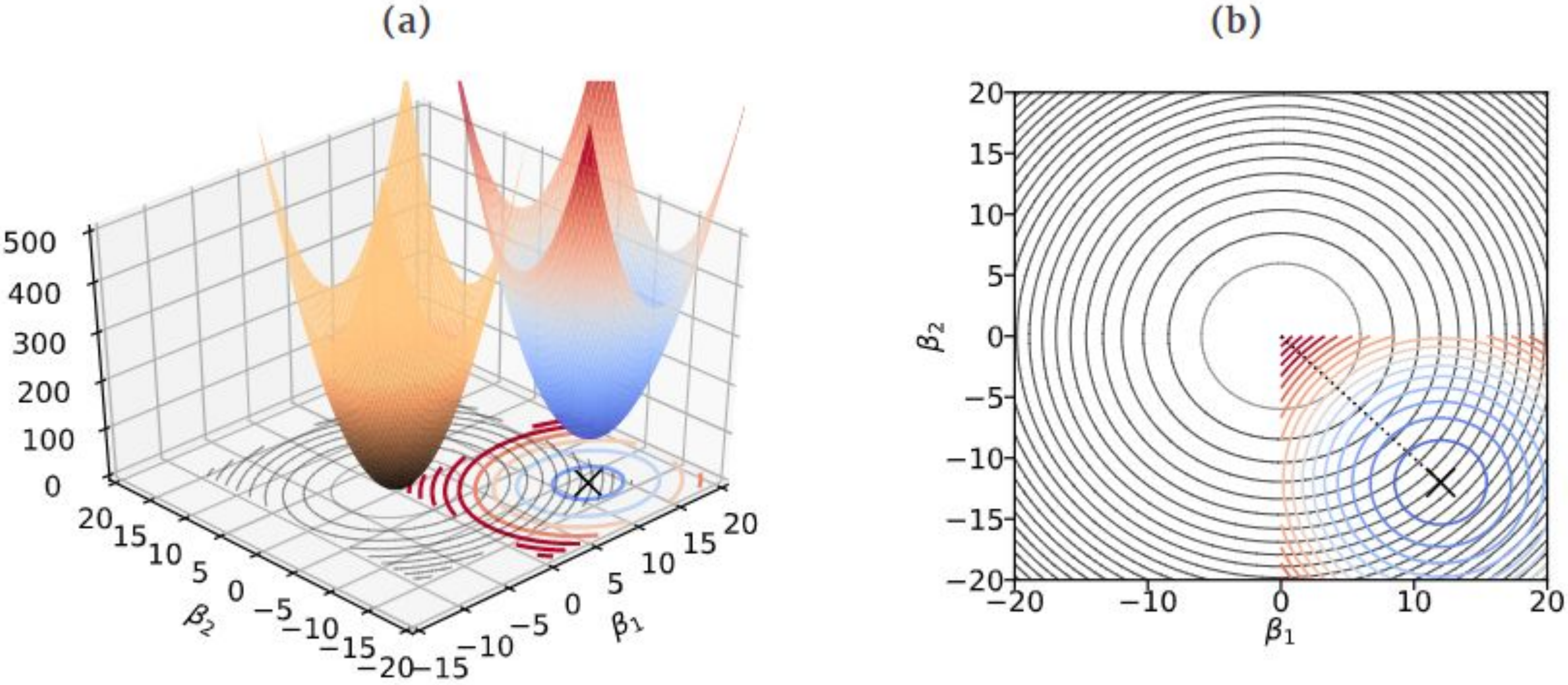


ElasticNet

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

¿Qué β se reduce más?

Regularización



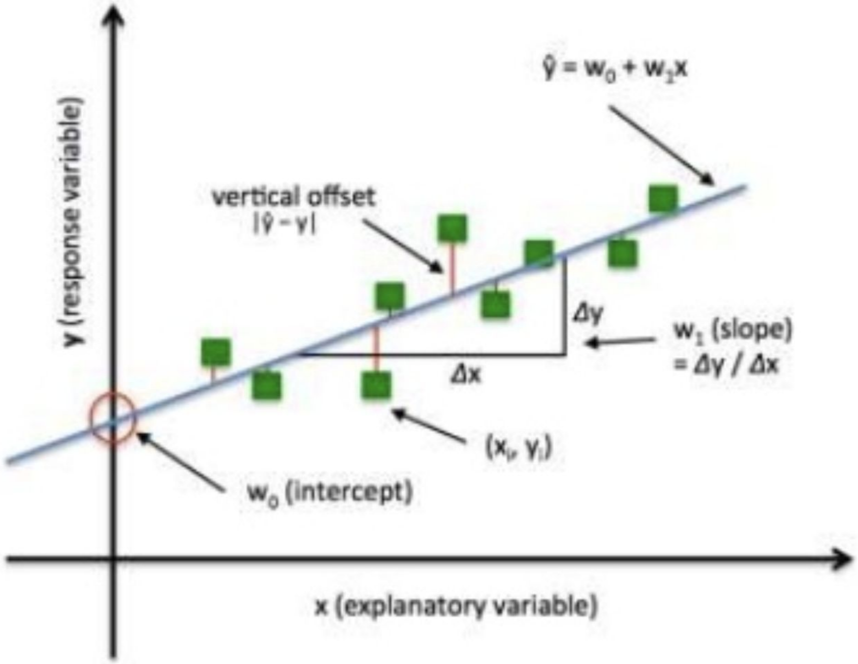
Gradiente Descendente

Implementación de Gradiente Descendente

Solucion analitica

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

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

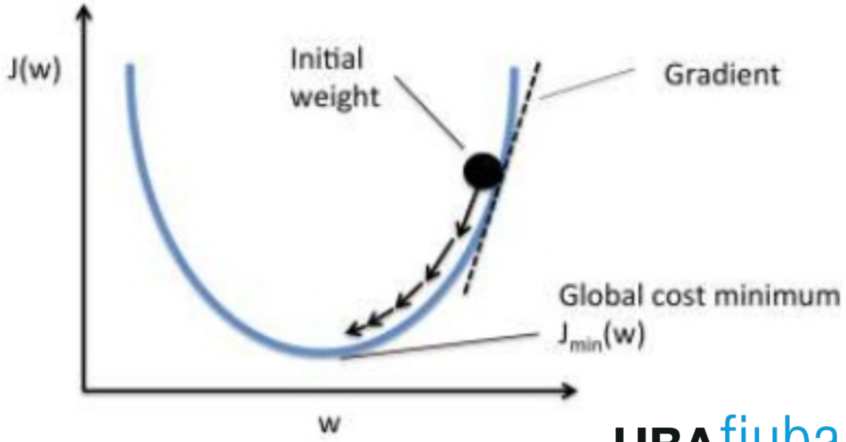


Implementación de Gradiente Descendente

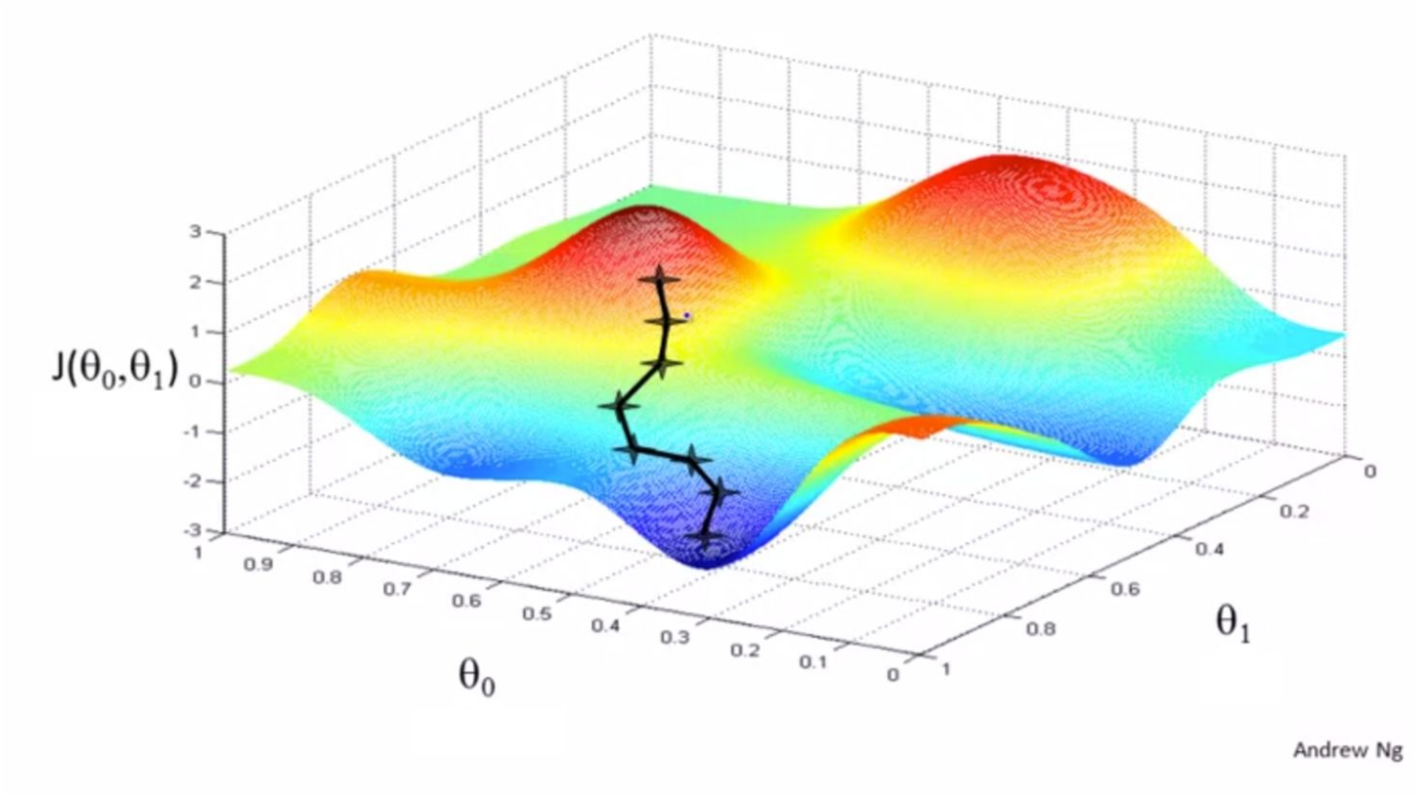
Solución numérica

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

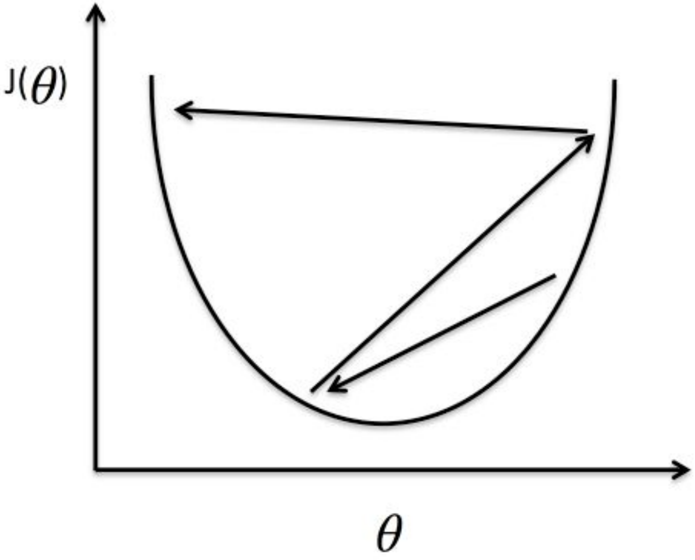
$$W \longleftarrow W - \alpha \nabla \left(\sum_i (y_i - X_i \cdot W)^2 \right)$$



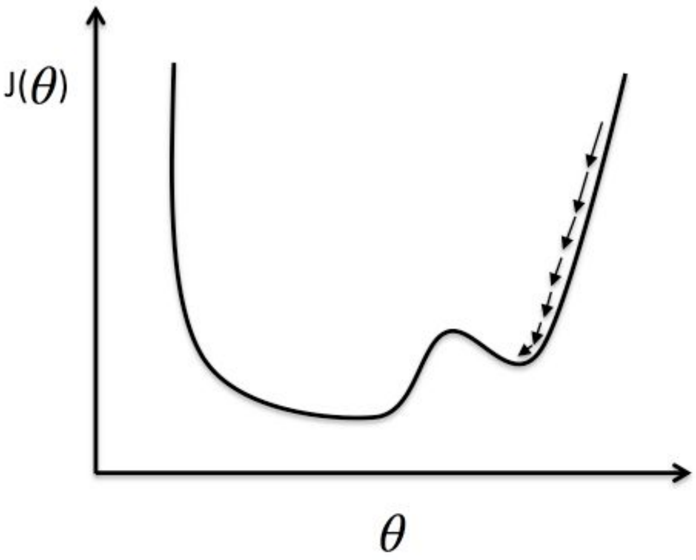
Gradiente Descendente



Gradiente Descendente



Large learning rate: Overshooting.



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

Implementación de Gradiente Descendente

Solución numérica

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

Gradient Descent

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:

- 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

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

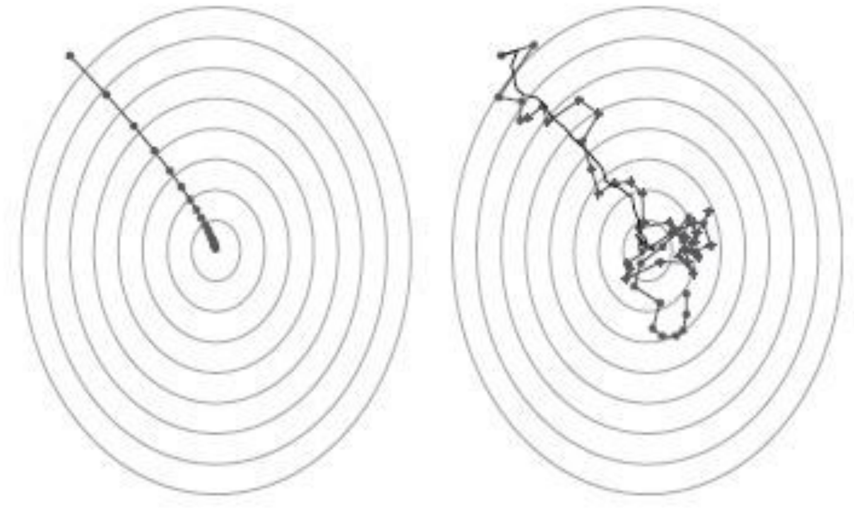
Stochastic Gradient Descent algorithm

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

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

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

Gradient Descent

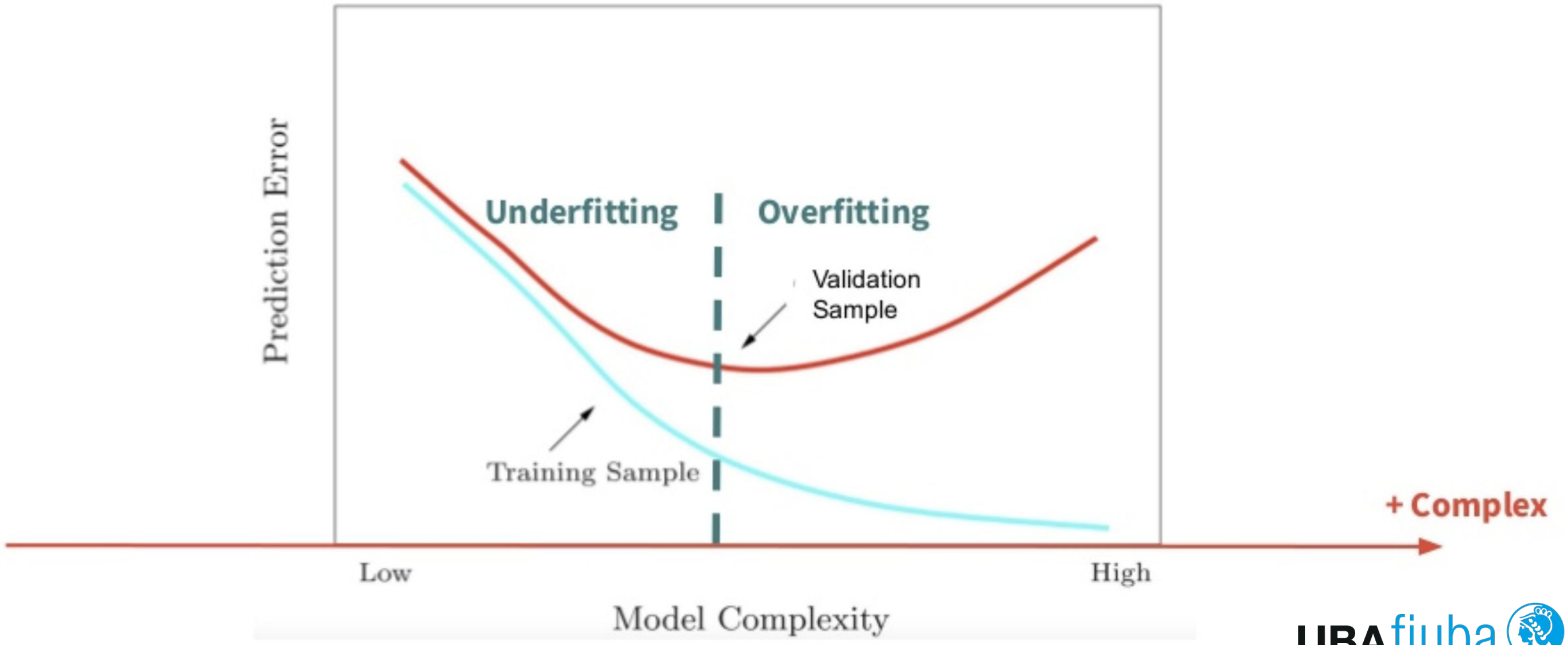
\sum_i \Big(-2 (y_i - \hat{y}_i) x_{ij}
\Big)

Comparativa de gradientes

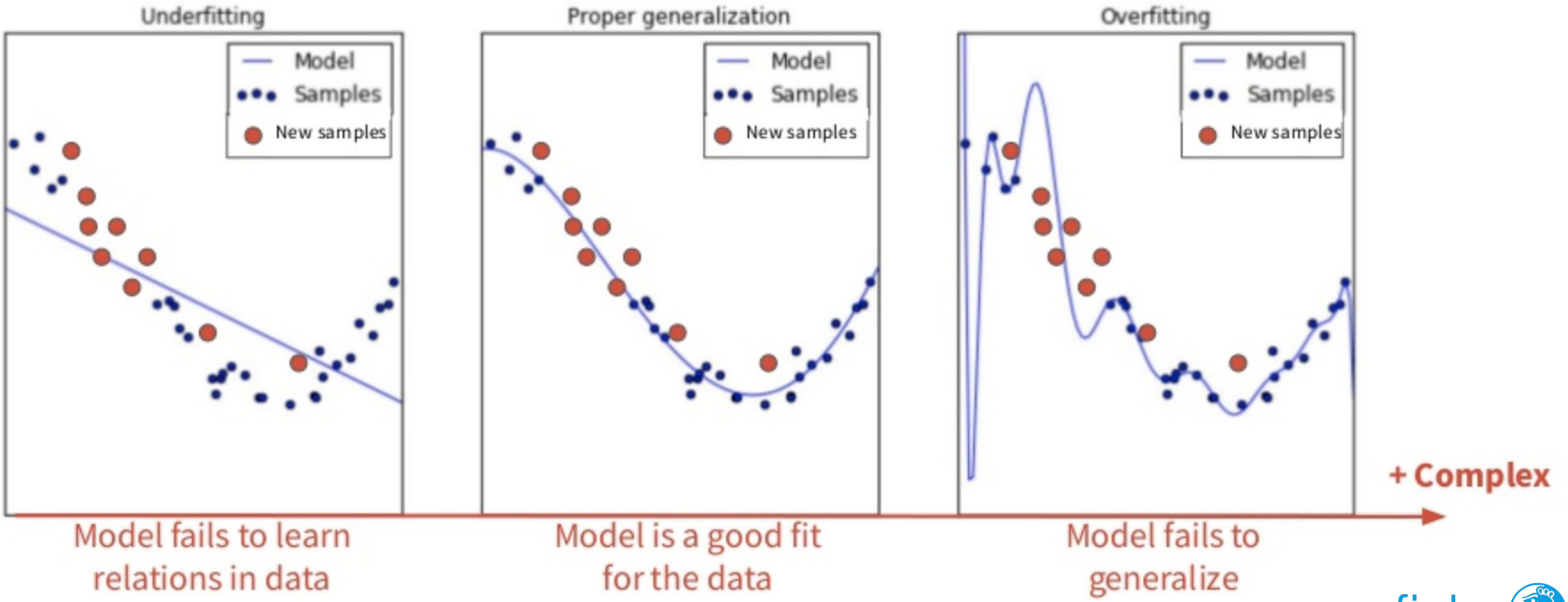
	Gradient Descent	Stochastic Gradient Descent	Mini-Batch Gradient Descent
Gradient	$\nabla\left(\sum_{\text{all samples}}(y_i - f_W(X_i))^2\right)$	$\nabla\left((y_i - f_W(X_i))^2\right)$	$\nabla\left(\sum_{\text{batch samples}}(y_i - f_W(X_i))^2\right)$
Speed	Very Fast (vectorized)	Slow (compute sample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

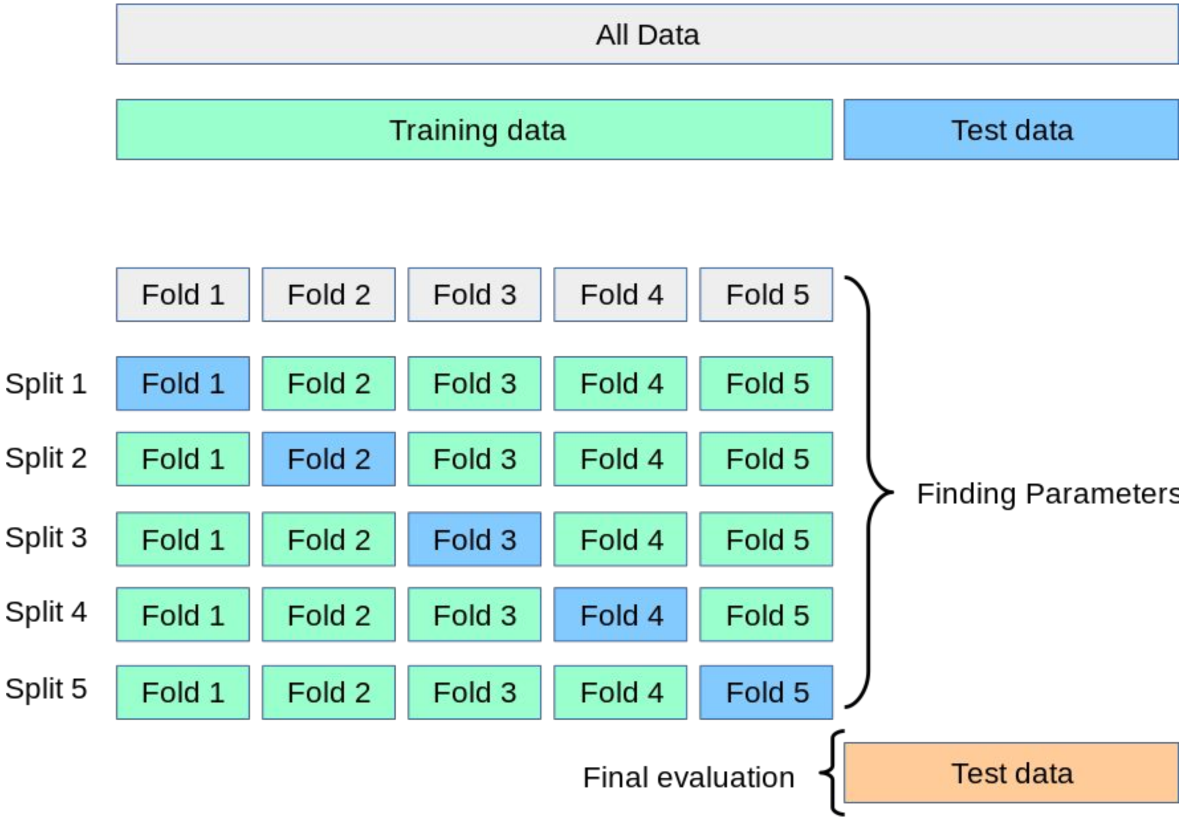
Selección de modelos



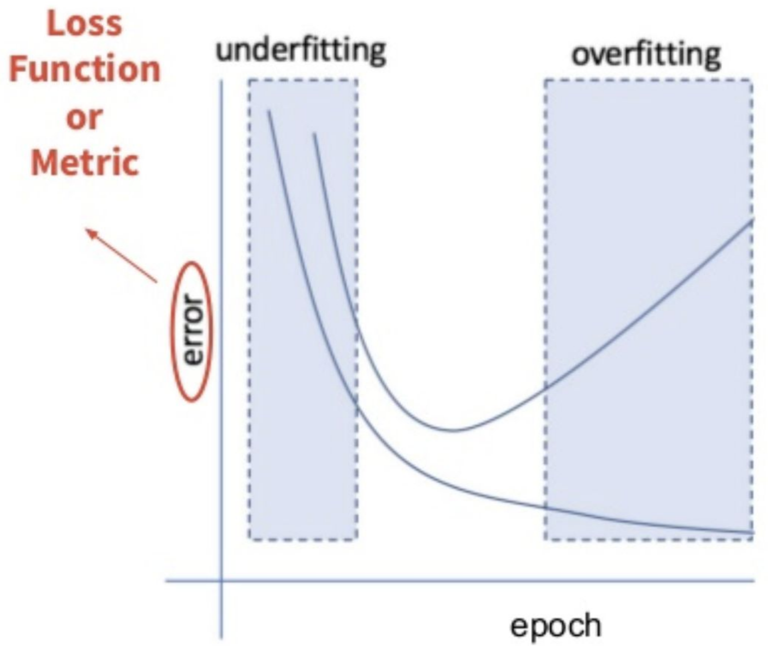
Selección de modelos



Cross-Validation



Entrenamiento numérico del modelo seleccionado - Obtención de parámetros

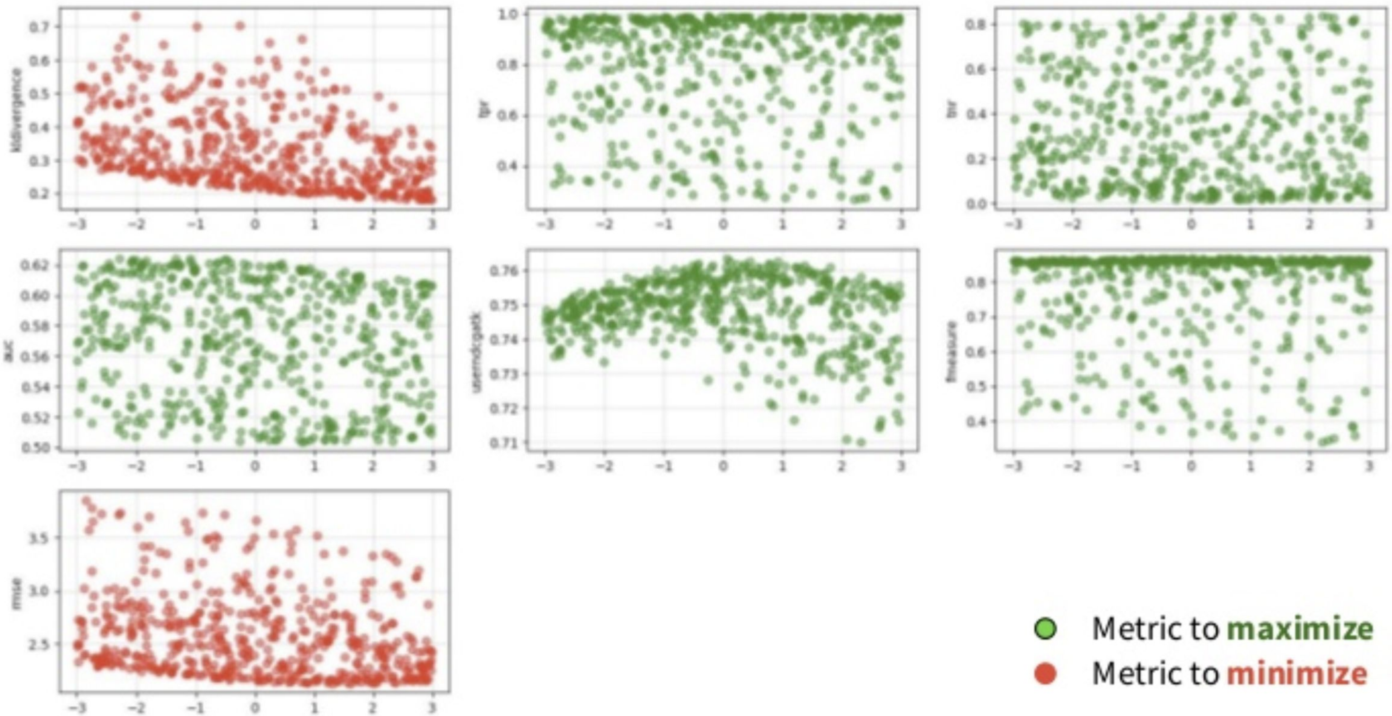


Mini-Batch Gradient Descent

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

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