Introducción a la Inteligencia Artificial
Facultad de Ingeniería
Universidad de Buenos Aires

Índice

Clase 5

1. Regularización en Regresión Lineal
    a. Ridge Regression
    b. Lasso Regression
2. Gradiente Descendente
    a. Implementación de Gradiente Descendente
    b. Gradiente Descendente Estocástico
    c. Gradiente Descendente Mini-Batch
3. Entrenamiento de modelos
    a. Selección de modelos
    b. Cross-Validation
    c. Hyperparameters
4. Ejercicio integrador

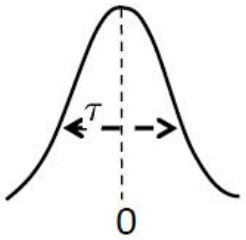
(1) REGULARIZACIÓN EN EL PIZARRÓN

Regularización

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

I) Gaussian Prior

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



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

Ridge Regression

λ → constant(σ^2, τ^2)

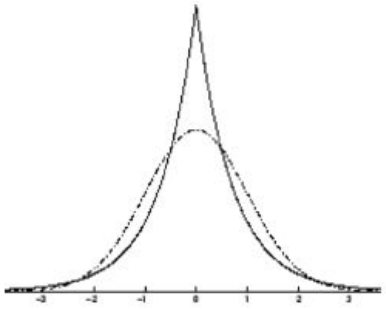
$$\widehat{\beta}_{\text{MAP}} = (\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})^{-1} \boldsymbol{A}^\top \boldsymbol{Y}$$

Regularización

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

II) Laplace Prior

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



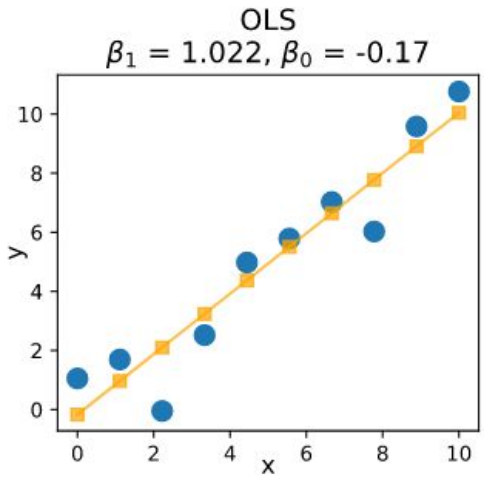
$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_1$$

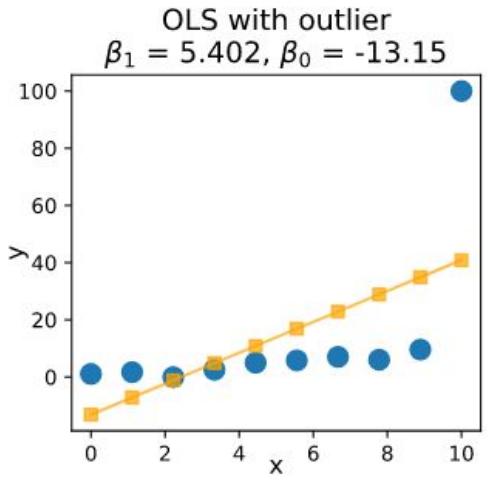
Lasso

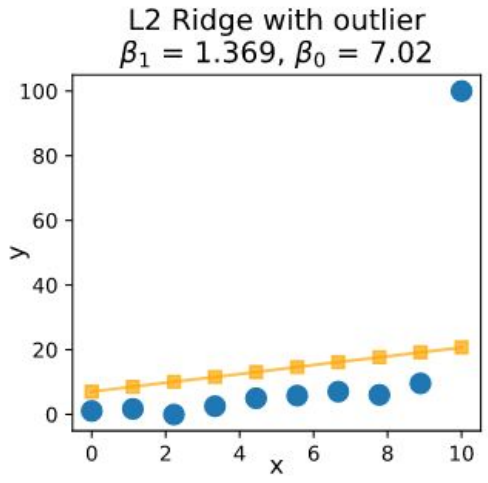
$\downarrow$

$\text{constant}(\sigma^2, t)$

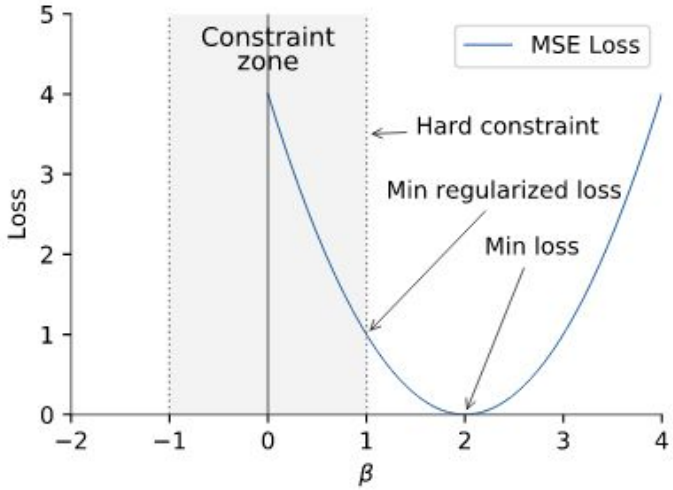
Regularización - Motivación

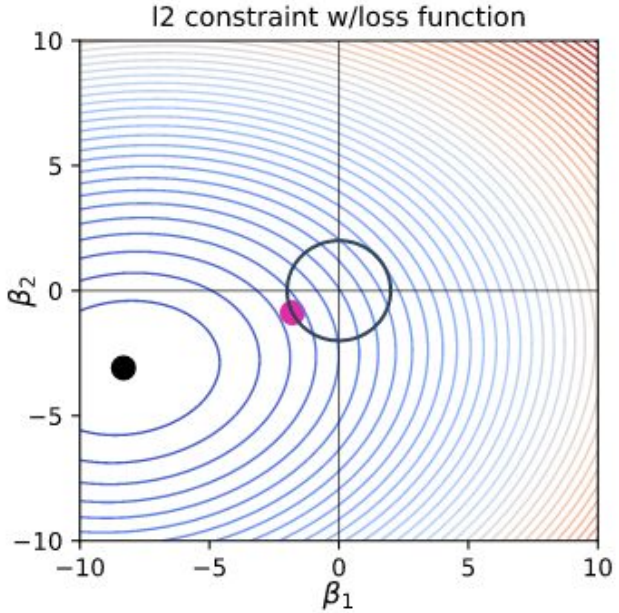




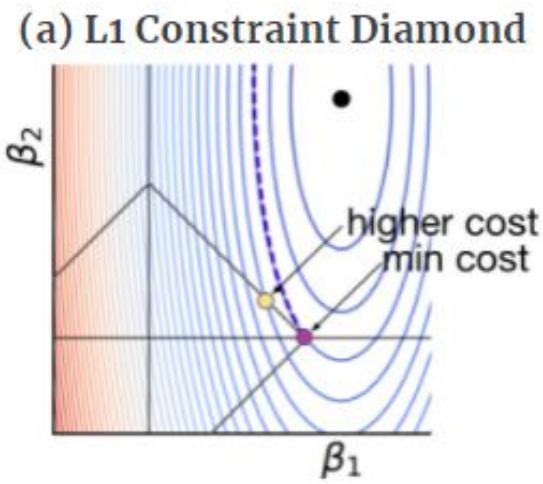


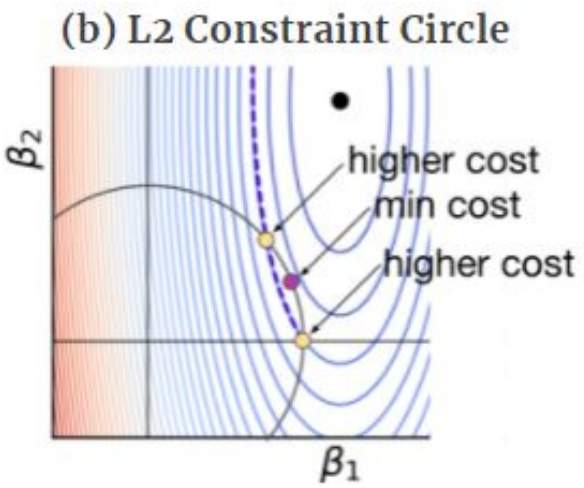
Regularización





Regularización

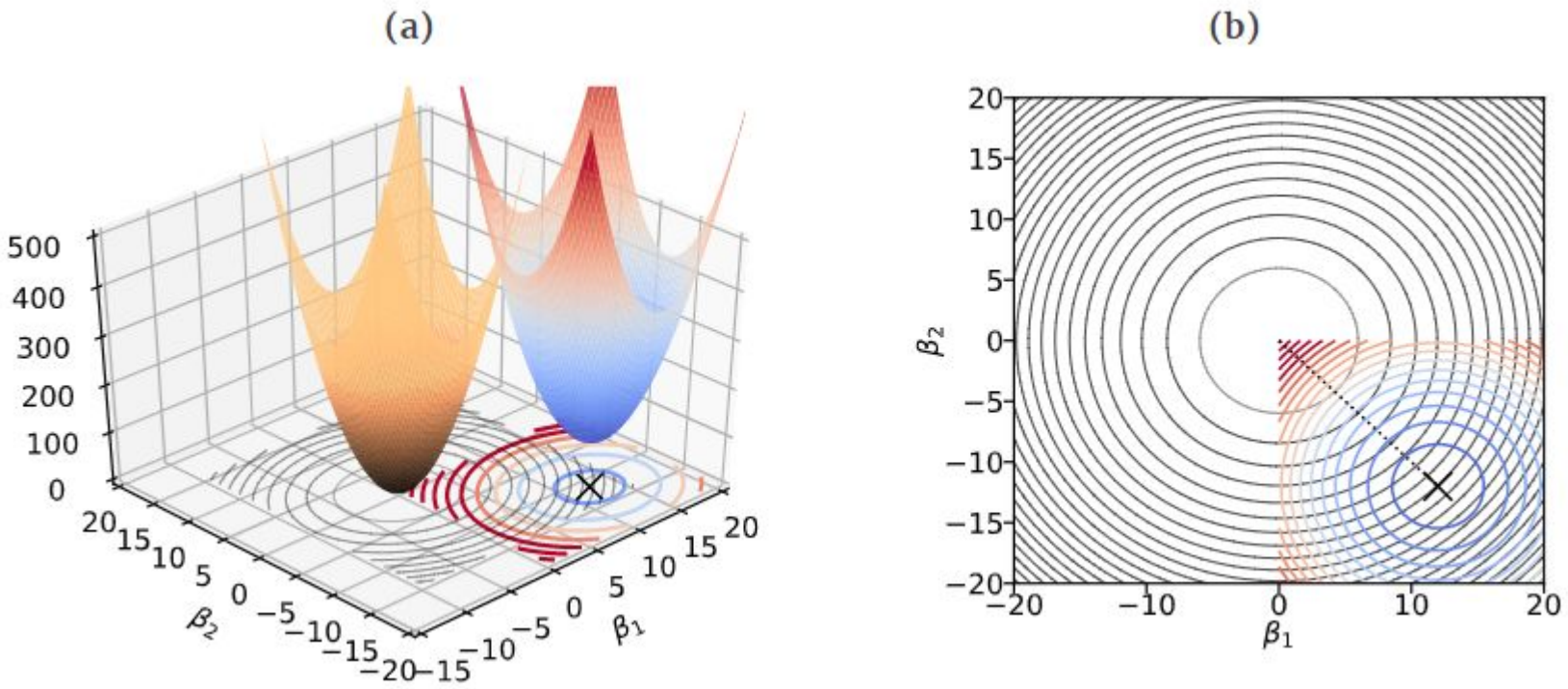




ElasticNet

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

Regularización



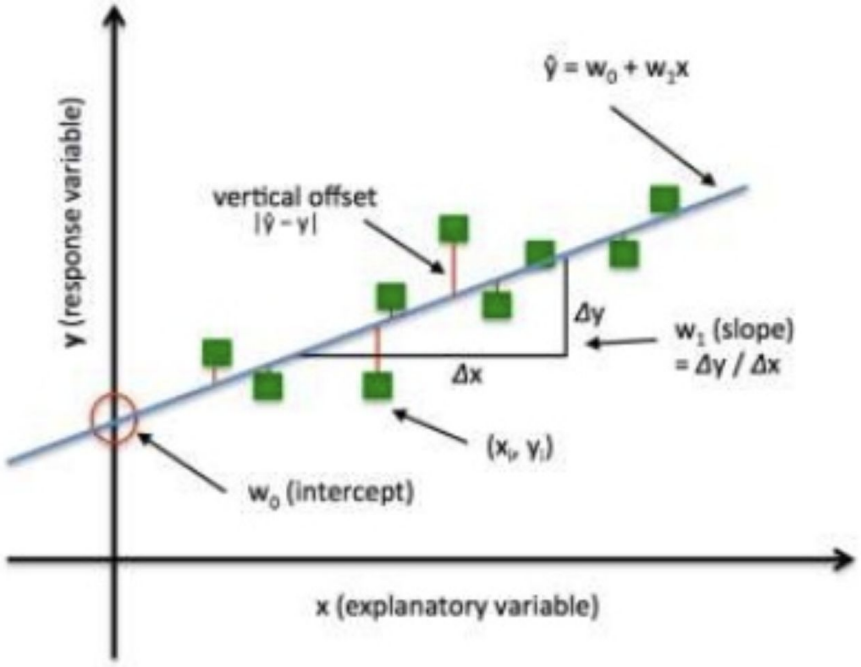
(1) RESUMEN DE RESULTADOS EN PIZARRÓN

Implementación de Gradiente Descendente

Solucion analitica

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

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

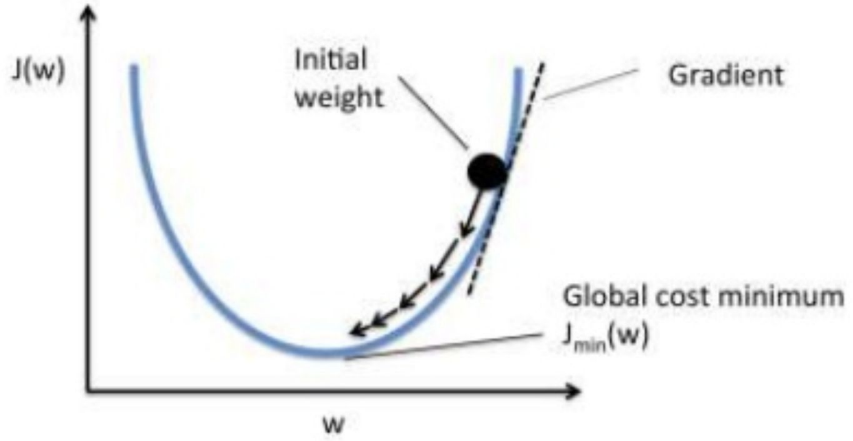


Implementación de Gradiente Descendente

Solución numérica

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

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



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

(1) OBTENER GRADIENTE DESCENDENTE EN PIZARRÓN

(2) IMPLEMENTAR EN NUMPY

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:

- 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

(1) OBTENER GRADIENTE DESCENDENTE ESTOCÁSTICO EN PIZARRÓN

(2) IMPLEMENTAR EN NUMPY

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

(1) OBTENER GRADIENTE DESCENDENTE MINI-BATCH EN PIZARRÓN

(2) IMPLEMENTAR EN NUMPY

Regularización - Cálculo Ridge

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

$$\frac{\partial}{\partial w_j} Cost\ (W) = -2 \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k\, x_{ik} \right\} + 2\lambda w_j$$

$$w_j^{t+1} = w_j^t - \eta \left[-2 \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k * x_{ik} \right\} + 2\lambda w_j \right]$$

$$w_j^{t+1} = (1 - 2\lambda\eta) w_j^t + 2\eta \sum_{i=1}^{N} x_{ij} \left\{ y_i - \sum_{k=0}^{M} w_k * x_{ik} \right\}$$

Regularización - Cálculo LASSO

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ absolute\ value\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} |w_j|$$

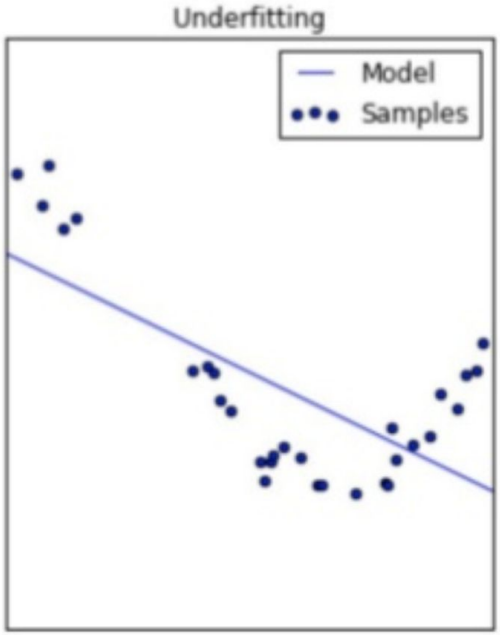
$$w_j = \begin{cases} g(w_{-j}) + \frac{\lambda}{2}, & if\ g(w_{-j}) < -\frac{\lambda}{2} \\ 0, & if\ -\frac{\lambda}{2} \le g(w_{-j}) \le \frac{\lambda}{2} \\ g(w_{-j}) - \frac{\lambda}{2}, & if\ g(w_{-j}) > \frac{\lambda}{2} \end{cases}$$

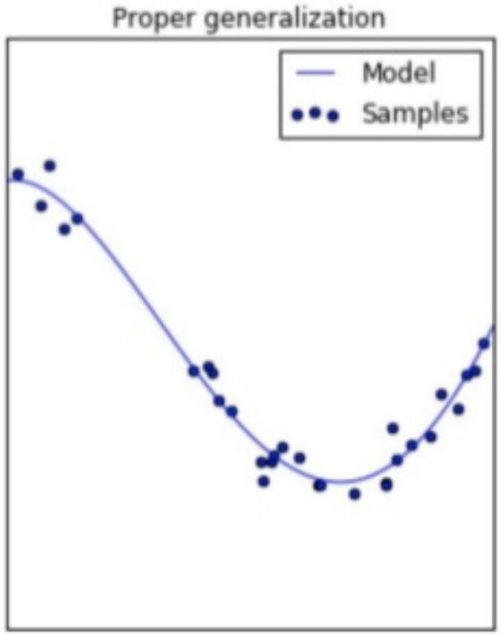
Comparativa de gradientes

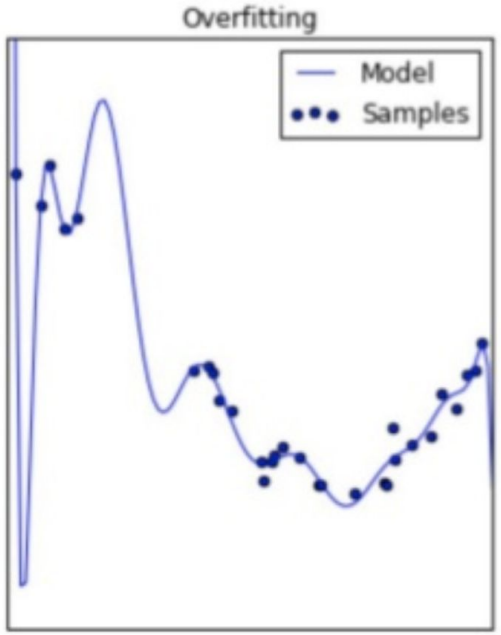
	Gradient Descent	Stochastic Gradient Descent	Mini-Batch Gradient Descent
Gradient	$\nabla \left(\sum_{\text{all samples}} (y_i - f_W(X_i))^2 \right)$	$\nabla \left((y_i - f_W(X_i))^2 \right)$	$\nabla \left(\sum_{\text{batch samples}} (y_i - f_W(X_i))^2 \right)$
Speed	Very Fast (vectorized)	Slow (compute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

Entrenamiento de modelos - Cross-Validation

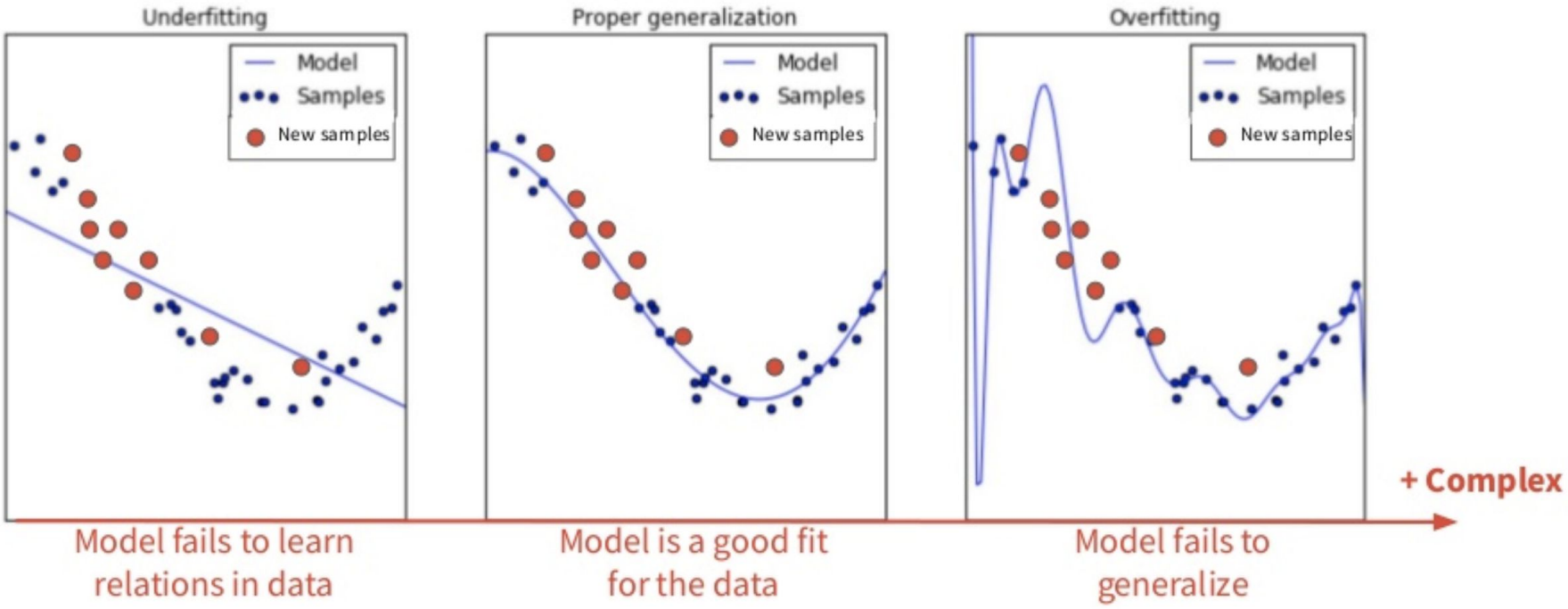
Selección de modelos



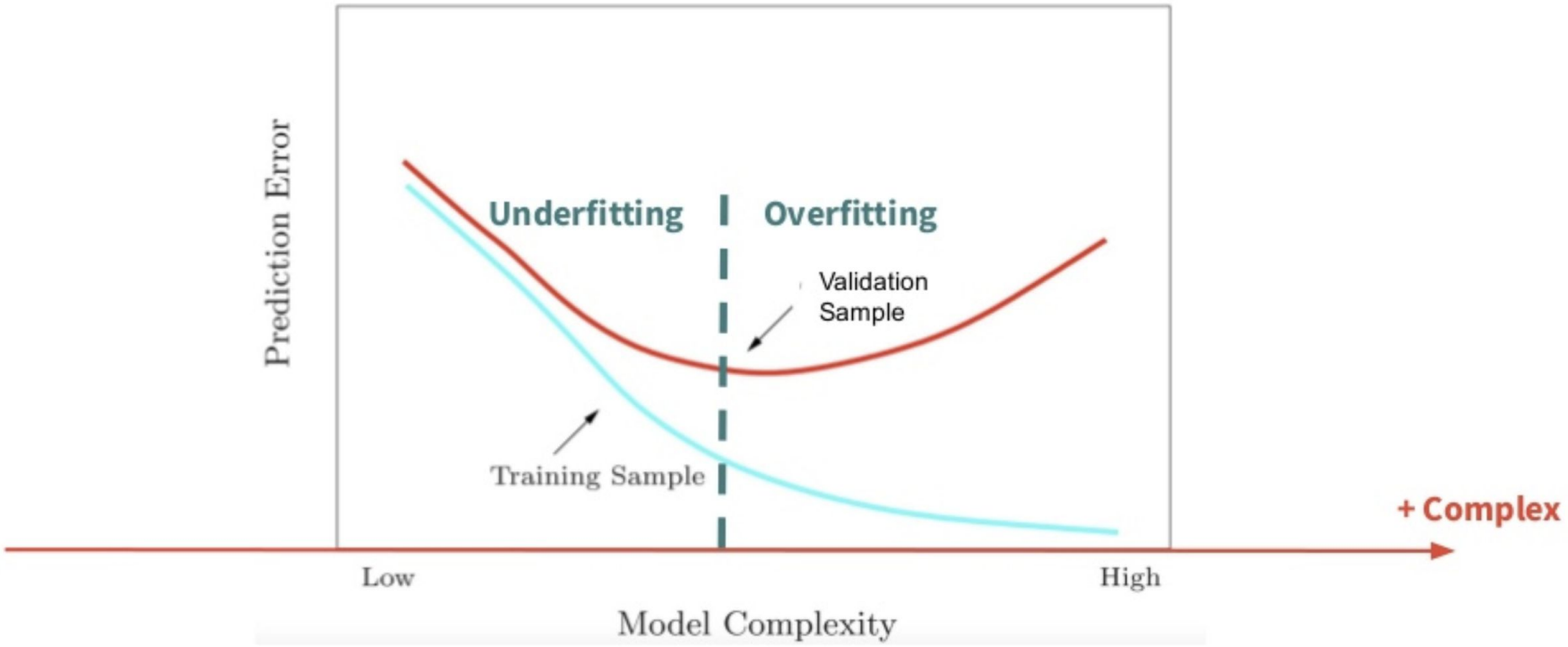




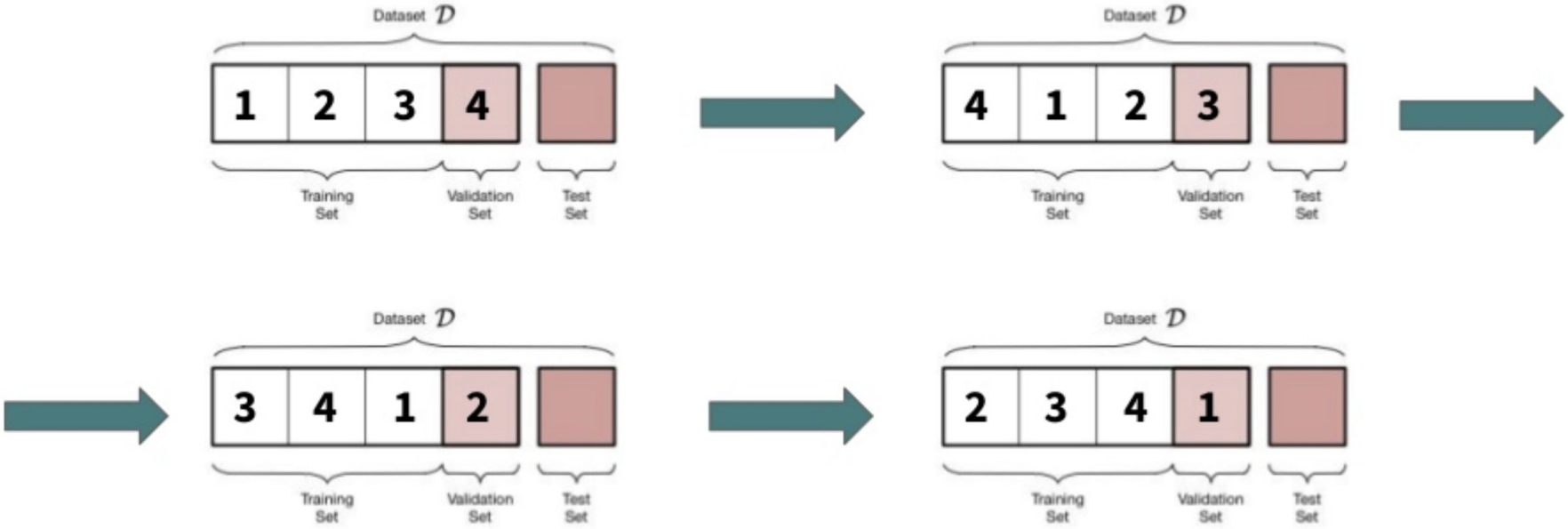
Selección de modelos



Selección de modelos

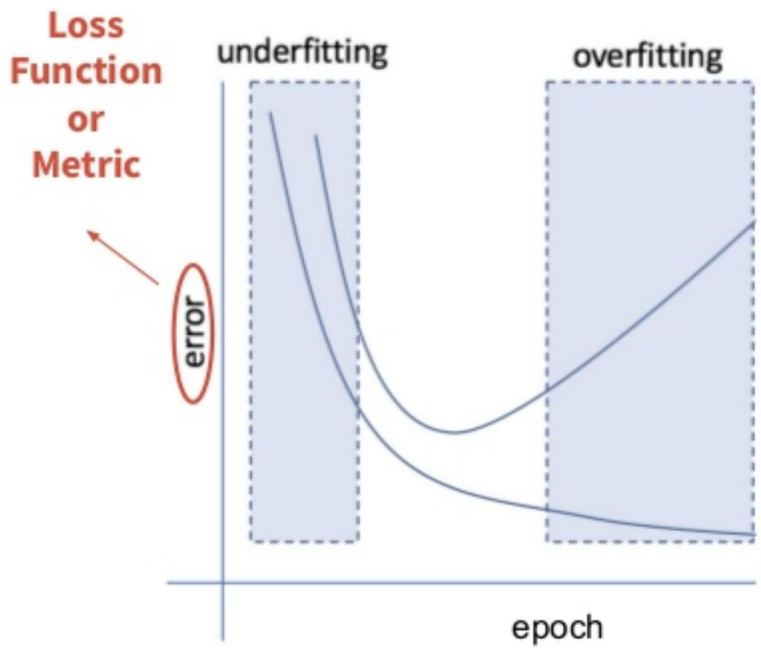


Cross-Validation



(1) IMPLEMENTAR K-FOLDS EN NUMPY

Entrenamiento numérico del modelo seleccionado - Obtención de parámetros

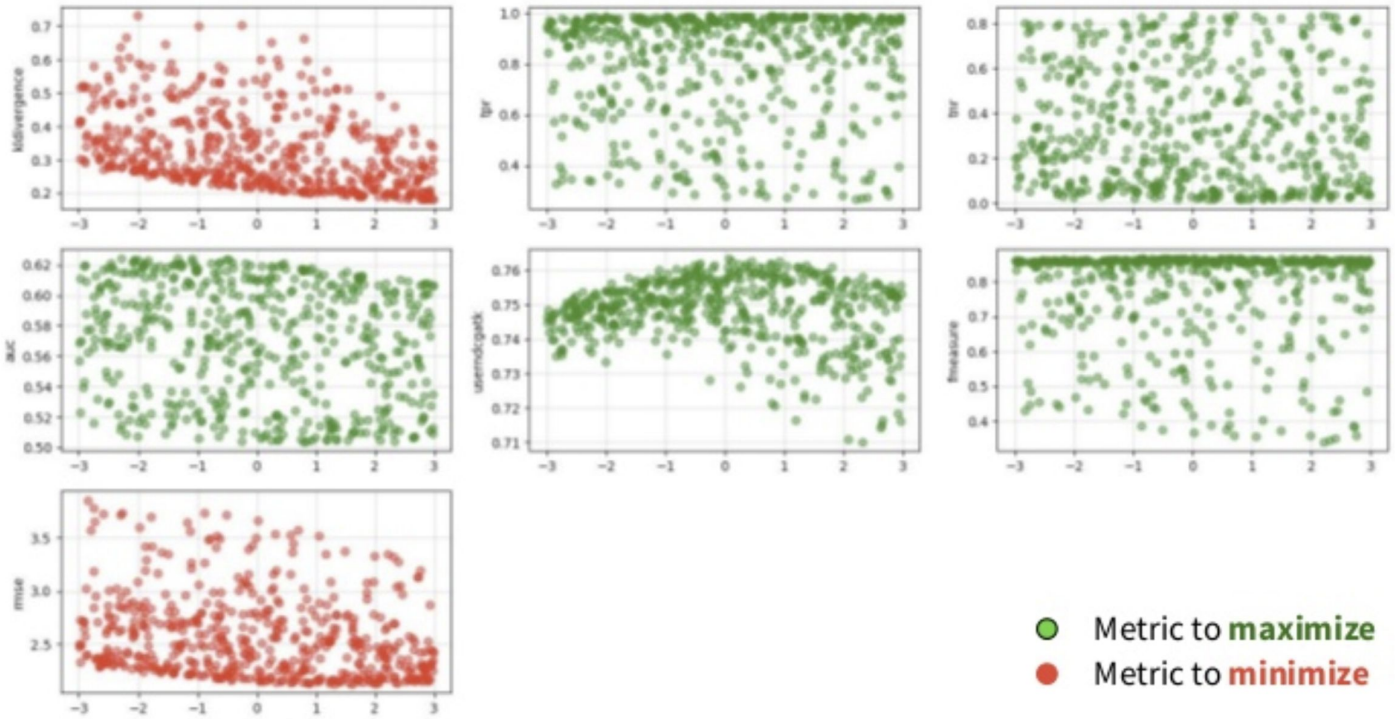


Mini-Batch Gradient Descent

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

Gaussian Process

Ejercicio integrador

1. Generar un dataset sintético con una función seno (pueden usar otras funciones no lineales)
2. Hacer el gráfico de los datos
3. Hacer fit con la solución cerrada de regresión lineal y con los algoritmos de descenso por el gradiente vistos en clase, para diferentes polinomios hasta el orden 15. Comenten los problemas numéricos encontrados.
4. Obtener mediante cross-validation para cada polinomio el error de validación.
5. Seleccionar el modelo con complejidad correcta para el dataset.
6. Regularizar el modelo y comparar los resultados.

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