UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Non-linear Regression

Cristian López Del Alamo
clopezd@utec.edu.pe
GINIA - Research group

2024

TRANSFORMATEC

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

1 Remembering : Linear Regression

TRANSFORMATEC

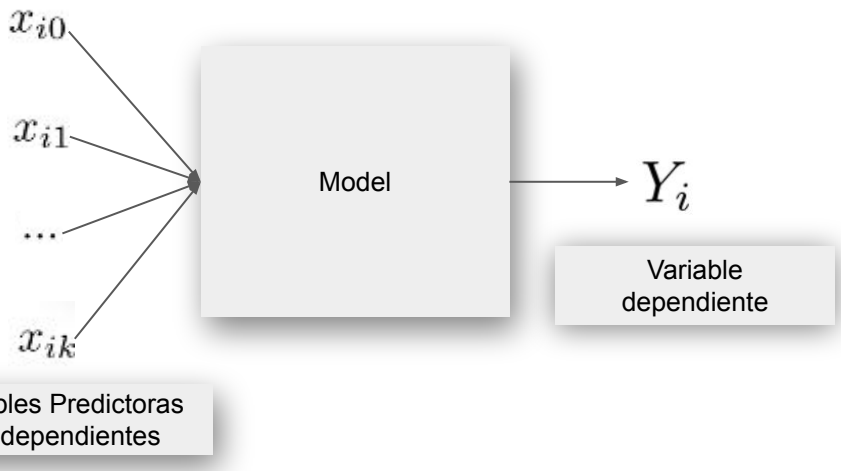
Cristian López Del Alamo

Objective: In this class, the goal is to understand the similarity between linear regression and non-linear regression, their applications, and the related machine learning algorithm.

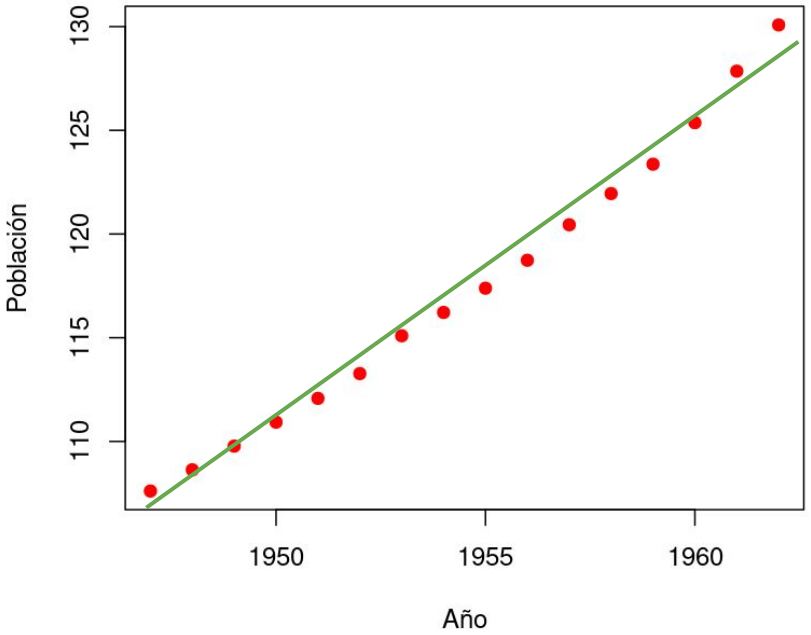
$$Y_i = b + x_{i0}w_0 + x_{i1}w_1 + ... + x_{ik}w_k$$

Variable dependiente

Variables Predictoras o Independientes



Simple Linear Regression



1. Hypothesis

$$h(x_i) = b + wx_i$$

2. Loss Function

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

3. Derivative Calculus

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-x_i)}{n}$$

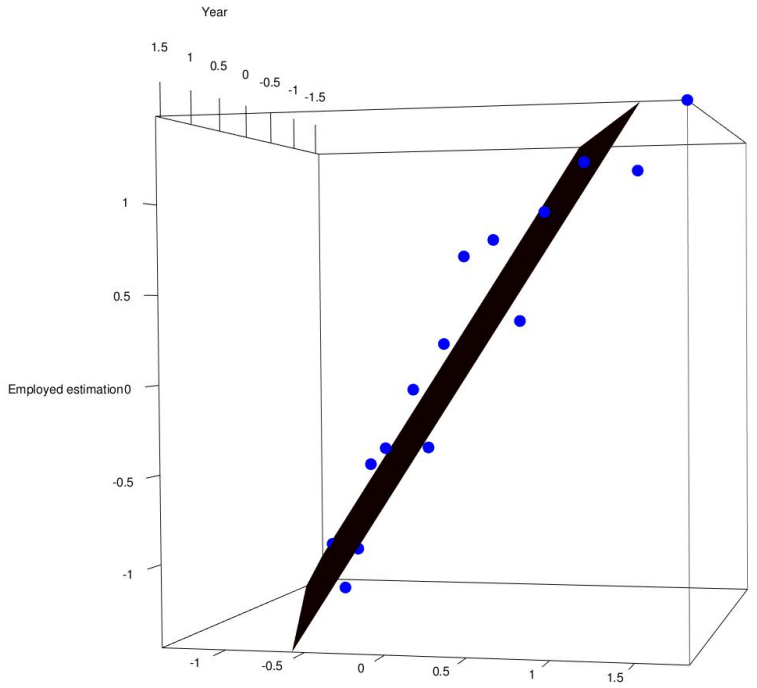
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-1)}{n}$$

4. Weights Update

$$w = w - \alpha\frac{\partial \mathcal{L}}{\partial w}$$

$$b = b - \alpha\frac{\partial \mathcal{L}}{\partial b}$$

Multiple Linear Regression: Case of 2 variables



1. Hypothesis

$$h(x_i) = b + x_{(i,1)}w_1 + x_{(i,2)}w_2$$

2. Loss Function

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

3. Derivative Calculus

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-x_{(i,j)})}{n}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))(-1)}{n}$$

4. Weights Update

$$w_j = w_j - \alpha\frac{\partial \mathcal{L}}{\partial w_j}$$

$$b = b - \alpha\frac{\partial \mathcal{L}}{\partial b}$$

Multiple Linear Regression: Case of k variables

Hypothesis

$$h(x_i) = b + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = 1w_0 + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = x_{(i,0)}w_0 + x_{(i,1)}w_1 + x_{(i,2)}w_2 + ... + x_{(i,k)}w_k$$

$$h(x_i) = [x_{(i,0)} \;\; x_{(i,1)} \;\; x_{(i,2)} \;\; ... \;\; x_{(i,k)}][w_0 \;\; w_1 \;\; w_2 \;\; ... \;\; w_k]^T$$

$$h(x_i) = x_i * w^T$$

Multiple Linear Regression: Case of k variables

Hypothesis

$$X = \begin{bmatrix} 1 & x_{01} & \cdots & x_{0p} \\ 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \times W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_p \end{bmatrix} = h(X) = \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(i) \\ \vdots \\ h(p) \end{bmatrix}$$

$$h(x_i) = [1, x_{i,1}, \ldots, x_{i,p}] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix}$$

$$h(X) = XW^t$$

Multiple Linear Regression: Case of k variables

Loss Function

$$\left[Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = h(X) = \begin{bmatrix} h(x_0) \\ h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} \right] \longrightarrow Loss = \begin{bmatrix} (y_0 - h(x_0))^2 \\ (y_1 - h(x_1))^2 \\ \vdots \\ (y_n - h(x_n))^2 \end{bmatrix} \longrightarrow \mathcal{L} = ||y - h(X)||_2^2$$

Normal L_2

Multiple Linear Regression: Case of k variables

Derivative Computation

$$(Y - h(X) = \begin{bmatrix} y_0 - h(x_0) \\ y_1 - h(x_1) \\ \vdots \\ y_i - h(x_i) \\ \vdots \\ y_n - h(x_n) \end{bmatrix}^T \otimes (-1) \quad X = \begin{bmatrix} 1 & x_{01} & \cdots & x_{0p} \\ 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{ip} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \Rightarrow \frac{\partial \mathcal{L}}{\partial w} = \frac{(Y - h(X))^T * (-1 * X)}{n}$$

Multiple Linear Regression: Case of k variables

Updating w

$$W = W - \alpha * \frac{\partial \mathcal{L}}{\partial w}$$

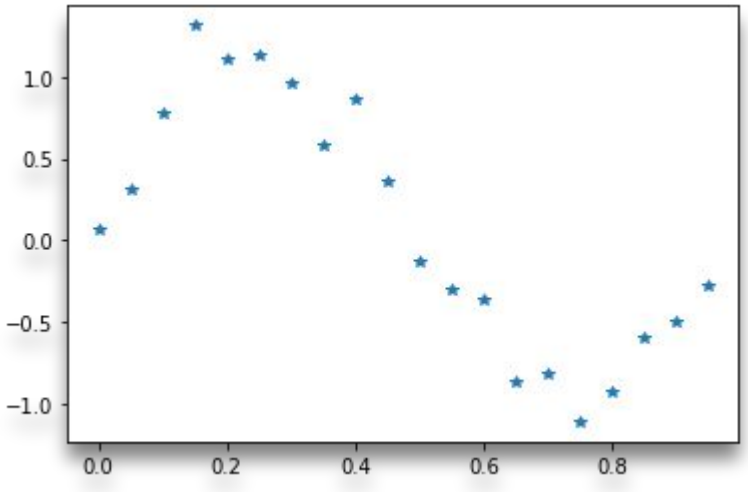
UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

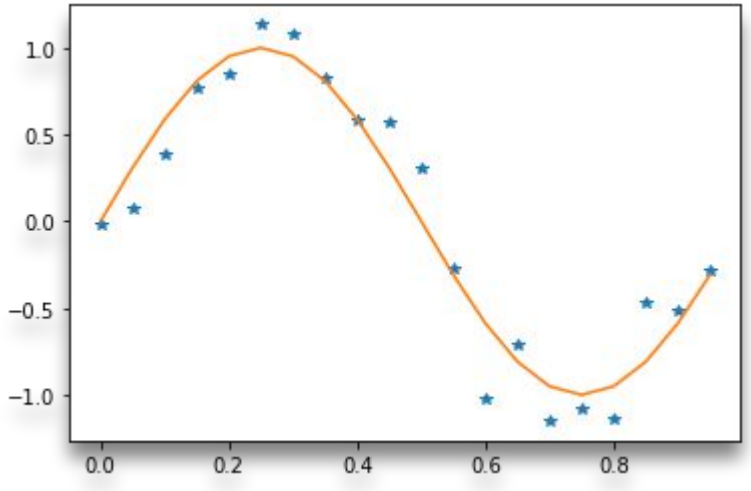
> Reinventa el mundo <

2 Nonlinear Regression

TRANSFORMATEC

Objective: Understand the mathematics behind multivariable nonlinear regression.


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8

Hypothesis: Polynomial

$$h(x_i) = b + x_i w_1 + x_i^2 w_2 + x_i^3 w_2 + ... + + x_i^p w_2$$

$$h(x_i) = b + \sum_{j=1}^{p} x_i^j w_j$$

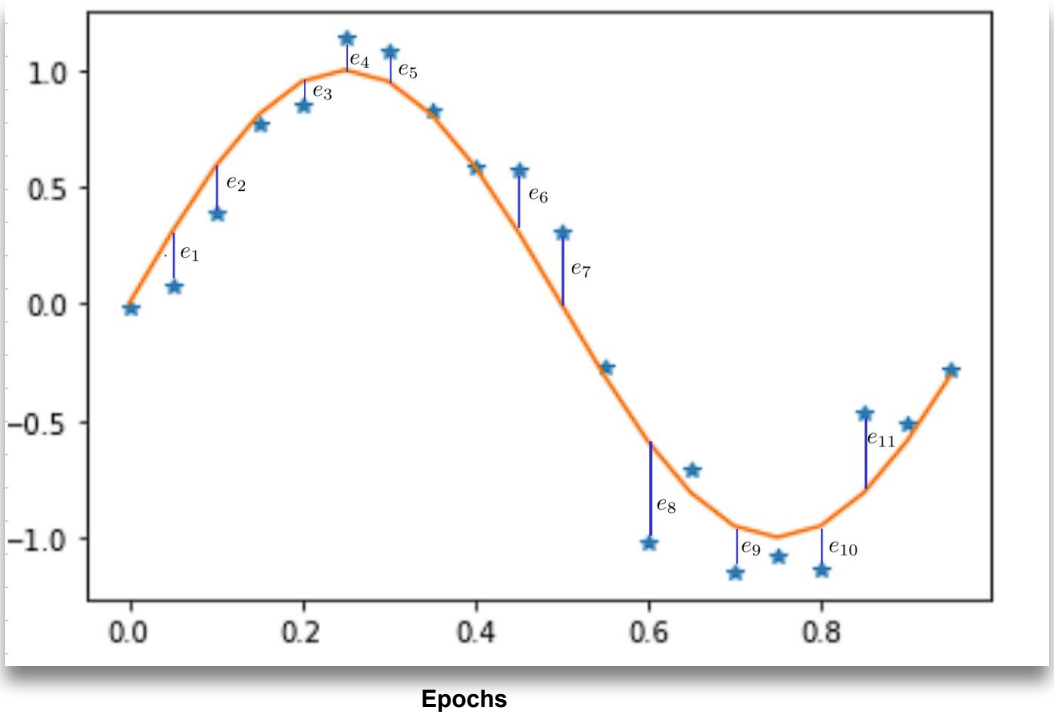
$$h(x_i) = x_i^0 w_0 + \sum_{j=1}^{p} x_i^j w_j$$

$$h(x_i) = \sum_{j=0}^{p} x_i^j w_j$$

$$h(X) = \begin{pmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_i) \\ \vdots \\ h(x_n) \end{pmatrix} = X = \begin{pmatrix} x_1^0 & x_1^1 & \cdots & x_1^j & \cdots & x_1^p \\ x_2^0 & x_2^1 & \cdots & x_2^j & \cdots & x_2^p \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots \\ x_i^0 & x_i^1 & \cdots & x_i^j & \cdots & x_i^p \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots \\ x_n^0 & x_n^1 & \cdots & x_n^j & \cdots & x_n^p \end{pmatrix} \times W^T = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_p \end{pmatrix}$$

$$h(X) = XW^T$$

Loss Function



Loss Function

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} \quad h(X) = \begin{pmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_i) \\ \vdots \\ h(x_n) \end{pmatrix} \quad Y - \hat{Y} = \begin{pmatrix} y_1 - h(x_1) \\ y_2 - h(x_2) \\ \vdots \\ y_i - h(x_i) \\ \vdots \\ y_n - h(x_n) \end{pmatrix} \quad E = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \\ \vdots \\ e_n \end{pmatrix}$$

$$E = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_i \\ \vdots \\ e_n \end{pmatrix} \times E^T = (e_1 \quad e_2 \quad \cdots \quad e_i \quad \cdots \quad e_n) = E\ E^T$$

$$L = \frac{EE^T}{2n}$$

$$\mathcal{L} = \frac{||Y - h(X)||_2^2}{2n}$$

L2 norm

```
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

```
1 def train(x, y, umbral, alfa, p):
2     np.random.seed(2001)
3     w = [np.random.rand() for i in range(1,p+1)]
4     y_pred = h(x,w)
5     L = Error(y, y_pred)
6     while (L > umbral):
7         dw = derivada(x, y, w)
8         w = update(w,db, alfa)
9         y = h(x,w)
10        L = Error(y, y_pred)
11    return w,L
```

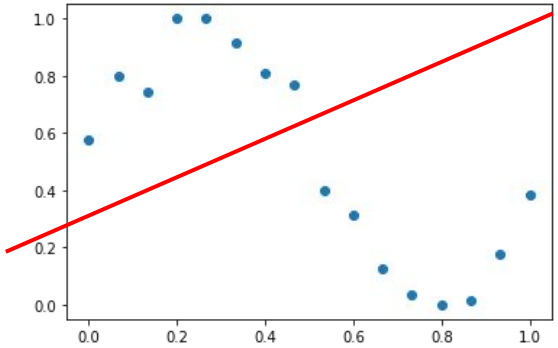
```python
def train(x, y, umbral, alfa, p):
    np.random.seed(2001)
    w = [np.random.rand() for i in range(1,p+1)]
    y_pred = h(x,w)
    L = Error(y, y_pred)
    while (L > umbral):
        dw = derivada(x, y, w)
        w = update(w,db, alfa)
        y = h(x,w)
        L = Error(y, y_pred)
    return w,L
```

```
1  def train(x, y, umbral, alfa, p):
2      np.random.seed(2001)
3      w = [np.random.rand() for i in range(1,p+1)]
4      y_pred = h(x,w)
5      L = Error(y, y_pred)
6      while (L > umbral):
7          dw = derivada(x, y, w)
8          w = update(w,db, alfa)
9          y = h(x,w)
10         L = Error(y, y_pred)
11     return w,L
```

```
1 def train(x, y, umbral, alfa, p):
2     np.random.seed(2001)
3     w = [np.random.rand() for i in range(1,p+1)]
4     y_pred = h(x,w)
5     L = Error(y, y_pred)
6     while (L > umbral):
7         dw = derivada(x, y, w)
8         w = update(w,db, alfa)
9         y = h(x,w)
10        L = Error(y, y_pred)
11    return w,L
```

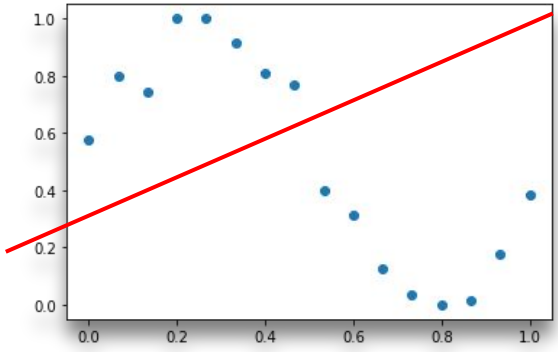
3 Regularization

Underfitting



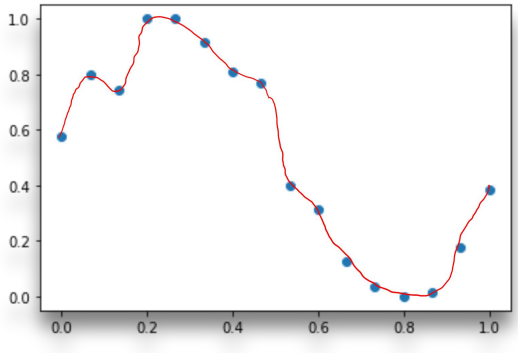
$$h(x_i) = x_i w + b$$

Underfitting



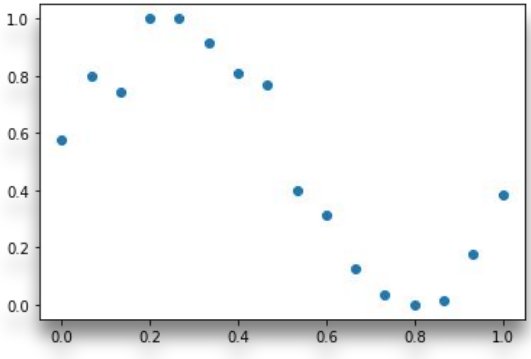
$$h(x_i) = x_i w + b$$

Overfitting



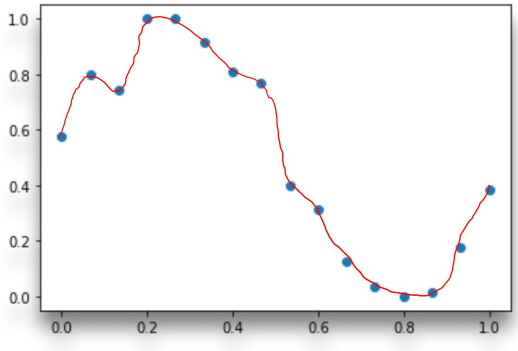
$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^{20} w_{20}$$

Underfitting



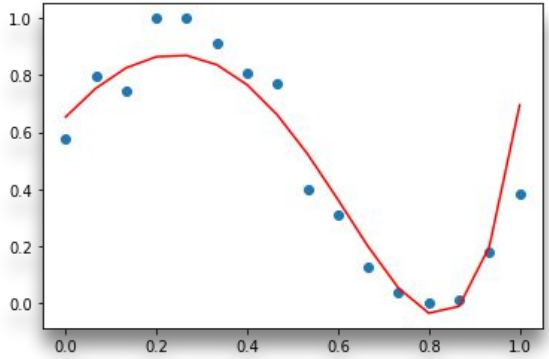
$$h(x_i) = x_i w + b$$

Overfitting



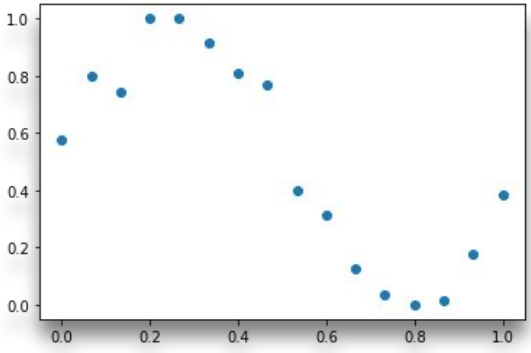
$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^{20} w_{20}$$

good



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^3 w_3$$

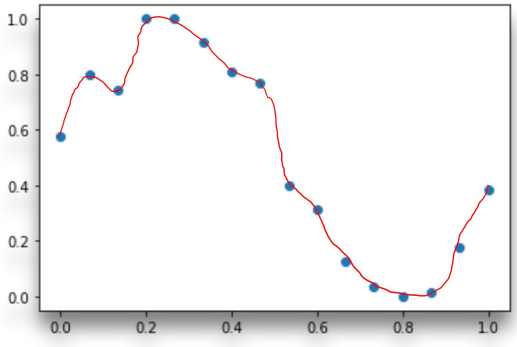
Underfitting



$$h(x_i) = x_i w + b$$

- Simple Model
- Low Capacity Model

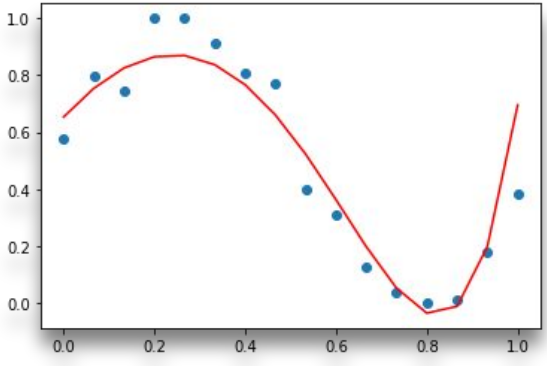
Overfitting



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^{20} w_{20}$$

- Highly Complex Model
- High Capacity Model

good



$$h(x_i) = x_i^0 w_0 + x_1^1 w_1 + ... + x_i^3 w_3$$

- Data-Fitting Model
- Model with Adequate Capacity

How do we decrease the complexity of our model?

Regularization

Regularization term

$$\mathcal{L} = \frac{||Y - h(X)||_2^2}{2n} + \mathcal{R}(w)$$

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \frac{\lambda}{n}\sum_{j=1}^{p} w_j^2$$

$$\mathcal{L} = min\{ \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \frac{\lambda}{n}\sum_{j=1}^{p} w_j^2 \}$$

$$\mathcal{L} = \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \lambda \sum_{j=1}^{p} w_j^2$$

$$\mathcal{L} = min\{ \frac{\sum_{i=0}^{n}(y_i - h(x_i))^2}{2n} + \lambda \sum_{j=1}^{p} w_j^2 \}$$

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda ||W||_2^2$$

If the parameter λ is very large, what happens?

λ = 10000

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

What happens if the parameter λ is very small?

λ = 0.01

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

Who is affected or what changes if we modify the loss function

The Derivatives

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial(\frac{||Y - XW^t||_2^2}{2n} + \lambda|W||_2^2)}{\partial w_i}$$

1. Hypothesis $$h(X) = X * w^T$$

2. Loss Function $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \frac{\lambda}{n}||W||_2^2$$

3. Derivatives $$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\sum_{i=0}^{n}(y_i - h(x_i)) * (-x_i^j)}{n} + \frac{2\lambda w_j}{n}$$

2.1 Regularization Methods in Regression

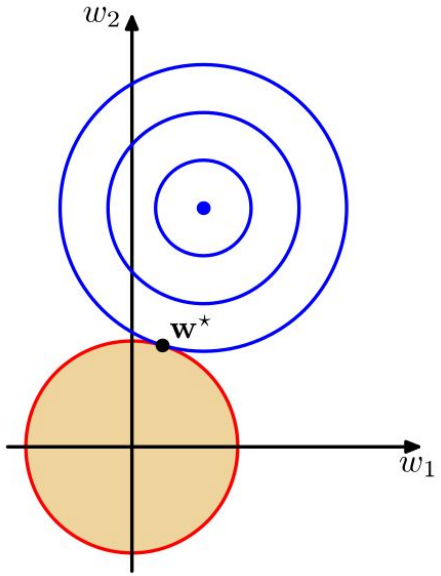
Ridge

Loss Function $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

Lasso

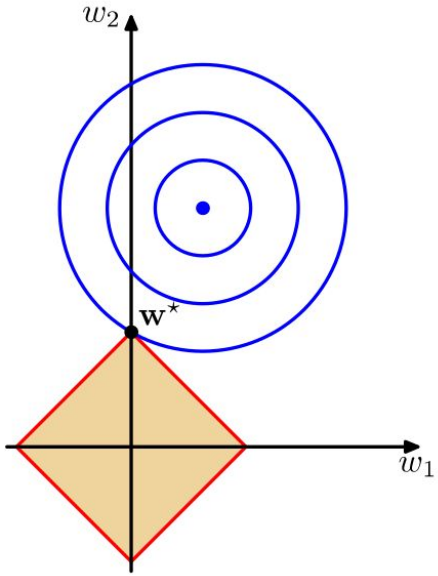
Loss Function $$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_1$$

Ridge



$$w_1^2 + w_2^2 <= s$$

Lasso



$$|w_1| + |w_2| <= s$$

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_2^2$$

$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \lambda||W||_1$$

Small values of W

Vector with multiple zero values

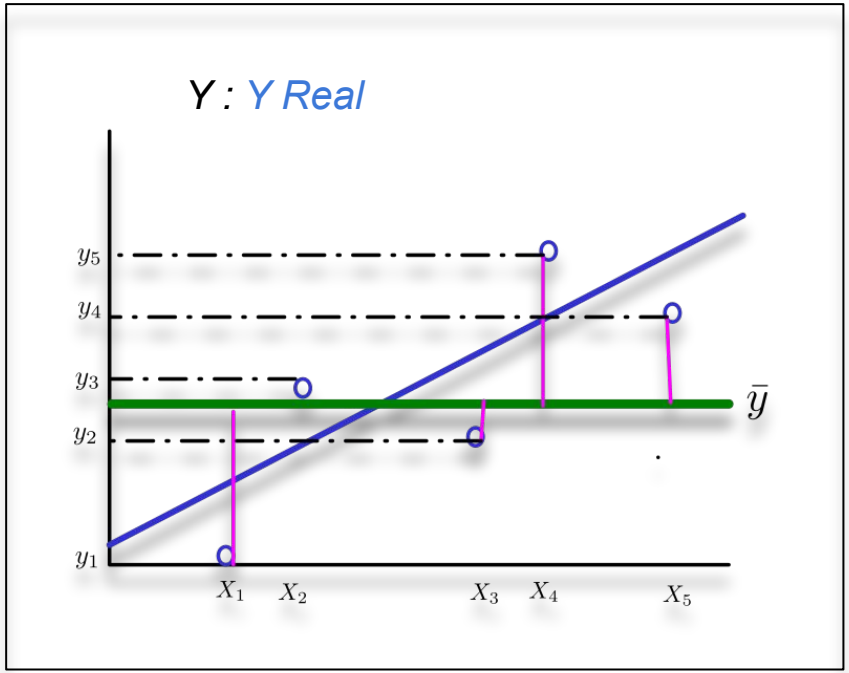
Elastinet

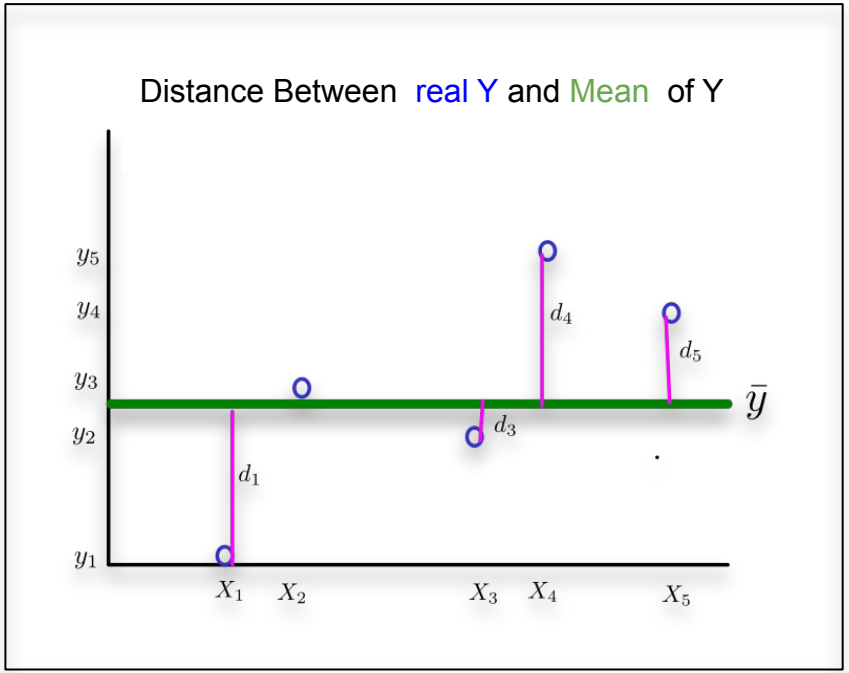
$$\mathcal{L} = \frac{||Y - XW^t||_2^2}{2n} + \rho\lambda||W||_1 + (1-\rho)\lambda||W||_2$$

2.3 Measure of Regression Quality

- Coefficient of determination R^2
- Adjusted R^2

Coefficient of determination R^2

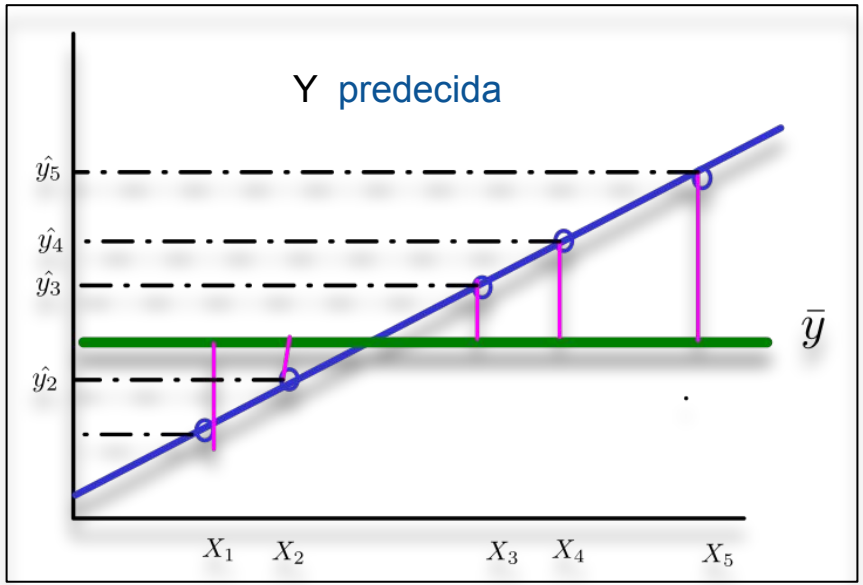


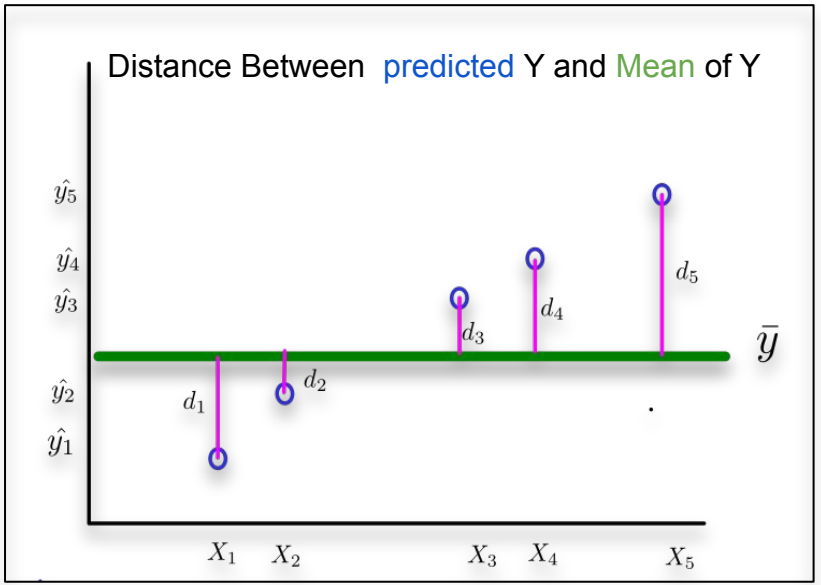


Distance Between real Y and Mean of Y :

$$(Y - \bar{Y})^2$$

Coefficient of determination R^2

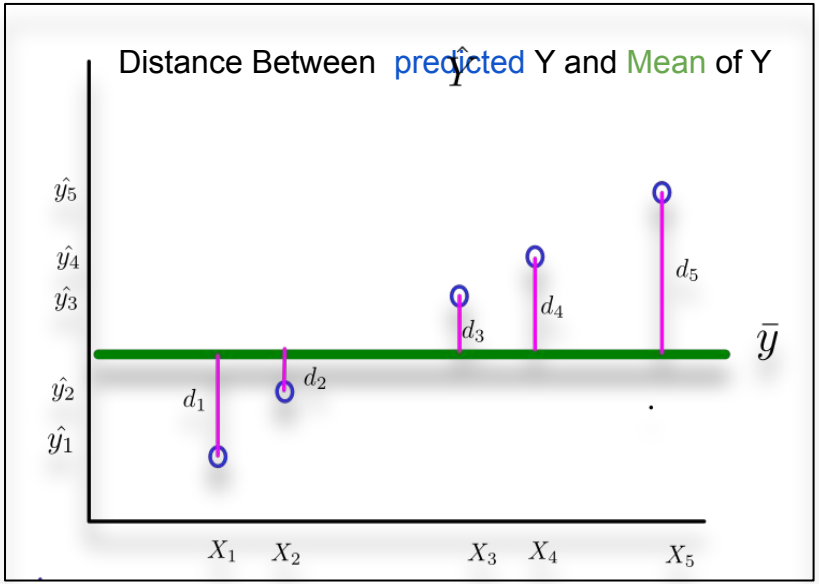


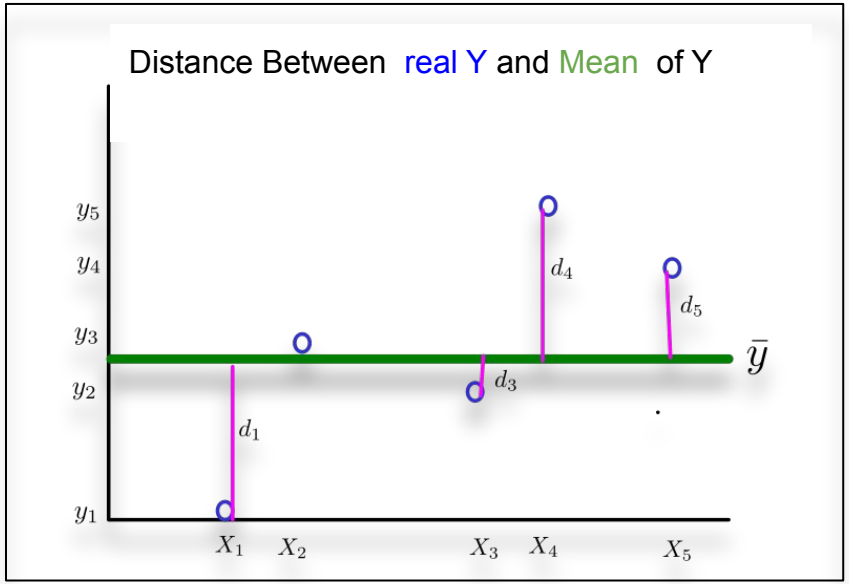


Distance Between predicted Y and Mean of Y :

$$(\hat{Y} - \bar{Y})^2$$

Coefficient of determination R^2





Coefficient of Determination

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \overline{Y})^2}{\sum_{i=1}^{n}(y_i - \overline{Y})^2} = \frac{\|\hat{Y} - \bar{Y}\|_2^2}{\|Y - \bar{Y}\|_2^2}$$

Coefficient of determination R^2

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y_i} - \overline{Y})^2}{\sum_{i=1}^{n}(y_i - \overline{Y})^2}$$

Interpretation

- R^2 = 1: The model completely explains the variation of the dependent variable.
- R^2 = 0: The model does not explain the variation of the dependent variable at all.
- Value between 0 and 1: The closer the R-squared value is to 1, the better the model.

Disadvantages

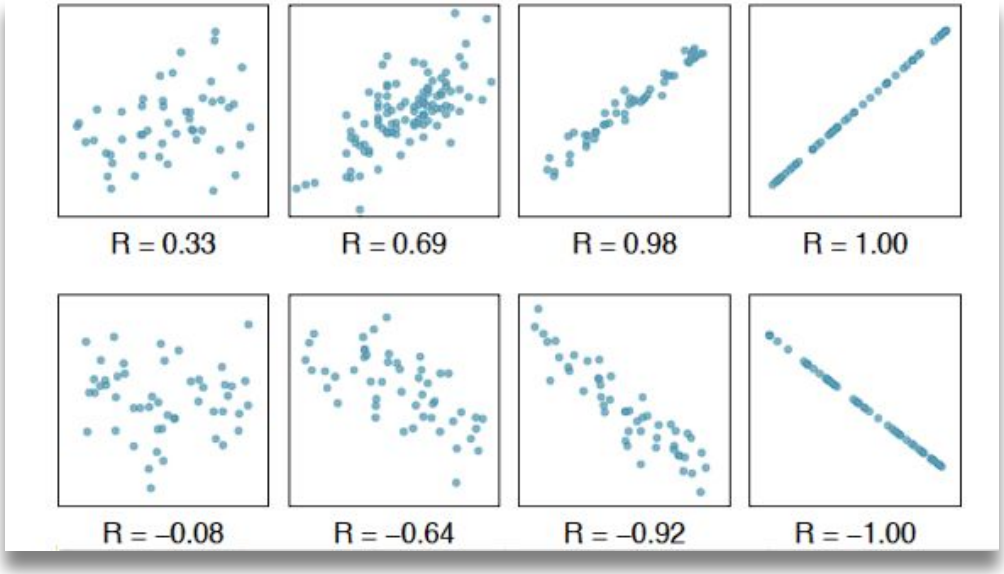
- Does not indicate causality.
- Sensitive to the number of predictors.
- Not suitable for non-linear models
- Not suitable when there are outliers
- No ajusta por tamaño de la muestra

Adjusted R^2

$$R^2_{ajustado} = 1 - \left(1 - R^2\right) \frac{n-1}{n-p-1}$$

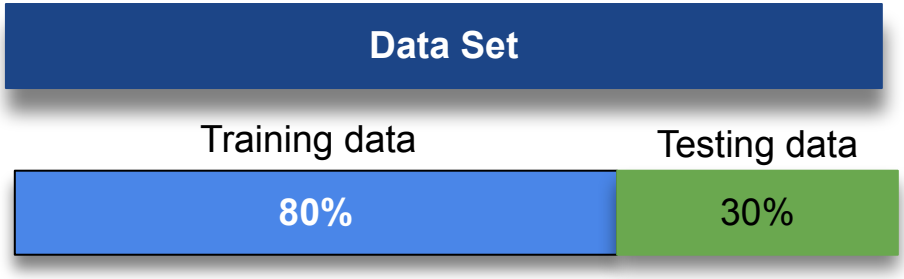
- n es el número total de observaciones
- p es el número de predictores independientes en el modelo,
- R^2 es el coeficiente de determinación no ajustado.
- Puede tomar valores negativos. En este caso las variables predictoras no tiene relación con la variable dependiente Y.

Example of correlation coefficient

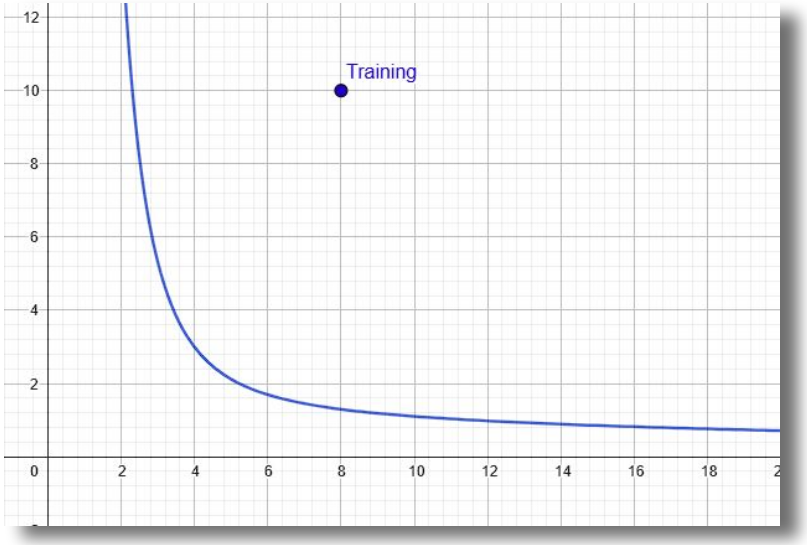


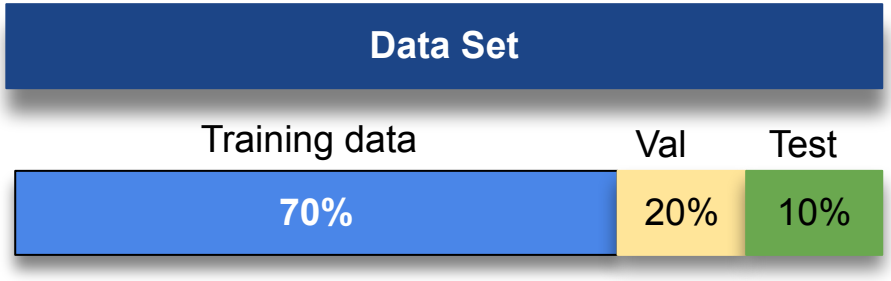
Source: Click

3 Training, Bias, and Variance

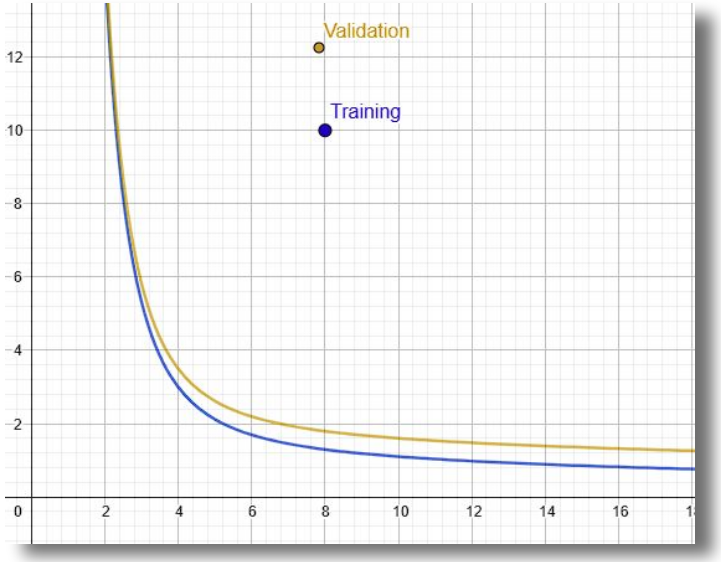


loss, w = training(x_train, y_traing, alpha, epochs)





loss, w = training(x_train, y_traing, alpha, epochs)



```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
     x,y  =  get_data(x_train, y_train, batch)
     dw = derivatives(x,y,w)
     w   = Change_Parameters(w,dw, alpha)
     loss_training = Error(x,y,w,batch)
     loss_val = Error(x_val,y_val,w,batch)
     L_T.append(loss_training)
     L_V.append(loss_val)
  return L_T, L_V, w
```

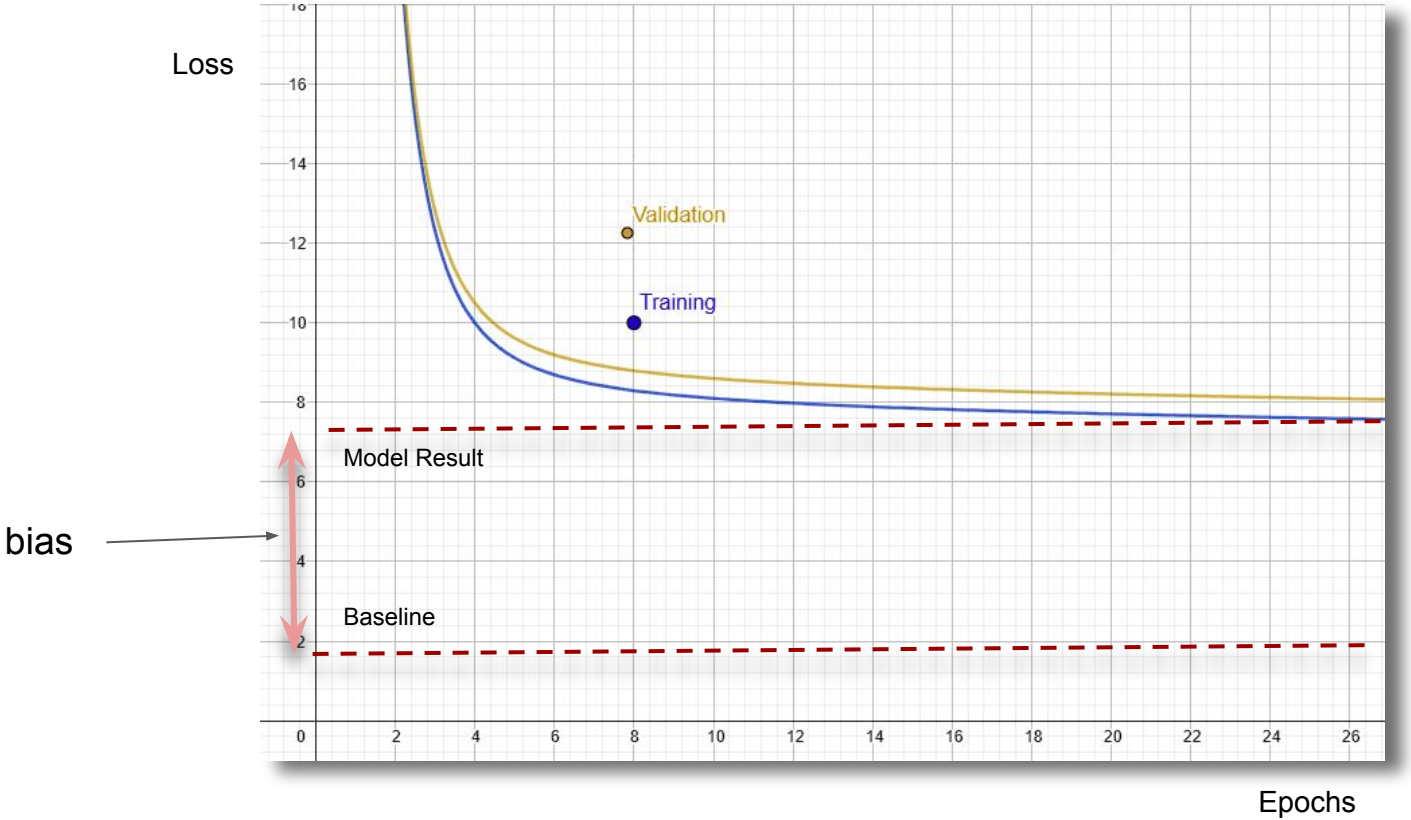
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w
```

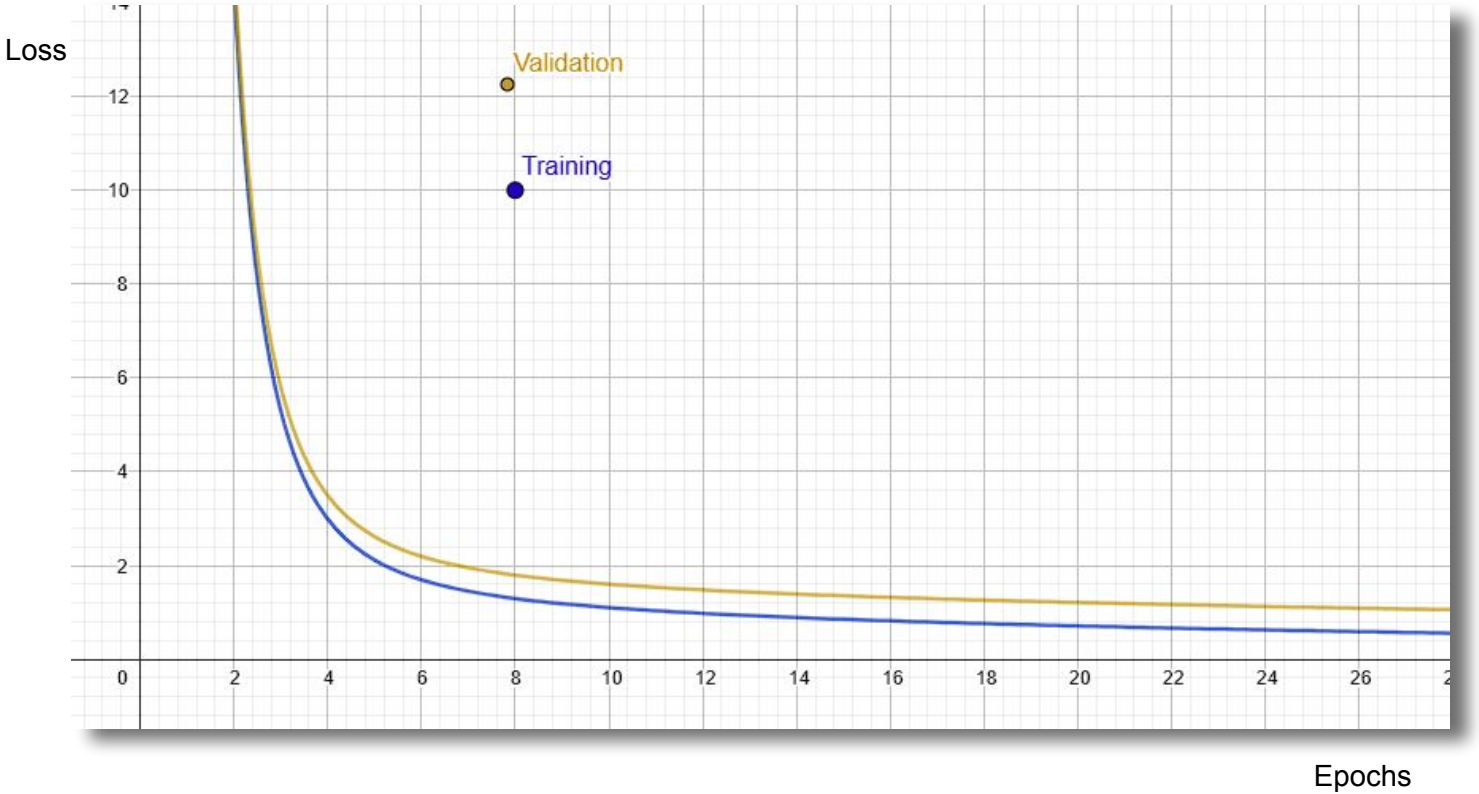
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w
```

```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w


  plot(loss_t, loss_v, epochs)
```

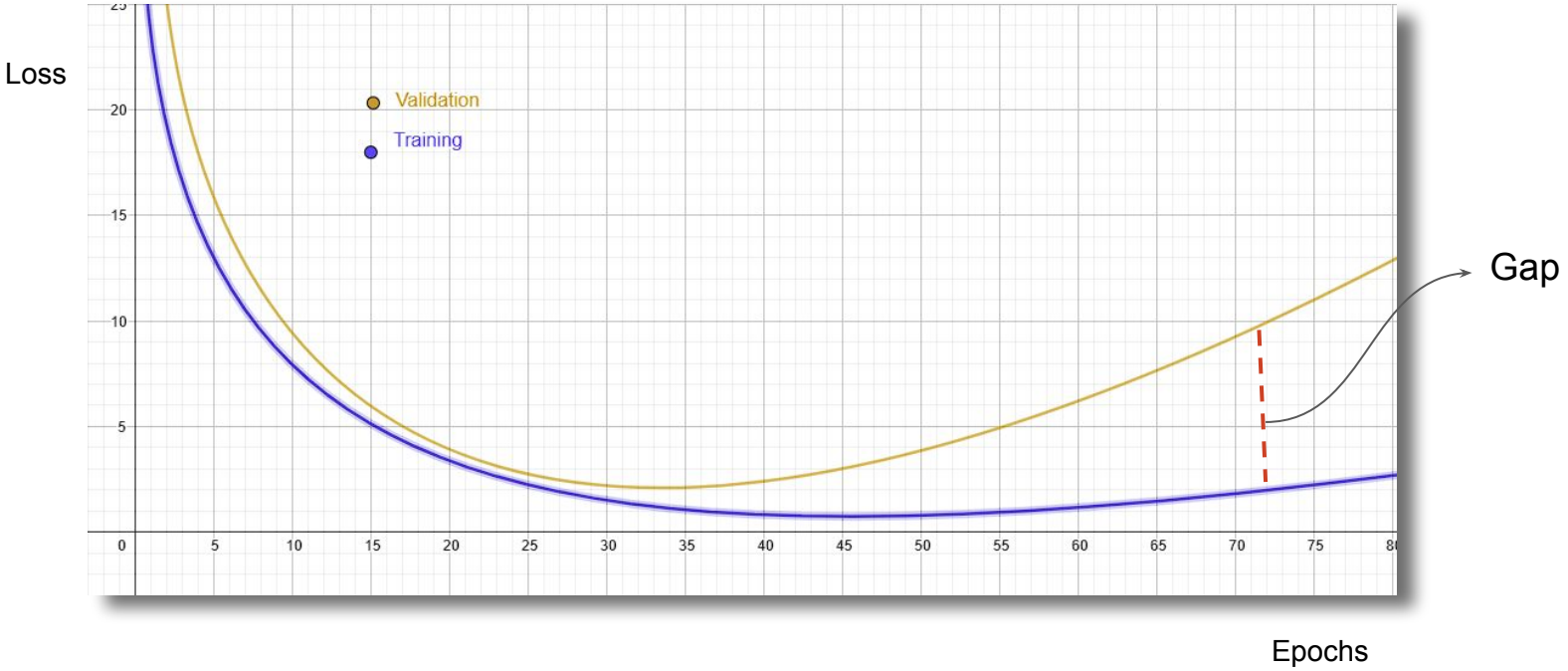
```
traing(x_train, y_train, alpha, epochs):
  w = # Initialize parameters
  for i in range(epochs):
      x,y  =  get_data(x_train, y_train, batch)
      dw = derivatives(x,y,w)
      w   = Change_Parameters(w,dw, alpha)
      loss_training = Error(x,y,w,batch)
      loss_val = Error(x_val,y_val,w,batch)
      L_T.append(loss_training)
      L_V.append(loss_val)
  return L_T, L_V, w

plot(loss_t, loss_v, epochs)
```

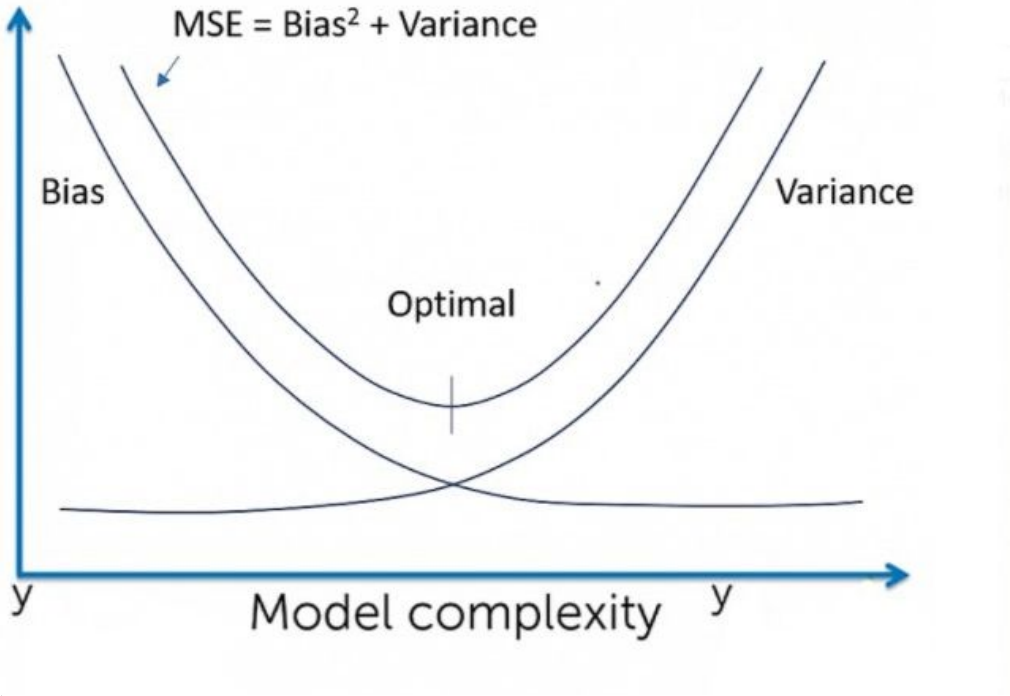
Underfitting



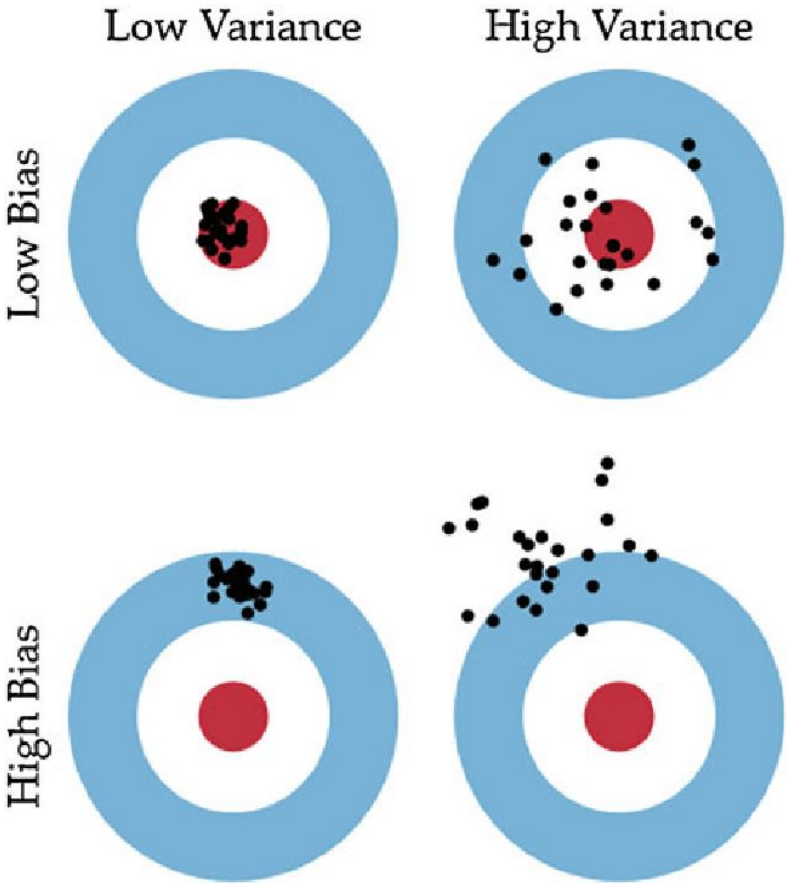

Loss
Validation
Training
12
10
8
6
4
2
0
2
4
6
8
10
12
14
16
18
20
22
24
26
Epochs

Overfitting

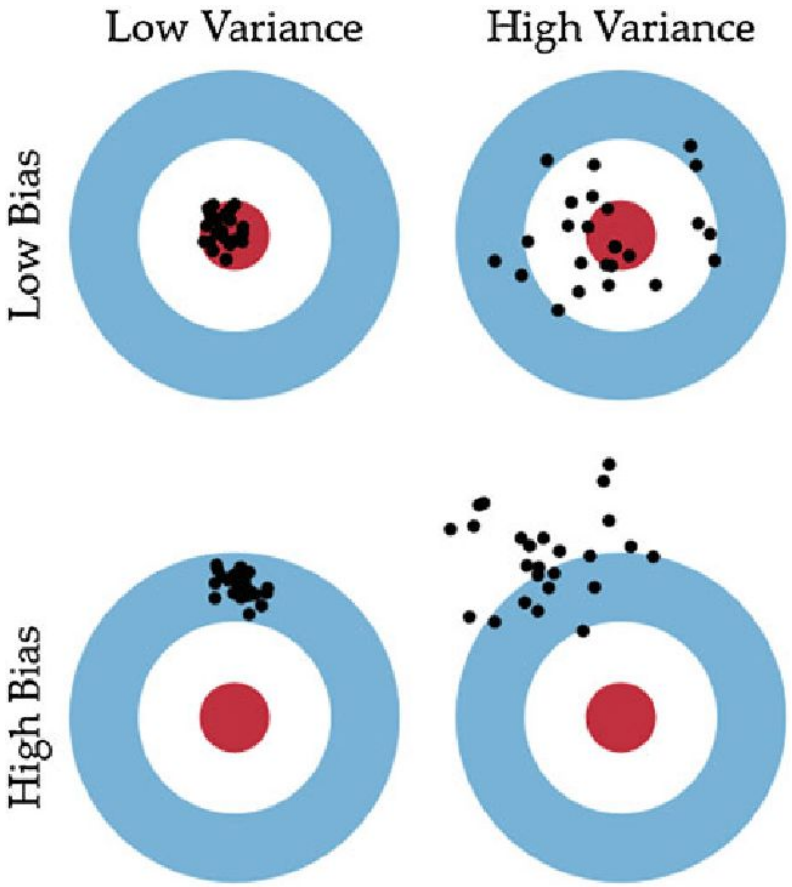




Source: https://editor.analyticsvidhya.com/uploads/983161.png


Low Variance
High Variance
Low Bias
High Bias

The model is consistent and accurate.



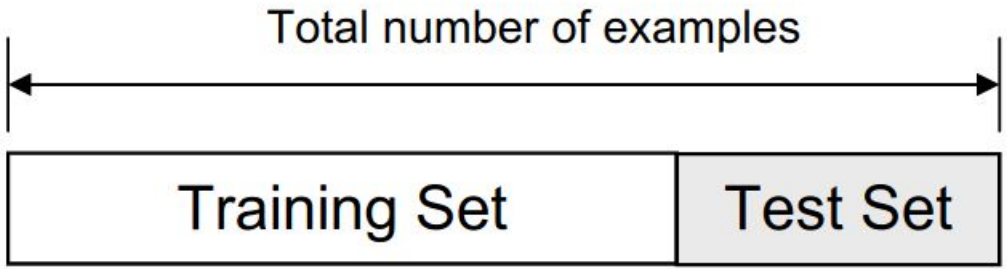
The model is uncertain but accurate

The model is consistent but inaccurate.

The model is uncertain and on average inaccurate

4 Method for evaluating the quality of generalization

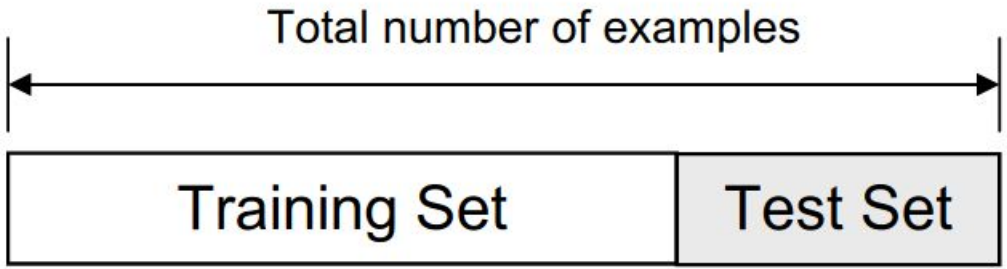
Holdout method



Problems

- What happens if we have a sparse database, or a very small one?
- Since it is only a training and testing experiment, the estimation of the error rate will be misleading in case we have a faulty split.

El método holdout



Problemas

- ¿Qué ocurre si tenemos una base de datos poco densa, o muy pequeña?
- Dado que sólo es un experimento de entrenamiento y prueba, la estimación de la tasa de error será engañosa en caso de que tengamos una división defectuosa.

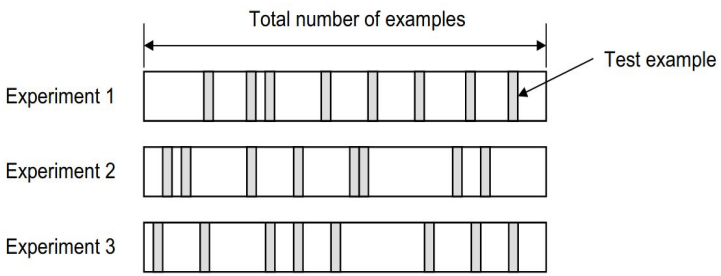
Resampling method

- Cross Validation
  - Random Subsampling
  - K-Fold Cross-Validation
  - Leave-one-out-Validation
- Bootstrap

Problems

- High computational cost

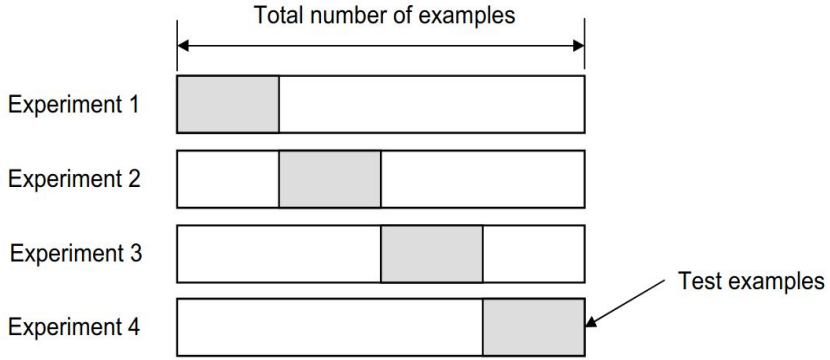
Random Subsampling



The error estimate is obtained:

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

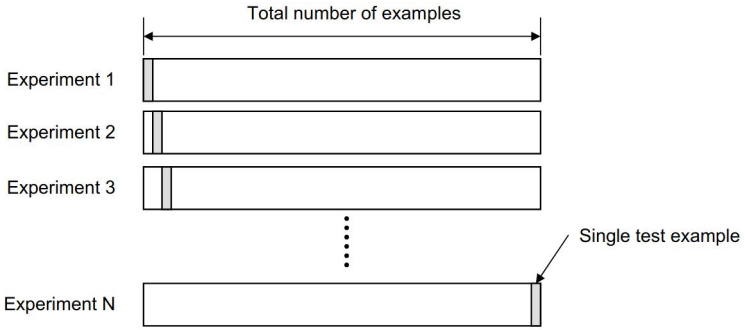
K-Fold Cross-Validation



The error estimate is obtained:

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Leave-one-out-Validation



- Para un dataset con N ejemplos, se realizan N experimentos.
- Para cada experimento se utiliza N-1 ejemplos para entrenamiento y el restante para testing.

The error estimate is obtained: $$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

How many folds are necessary?

A high number of folds

- The bias of the estimated error ratio will be small (the estimator will be more precise).
- The variance of the error estimator will be higher.
- The computational cost will be greater as we will have more experiments.

A low number of folds

- The number of experiments will be lower and therefore the computational cost.
- The variance of the estimator will be lower.
- The bias of the estimator will be higher (less precise).

In practice

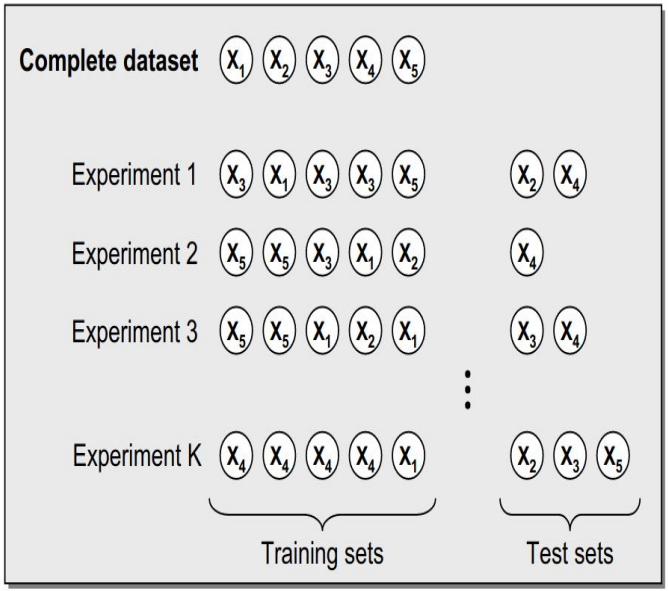
- For large databases, even 3-fold Cross Validation will be sufficient.
- A common choice in K-fold Cross Validation is K=10.
- For small databases, we should use leave-one-out.

Método Bootstrap

- Select N elements (with replacement) and use them for training.
- The remaining are used for testing.
- Repeat the process for k folds.

The error estimate is obtained:

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$



Chapter summary

Regression models predict a real value from a set of predictor variables $R^k \rightarrow R$.

What should we do in the process?

- Explore the data to intuit a possible hypothesis.
- Clean the data in case of missing data
- Formulate the regression model (hypothesis)
- Normalize the data if necessary.
- Randomly split the data into training, validation, and testing.
- Evaluate the quality of generalization using cross-validation techniques or other methods.
- Use adjusted R^2 to check if the model explains the variability of the data.
- Conduct experiments to fine-tune the hyperparameters.

Teamwork Time

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Classification

Cristian López Del Alamo
clopezd@utec.edu.pe
GINIA - Research group

2022

TRANSFORMATEC

Objective: In this class, the objective is to understand what classification is from the perspective of machine learning, what binary classification is and how it works, and finally to implement logistic regression.

Classification

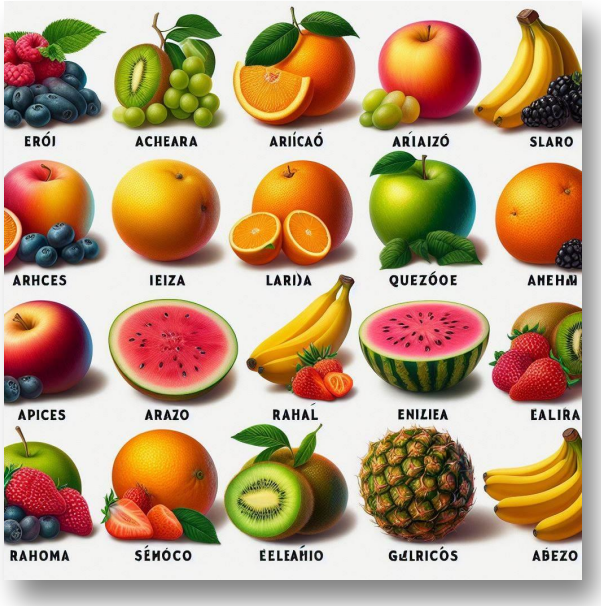
- Aprendizaje Supervisado.
- Proceso de Entrenamiento
- Datos de Entrenamiento.

- Proceso de Testing

- Proceso de Inferencia

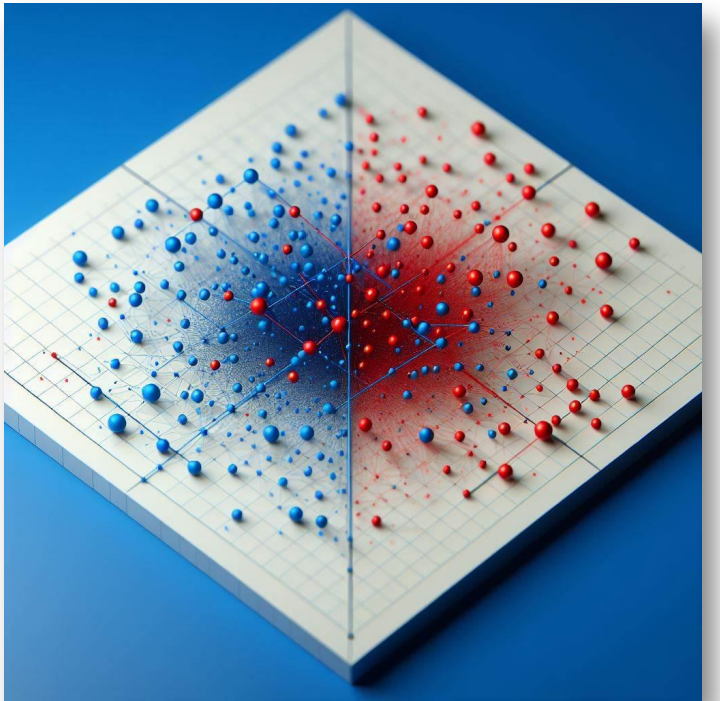
Classification

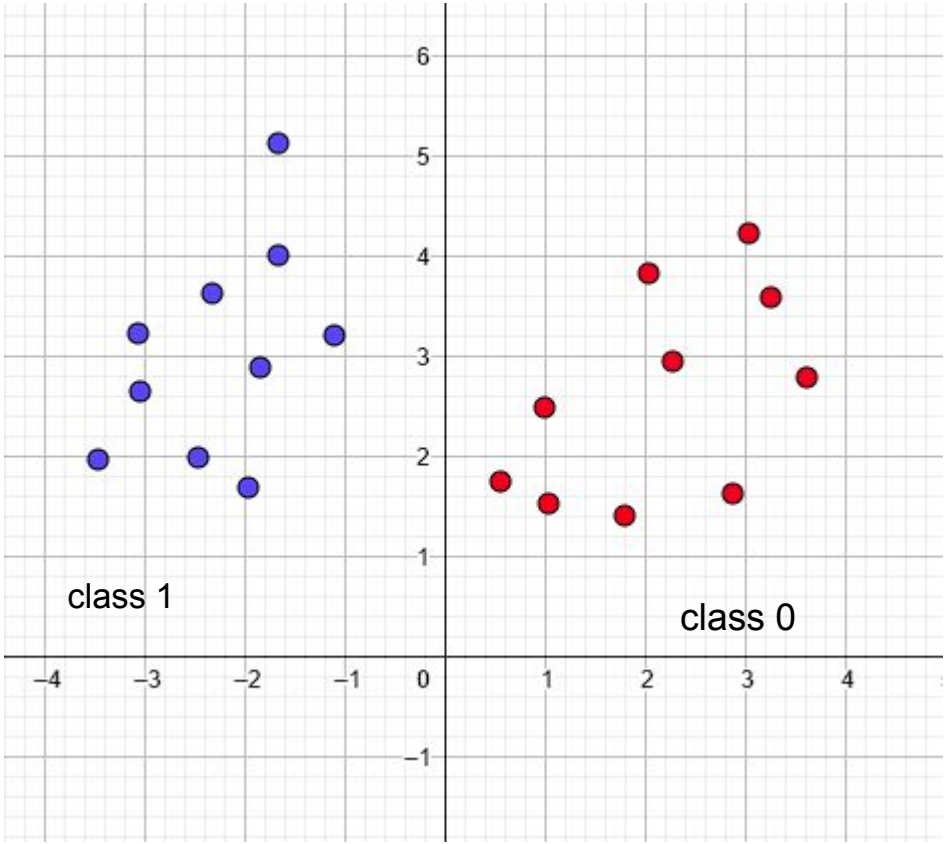
$$TrainingData = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$



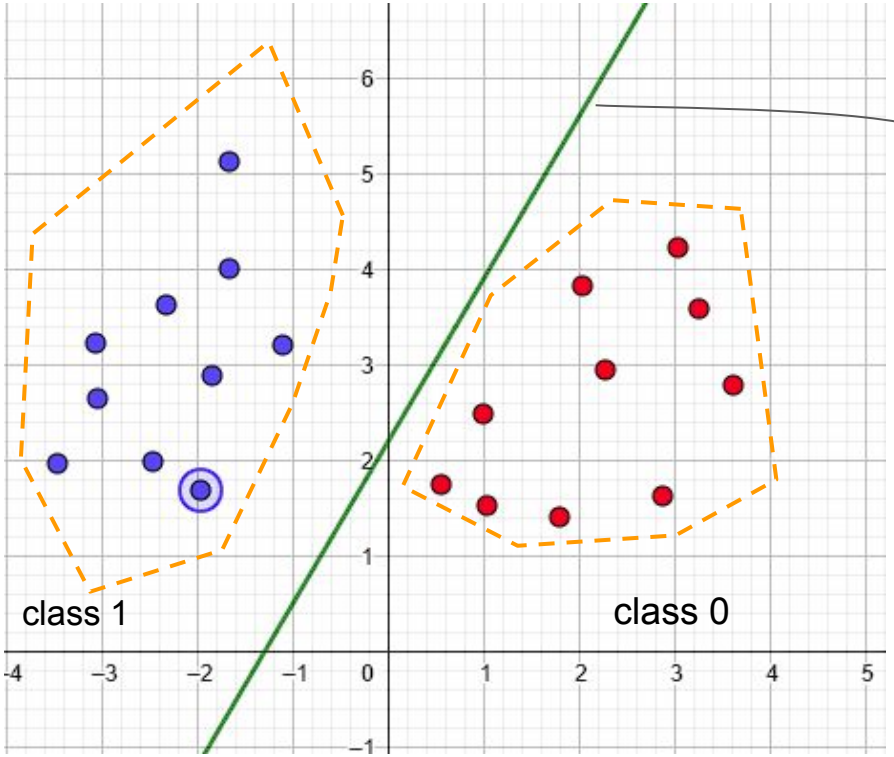
$$TestData = \{x_1, x_2, ..., x_n\}$$

Binary Classification



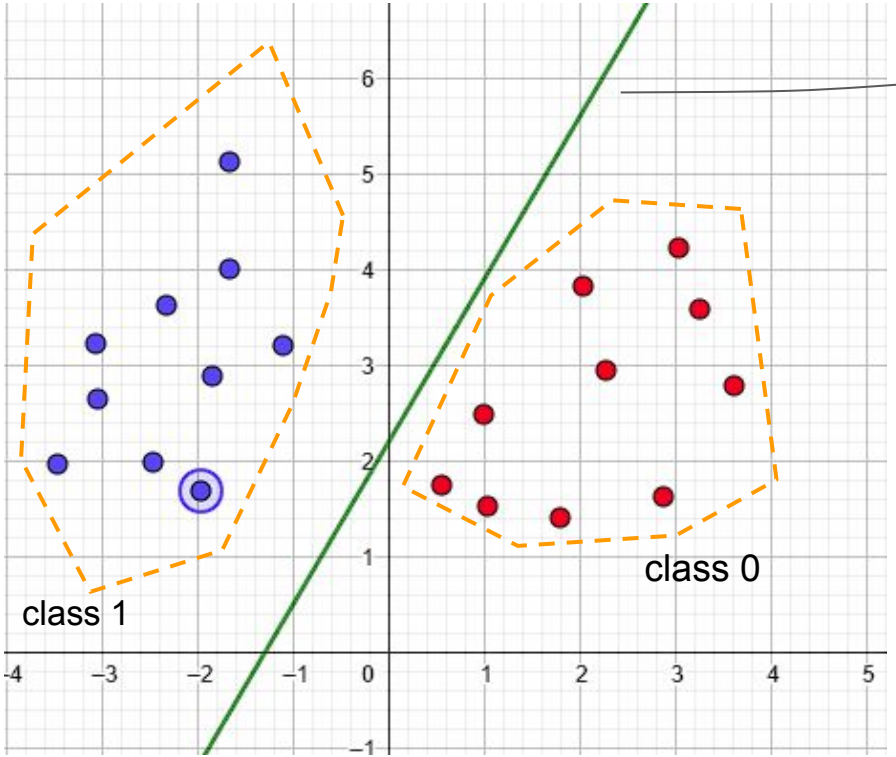

6
5
4
3
2
1
−1
−4
−3
−2
−1
0
1
2
3
4
class 1
class 0

Binary Classification



$$h(x_i) = w_0 + x_1 w_1$$

Binary Classification

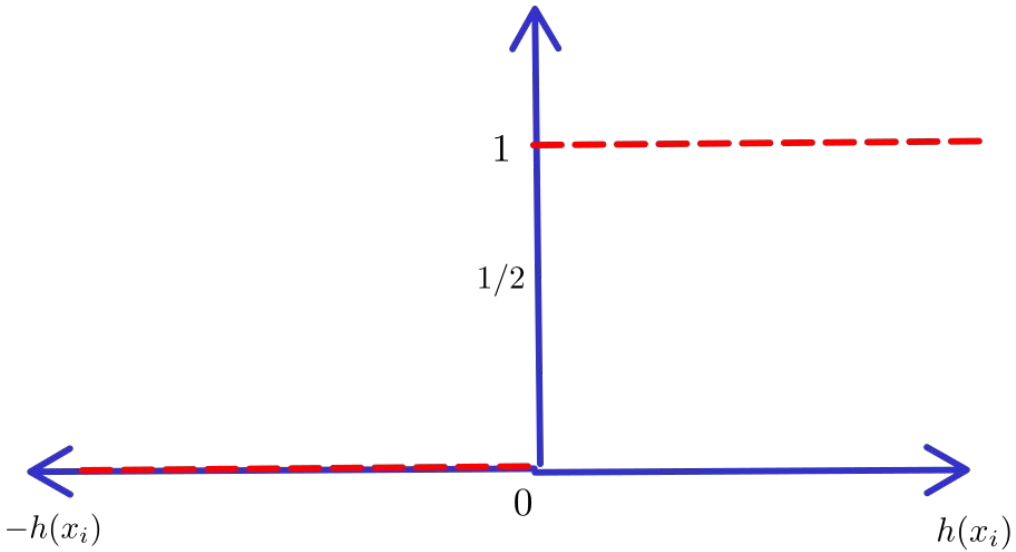


$$h(x_i) = w_0 + x_1 w_1$$

$$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

Logistic Function

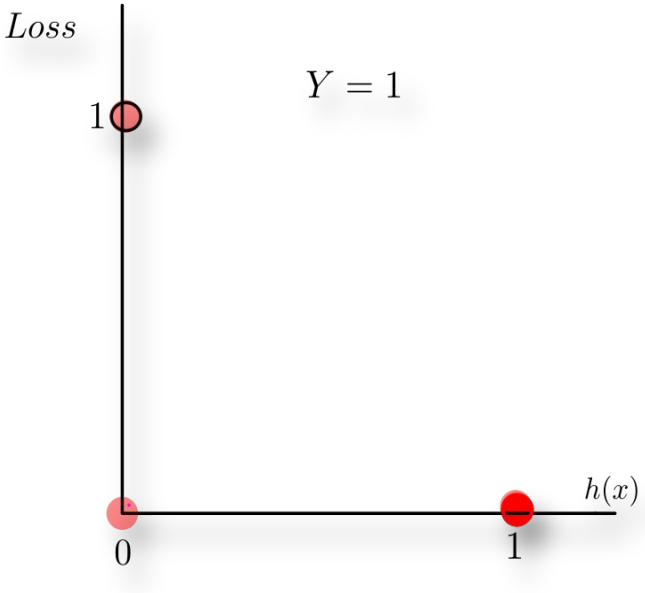
$$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

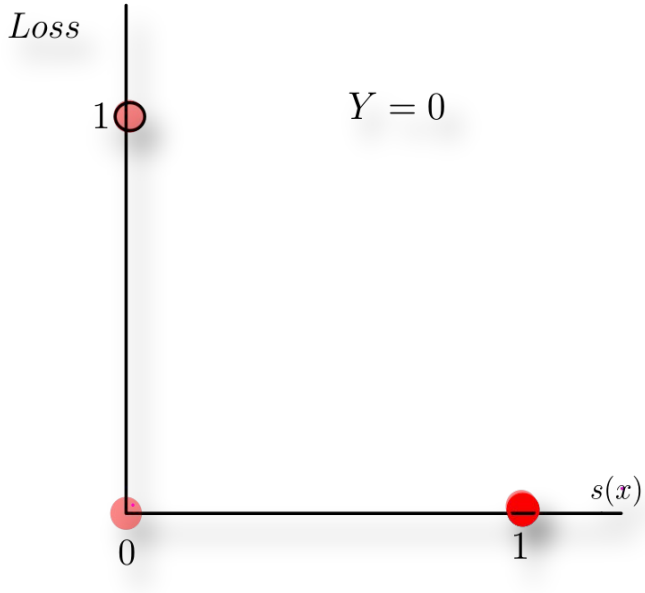


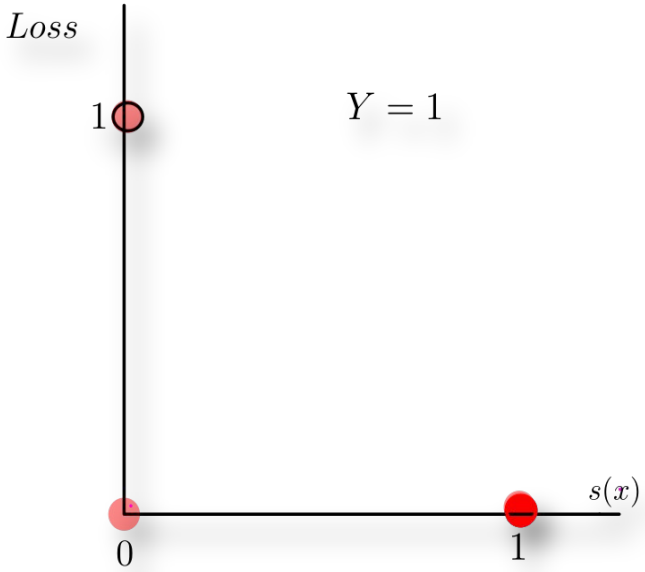
1. Hypothesis : $s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$

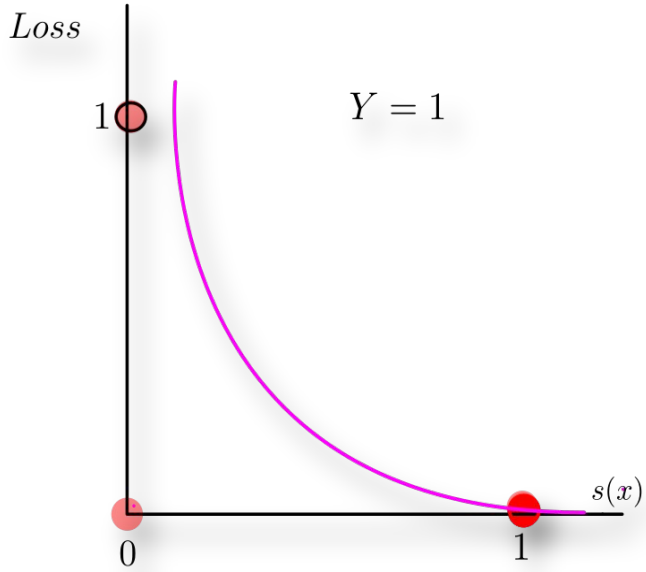
What would be the error function?

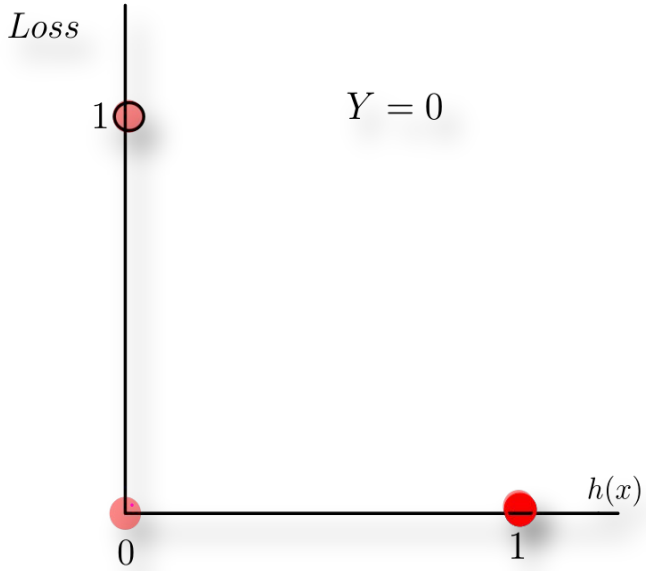
Cristian López Del Alamo

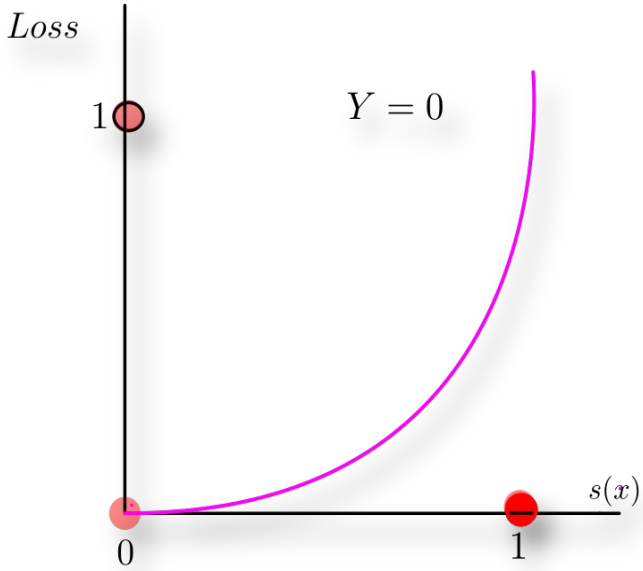

Loss
$Y = 1$
1
$h(x)$
0
1


Loss
$Y = 0$
1
$s(x)$
0
1


Loss
1
$Y = 1$
$s(x)$
0
1


Loss
1
$Y = 1$
$s(x)$
0
1


Loss
1
$Y = 0$
$h(x)$
0
1


Loss
1
$Y = 0$
$s(x)$
0
1

Si $y_i = 1 \implies \log(s(x_i))$

Si $y_i = 0 \implies \log(1 - s(x_i))$

2. Loss Function : $\mathcal{L} = -\sum_{i=1}^{n}(y_i \log(s(x_i)) + (1 - y_i) \log(1 - s(x_i)))$

1. Hypothesis : $$s(x_i) = \frac{1}{1 + e^{-h(x_i)}}$$

2. Loss Function : $$\mathcal{L} = -\sum_{i=1}^{n} (y_i \log(s(x_i)) + (1 - y_i) \log(1 - s(x_i)))$$

3. Derivadas : $$\frac{\partial L}{w_j} = \frac{1}{n} \sum_{i=1}^{n} (y_i - s(x_i))(-x_{ij})$$

```python
def train(X, Y, epochs, alfa,lam):
    np.random.seed(2001)
    W = np.array([np.random.rand() for i in range(X.shape[1])])
    L = Error(X,W,Y,lam)
    loss = []
    for i in range(epochs):
        dW = derivada(X, W, Y,lam)
        W = update(W, dW, alfa)
        L = Error(X, W,Y,lam)
        loss.append(L)
        if i%10000==0:
            print(L)
    return W, loss
```

Chapter summary

Binary classification is a type of supervised learning problem where a target variable with two possible values is predicted.

What should we do in the process?

- Explore the data to intuit a possible hypothesis.
- Clean the data in case of missing data
- We formulate the binary classification model.
- Normalize the data if necessary.
- Randomly split the data into training, validation, and testing.
- Evaluate the quality of generalization using cross-validation techniques or other methods.
- How do we measure the quality of the cluster? … **Coming soon**


UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA