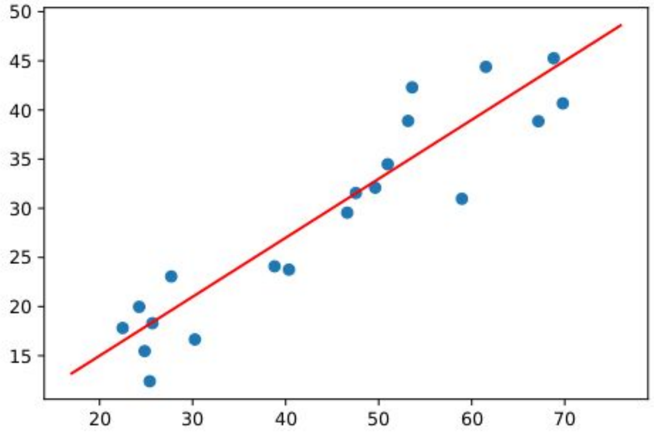
Modelos paramétricos sencillos para clasificación y regresión

MIAX-11, noviembre 2023

Contenidos

- Regresión lineal
- Regresión logística
- Problemas no lineales
- Regularización

Regresión lineal



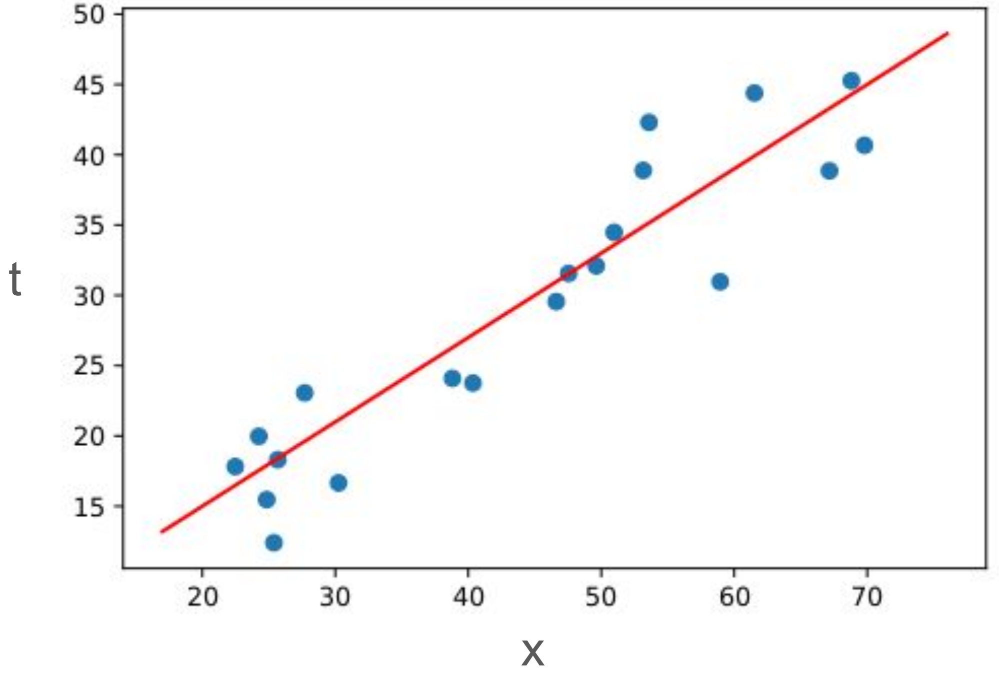
- **Problema:** $\{(\boldsymbol{x_1}, t_1), (\boldsymbol{x_2}, t_2), \ldots, (\boldsymbol{x_N}, t_N)\}$
  - $\boldsymbol{x_i}$ ≡ vector de atributos
  - t_i ≡ variable objetivo (target), $t_i \in \mathbf{R}$
  - N ≡ número de ejemplos/patrones
- **Objetivo:** predecir t a partir de $\boldsymbol{x}$
- **Modelo:** $y = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$
- **Función de coste:** $L = \sum_i (y_i - t_i)^2$ (**error cuadrático**)

El modelo se entrena buscando el conjunto de parámetros $\boldsymbol{w}$, b que minimiza la función de coste sobre los datos de entrenamiento.

Regresión lineal 1D

x es un escalar

$$y = f(x) = w\,x + b$$



$$\boxed{w = \frac{\mathrm{cov}(x,t)}{\mathrm{Var}(x)} \qquad b = \bar{t} - w\bar{x}}$$

Regresión lineal multidimensional

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$y = b + w_1 x_1 + w_2 x_2 + \ldots w_d x_d$

Transformación

$$\left.\begin{array}{l} \vec{x} = (1, x_1, x_2, \ldots, x_d) \\ y = \sum_{j=0}^{d} w_j x_j \\ \vec{w} = (w_0, w_1, \ldots, w_d) \end{array}\right\} \quad y = \vec{x} \cdot \vec{w}^{\,t}$$

Regresión lineal multidimensional

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$y = b + w_1\, x_1 + w_2\, x_2 + \ldots\, w_d\, x_d$

Matricialmente:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ & & \vdots & & \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nd} \end{pmatrix} \qquad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \qquad T = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}$$

Modelo: $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w}^T$

Solución: $\boldsymbol{w}^T = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{T}$

Ejemplos

sklearn.linear_model.LinearRegression

class `sklearn.linear_model.LinearRegression`(**, fit_intercept=True, copy_X=True, n_jobs=None, positive=False*) [source]

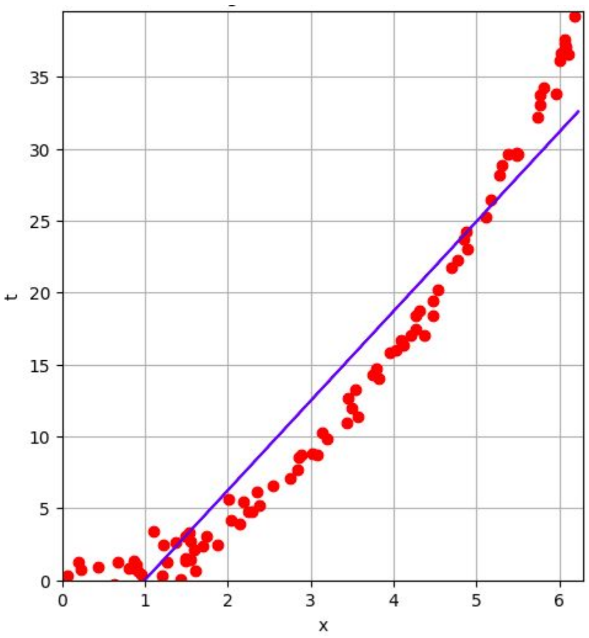
Ordinary least squares Linear Regression.

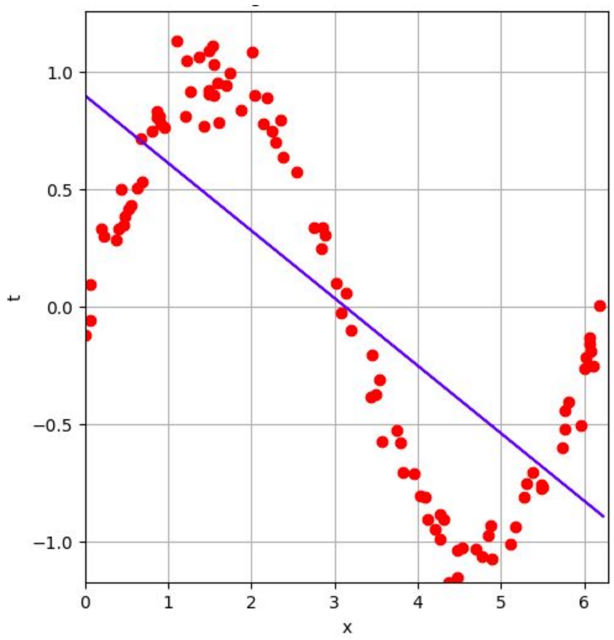
LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Ver notebook *9_1_modelos_lineales.ipynb*

Problemas no lineales



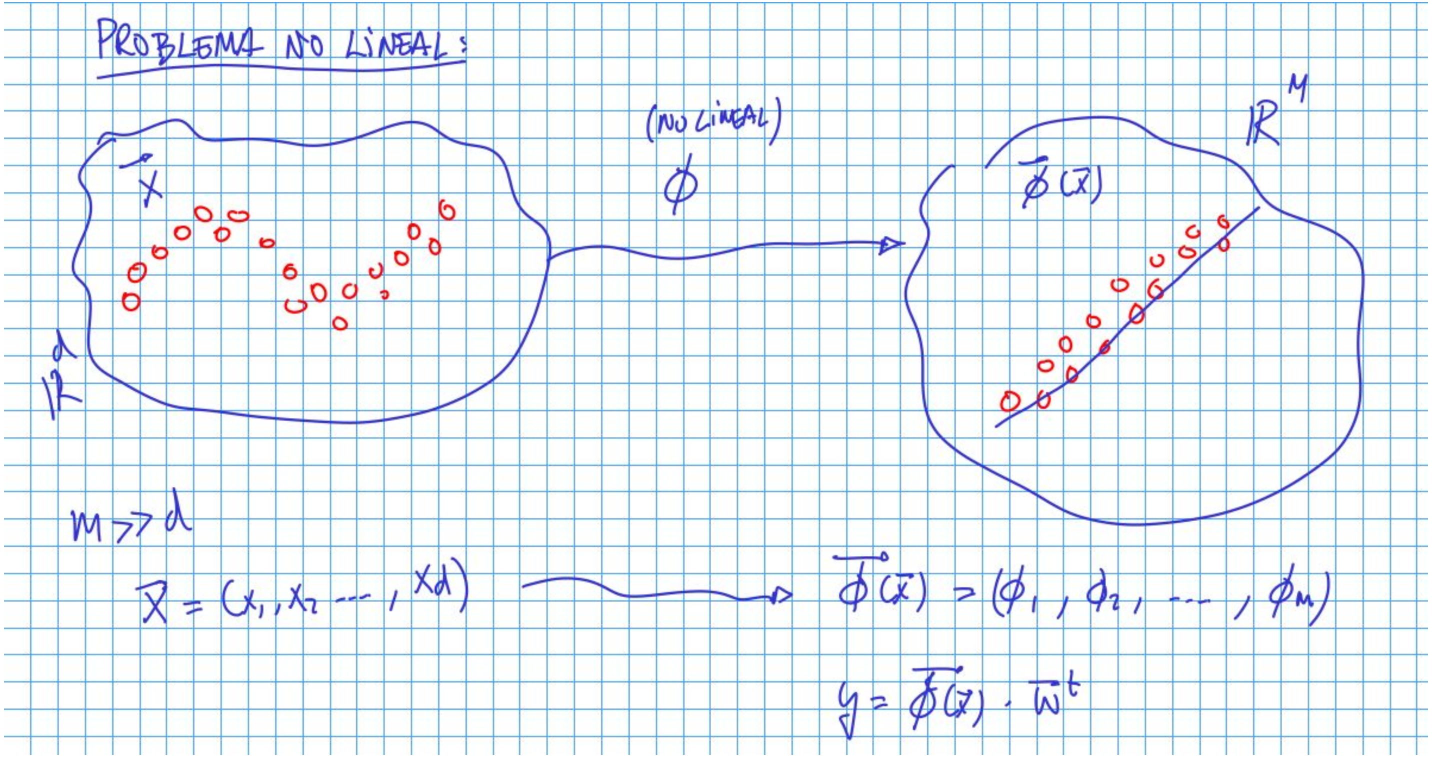


¿Podemos mantener el enfoque lineal?

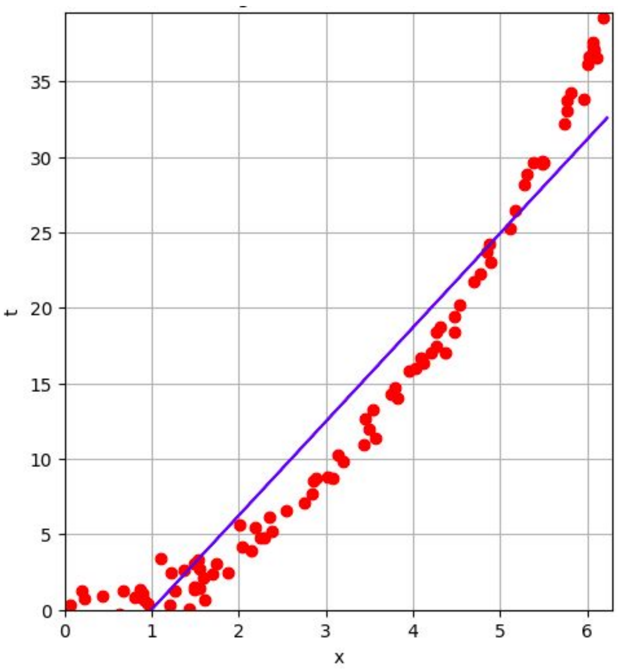
Problemas no lineales

1. Realizar una transformación no lineal de los atributos del problema

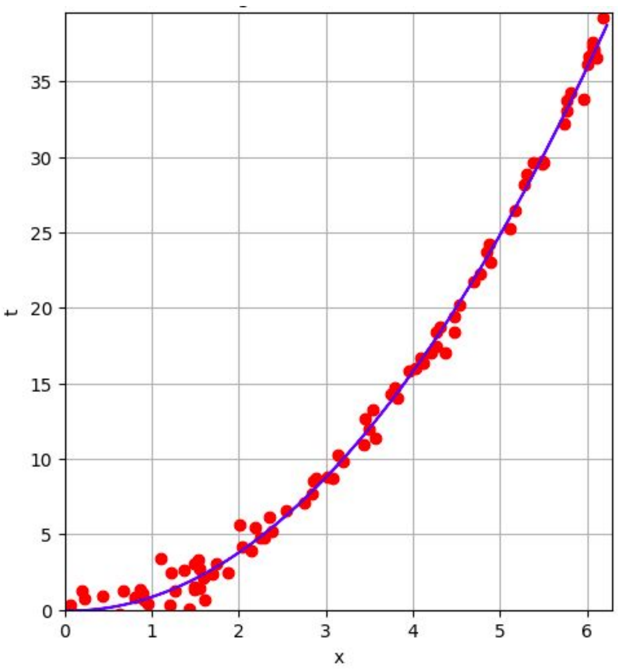
2. Resolver el problema linealmente en el nuevo espacio de atributos



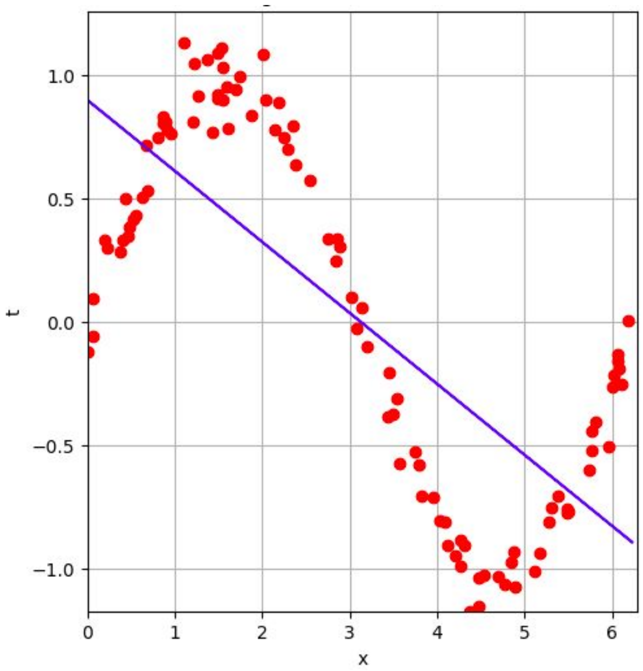
Problemas no lineales



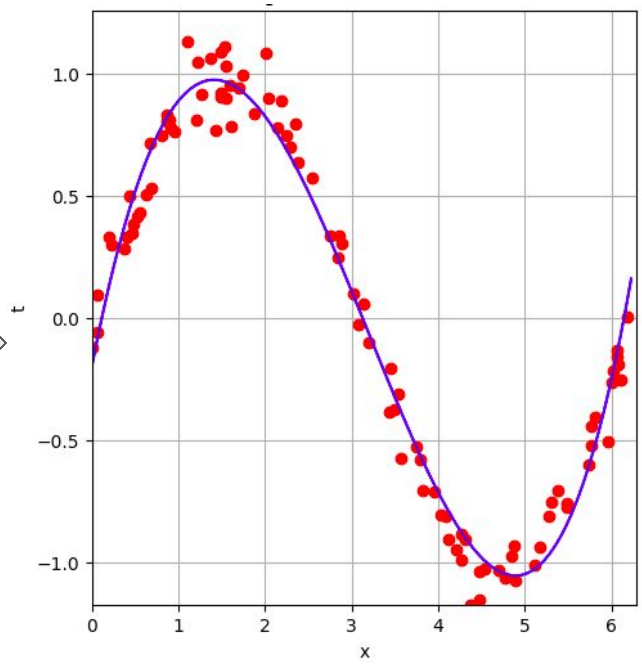
$\boldsymbol{\Phi}(x) = (x, x^2)$



Problemas no lineales

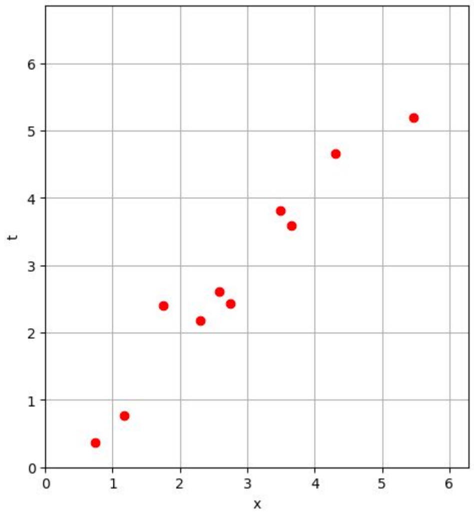


$$\boldsymbol{\Phi}(x) = (x, x^2, x^3, x^4)$$



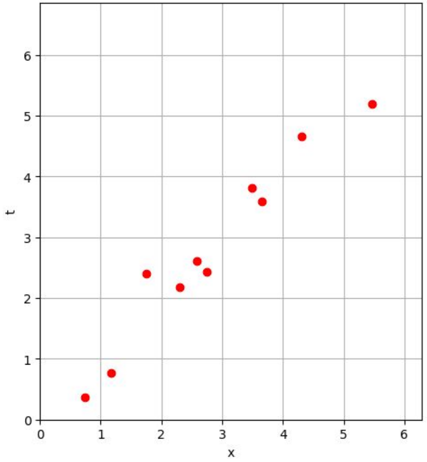
Sobre la complejidad del modelo

- ¿Qué complejidad elegir?
- ¿Deberíamos elegir un modelo lo más complejo posible? (Por si acaso…)

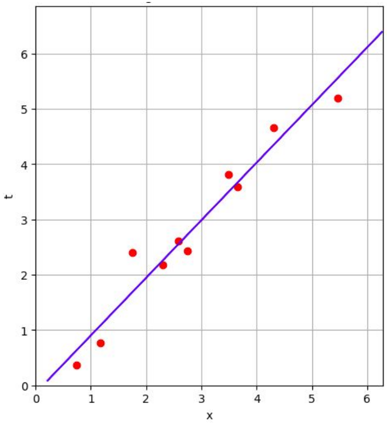


$$\left.\begin{aligned} \boldsymbol{\Phi}(x) &= (x) \\ \boldsymbol{\Phi}(x) &= (x, x^2, x^3, x^4) \\ \boldsymbol{\Phi}(x) &= (x, x^2, x^3, x^4, x^5, x^6, x^7) \end{aligned}\right\} \ \textbf{?}$$

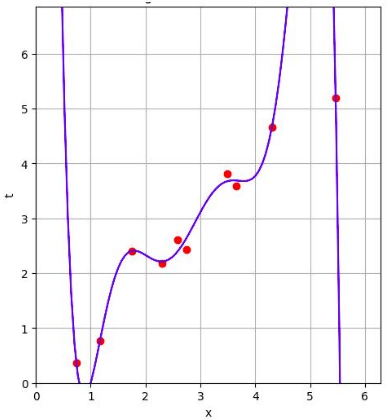
Sobre la complejidad del modelo



$\boldsymbol{\Phi}(x) = (x)$



$\boldsymbol{\Phi}(x) = (x, x^2, x^3, x^4, x^5, x^6, x^7)$



El dilema sesgo-varianza

Fuentes de error en un modelo:

- **Sesgo:** relacionado con la capacidad del modelo para ajustarse a los datos
- **Varianza:** relacionado con la sensibilidad del modelo a las variaciones en los datos
- **Error intrínseco:** inherente a los datos

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x}) - y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$

Normalmente los modelos más complejos reducen el sesgo, pero aumentan la varianza → Mayor riesgo de **overfitting**

El dilema sesgo-varianza

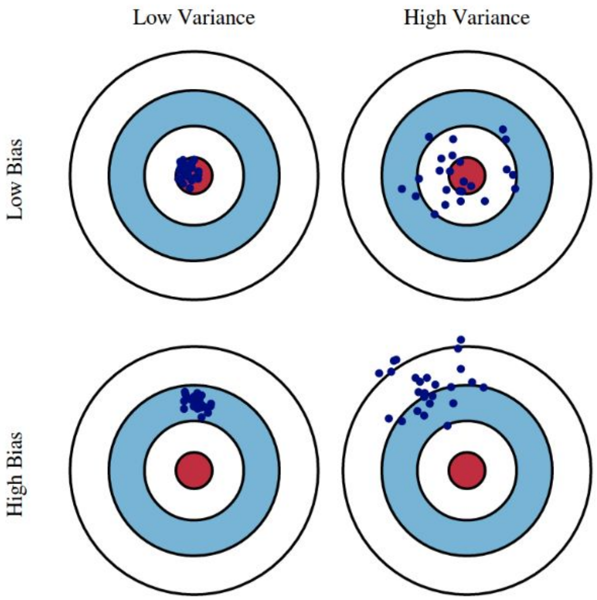
Variance: Captures how much your classifier changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set (overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

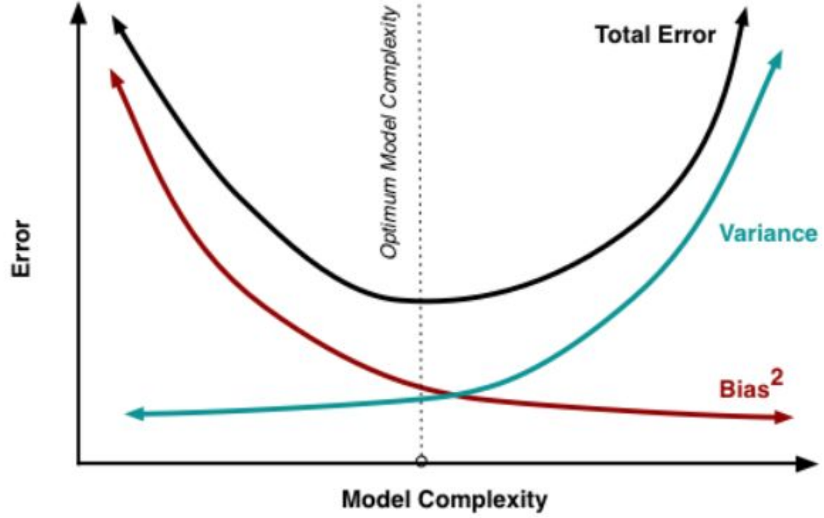
Bias: What is the inherent error that you obtain from your classifier even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to your model.

Noise: How big is the data-intrinsic noise? This error measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

El dilema sesgo-varianza





http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularización

Incluir en la función de coste términos que **penalizan la complejidad** del modelo

$$L = \underbrace{C(y,t)}_{\text{error}} + \lambda \cdot \underbrace{\text{COMPLEJIDAD}}_{\|\bar{w}\|^2}$$

¿Cómo medir la complejidad de un modelo?

Regularización L2 (Ridge)

sklearn.linear_model.Ridge

class `sklearn.linear_model.`**`Ridge`**(*alpha=1.0, *, fit_intercept=True, copy_X=True, max_iter=None, tol=0.0001, solver='auto', positive=False, random_state=None*) [source]

Linear least squares with l2 regularization.

Minimizes the objective function:

```
||y - Xw||^2_2 + alpha * ||w||^2_2
```

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape (n_samples, n_targets)).

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Regularización L1 (Lasso)

sklearn.linear_model.Lasso

class `sklearn.linear_model.`**`Lasso`**(*alpha=1.0, *, fit_intercept=True, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*) [source]

Linear Model trained with L1 prior as regularizer (aka the Lasso).

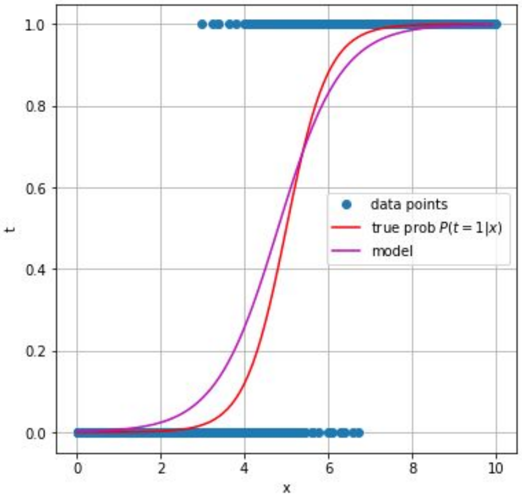
The optimization objective for Lasso is:

```
(1 / (2 * n_samples)) * ||y - Xw||^2_2 + alpha * ||w||_1
```

Technically the Lasso model is optimizing the same objective function as the Elastic Net with `l1_ratio=1.0` (no L2 penalty).

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html

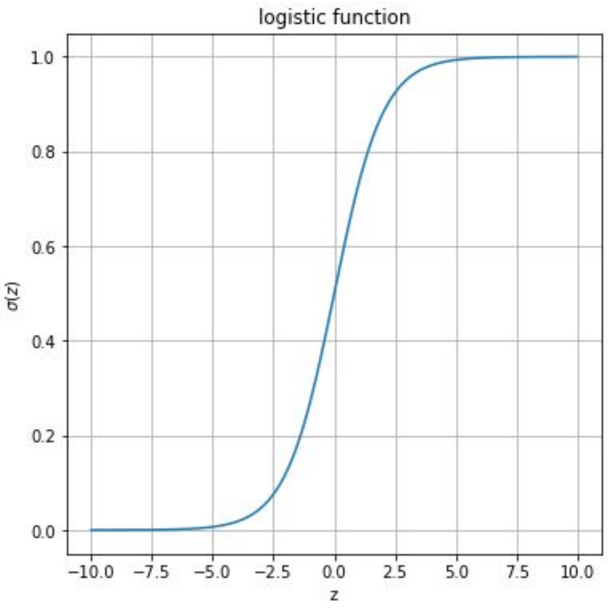
Regresión logística (clasificación)



- **Problema:** $\{(\boldsymbol{x_1}, t_1), (\boldsymbol{x_2}, t_2), \ldots, (\boldsymbol{x_N}, t_N)\}$
  - $\boldsymbol{x_i}$ ≡ vector de atributos
  - t_i ≡ variable objetivo (target), $t_i \in \{0, 1\}$
  - N ≡ número de ejemplos/patrones
- **Objetivo:** predecir t a partir de $\boldsymbol{x}$
- **Modelo:** $y = f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$
- **Función de coste:** $L = -\sum_i [t_i \log(y_i) + (1-t_i) \log(1-y_i)]$ (**cross-entropy**)

El modelo se entrena buscando el conjunto de parámetros $\boldsymbol{w}$, b que minimizan la función de coste sobre los datos de entrenamiento.

Función logística (sigmoide)



- Siempre toma valores entre 0 y 1
- La interpretamos como la probabilidad que da el modelo a la clase 1

Regresión logística en scikit learn

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.`**`LogisticRegression`**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Ejemplos

Ver notebook *9_1_modelos_lineales.ipynb*

Siguientes pasos

- Métodos de kernel
- Regresión lineal basada en kernel (kernel ridge)
- Support vector machines