Unidad IV - Regresión

Germán Braun

Facultado de Informática - Universidad Nacional del Comahue

german.braun@fi.uncoma.edu.ar

20 de septiembre de 2025

Agenda

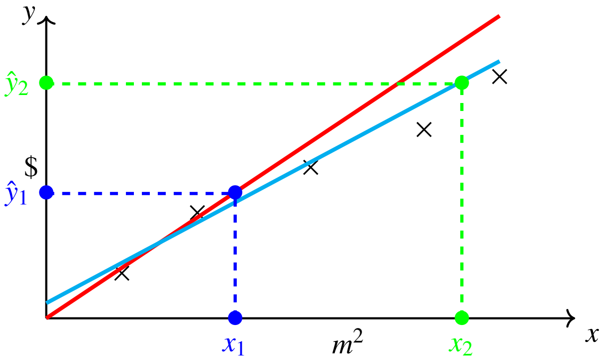
ML en la práctica

¡Recordatorio!

El aprendizaje automático es un proceso de prueba y error.

Regresión Lineal

Valores de viviendas ($) y dimensiones ($m^2$)



Dataset

m^2	$\$(k)$
50	55
65	70
120	150
150	200
...	...
210	250

El modelo de regresión predice números ($\mathbb{R}$)

(*) Adaptado del curso de Andrew Ng

Regresión Lineal

Hipótesis*

$$h(x) = \theta_0 + \theta_1 x$$

Regresión Lineal

Hipótesis*

$$h(x) = \theta_0 + \theta_1 x$$

Función de Costo (error cuadrático) *

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

Regresión Lineal

Hipótesis*

$$h(x) = \theta_0 + \theta_1 x$$

Función de Costo (error cuadrático) *

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

meta $\rightarrow$ minimizar $J(\theta_0, \theta_1)$

- θ_i: **parámetros** (o pesos)
- m: tamaño del dataset (# filas de la tabla)
- x_i: atributos ($x = m^2$, dimensión de las viviendas)
- n: # de atributos
- y: salida (*target*)
- (x, y): ejemplo de entrenamiento, $(x^{(i)}, y^{(i)})$ para fila i

(*) simplificado para una variable

Regresión Lineal (cont'd)

Hipótesis (compacta)*

$$h(x) = \sum_{j=0}^{1} \theta_j x_j$$

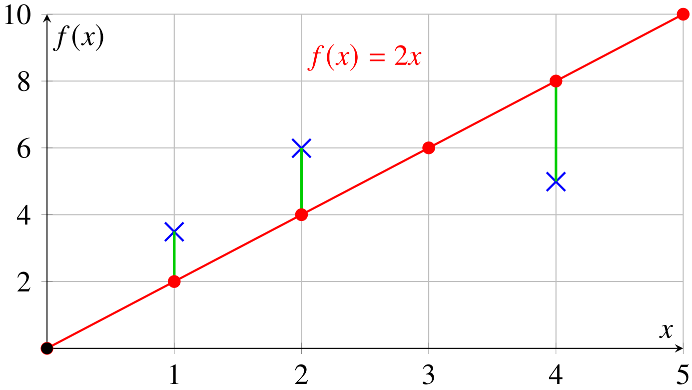
dónde $x_0 = 1$ (dummy)

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \qquad \vec{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

(*) simplificado para una variable

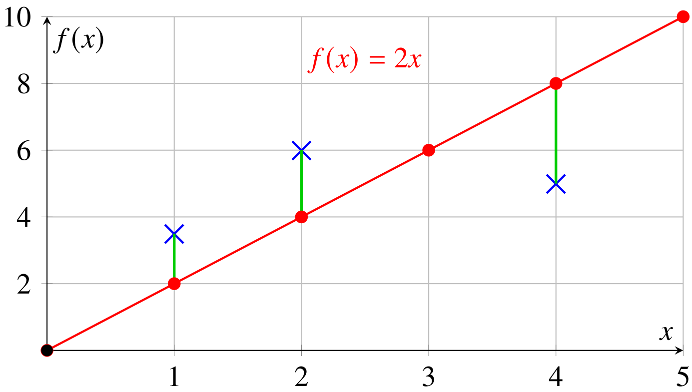
Función de costo - Intuición

(*)Para valores fijos de θ_0, θ_1, el costo es función de x



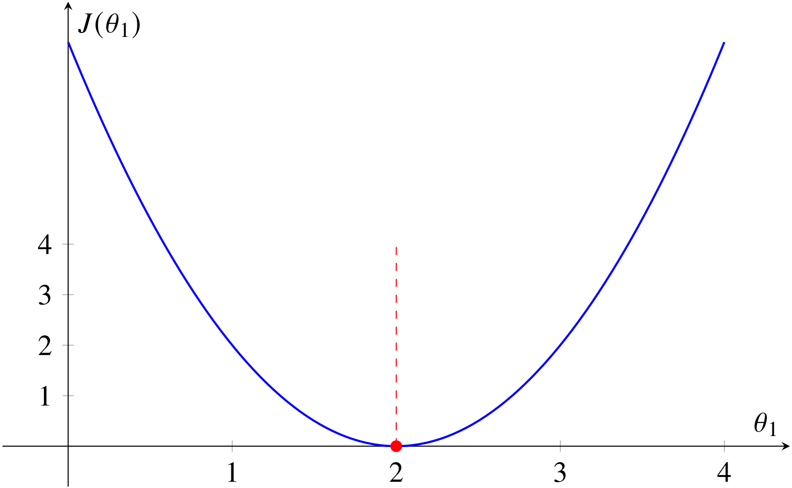
Función de costo - Intuición

(*)Para valores fijos de θ_0, θ_1, el costo es función de x



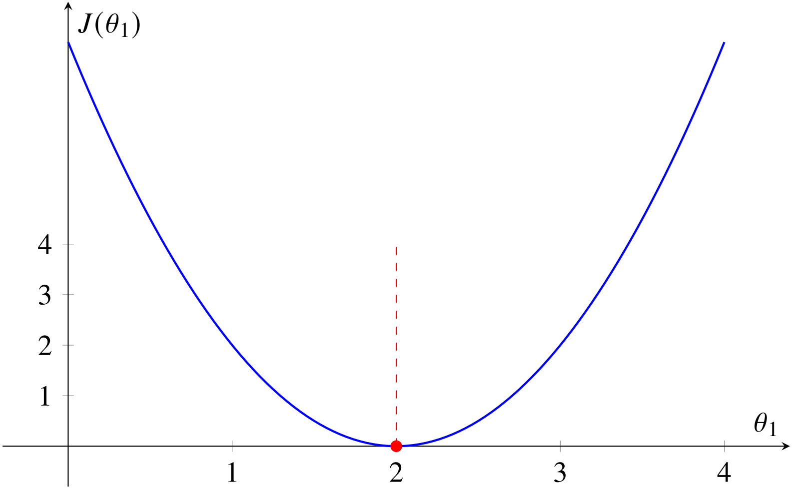
$$J(2) = \frac{1}{2*3}[(2-3{,}5)^2 + (4-6)^2 + (8-5)^2] = \frac{1}{6}[2{,}25 + 4 + 9] = \frac{15{,}25}{6} = 2,54$$

Función de costo - Intuición



- Asumimos $\theta_0 = 0$ y por lo anterior $\theta_1 = 2 \rightarrow f(x) = 2x$

Función de costo - Intuición



- Asumimos $\theta_0 = 0$ y por lo anterior $\theta_1 = 2 \longrightarrow f(x) = 2x$
- ? **¿Cómo buscamos los valores de θ tal que el error calculado sea el mínimo?**

Método del Gradiente - Intuición

Función de costo

$$J(\theta_0, \theta_1)$$

$$\rightarrow J(\theta_0, \theta_1, ..., \theta_n)$$

Objetivo

$$min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

$$\rightarrow min_{\theta_0, \theta_1, ..., \theta_n} J(\theta_0, \theta_1, ..., \theta_n)$$

Método del Gradiente - Intuición

Función de costo

$$J(\theta_0, \theta_1)$$

$$\rightarrow J(\theta_0, \theta_1, ..., \theta_n)$$

Objetivo

$$min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

$$\rightarrow min_{\theta_0, \theta_1, ..., \theta_n} J(\theta_0, \theta_1, ..., \theta_n)$$

Enfoque

- Comenzar con algunos valores para θ_0, θ_1
- Cambiar θ_0, θ_1 para reducir $J(\theta_0, \theta_1)$ hasta que terminemos en un mínimo (situación ideal!)

Método del Gradiente - Intuición

Función de costo

$$J(\theta_0, \theta_1)$$

$$\rightarrow J(\theta_0, \theta_1, ..., \theta_n)$$

Objetivo

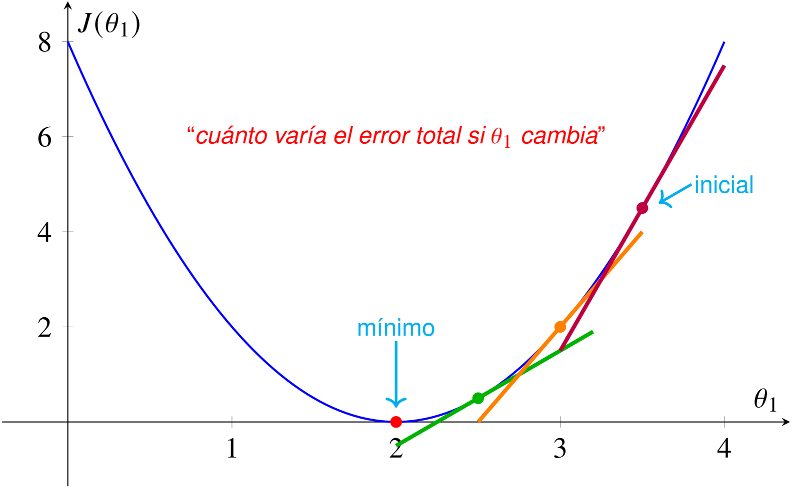
$$min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

$$\rightarrow min_{\theta_0, \theta_1, ..., \theta_n} J(\theta_0, \theta_1, ..., \theta_n)$$

Enfoque

- Comenzar con algunos valores para θ_0, θ_1
- Cambiar θ_0, θ_1 para reducir $J(\theta_0, \theta_1)$ hasta que terminemos en un mínimo (situación ideal!)

Método de Descenso del Gradiente (*Gradient Descent*)



$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \Big((\theta_0 + \theta_1 x_i) - y_i \Big) x_i$$

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo
- El algoritmo más simple se denomina *batch*, debido a que cada paso itera sobre el conjunto completo de entrenamiento antes de actualizar un parámetro

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo
- El algoritmo más simple se denomina *batch*, debido a que cada paso itera sobre el conjunto completo de entrenamiento antes de actualizar un parámetro
- Es ineficiente en caso de grandes datasets

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo
- El algoritmo más simple se denomina *batch*, debido a que cada paso itera sobre el conjunto completo de entrenamiento antes de actualizar un parámetro
- Es ineficiente en caso de grandes datasets
- Alternativas al enfoque *batch*: estocástico

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo
- El algoritmo más simple se denomina *batch*, debido a que cada paso itera sobre el conjunto completo de entrenamiento antes de actualizar un parámetro
- Es ineficiente en caso de grandes datasets
- Alternativas al enfoque *batch*: estocástico
  - Actualiza un parámetro según el gradiente del error con respecto a cada ejemplo de entrenamiento

Método del gradiente (cont'd)

- El gradiente es aplicado hasta converger, i.e. θ se estabiliza cerca del mínimo
- El método introduce el parámetro α, el cual refieren a la tasa de aprendizaje (*learning rate*)
  - controla el cambio de θ en cada paso
  - si es "muy bajo", gradiente es lento
  - si es "muy grande", puede no converger a un mínimo
- El algoritmo más simple se denomina *batch*, debido a que cada paso itera sobre el conjunto completo de entrenamiento antes de actualizar un parámetro
- Es ineficiente en caso de grandes datasets
- Alternativas al enfoque *batch*: estocástico
  - Actualiza un parámetro según el gradiente del error con respecto a cada ejemplo de entrenamiento
  - Funciona "bien" para grandes datasets

Algoritmo para Regresión Lineal

1: **Inicializar parámetros:**
  $\theta_0 \leftarrow 0,\ \theta_1 \leftarrow 0$ # parámetros iniciales
2: **Definir hiperparámetros:**
  learning_rate $\leftarrow \alpha$, max_iter, ϵ
3: **Cargar datos:**
  $X = [x_1, \dots, x_m],\ Y = [y_1, \dots, y_m],\ m \leftarrow |X|$
4: **Definir predicción:**
  $h_\theta(x) = \theta_0 + \theta_1 x$
5: **Definir función de costo:**
  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$
6: **for** $iter = 1$ to max_iter **do**
7:   Calcular gradientes:
  $grad_0 = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$
  $grad_1 = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \cdot x_i$
8:   Actualizar parámetros:
  $\theta_0 \leftarrow \theta_0 - learning\_rate \cdot grad_0$
  $\theta_1 \leftarrow \theta_1 - learning\_rate \cdot grad_1$
9:   Calcular costo actual: $cost = J(\theta_0, \theta_1)$
10:   **if** $|grad_0| < \epsilon$ **and** $|grad_1| < \epsilon$ **then**
11:     **break**
12:   **end if**
13: **end for**
14: **Retornar:** θ_0, θ_1, costo final

Consideraciones finales + Lab

- implementaciones actuales, tales como `scikt-learn`, ya hacen el cálculo de θ sin el gradiente *batch*. Por ej, con la función `LinearRegression()`[1]

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

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html

Consideraciones finales + Lab

- implementaciones actuales, tales como `scikt-learn`, ya hacen el cálculo de θ sin el gradiente *batch*. Por ej, con la función `LinearRegression()`[1]
- en caso de usar gradiente, se emplea la función `SGDRegressor()`[2], para computar el gradiente estocástico

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

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html

Regresión Logística

Intuición

Intutivamente, podríamos aplicar un algoritmo de regresión lineal para resolver problemas de clasificación

Intuición

Intutivamente, podríamos aplicar un algoritmo de regresión lineal para resolver problemas de clasificación

Sin embargo, esto no parece ser una buena idea!

Intuición

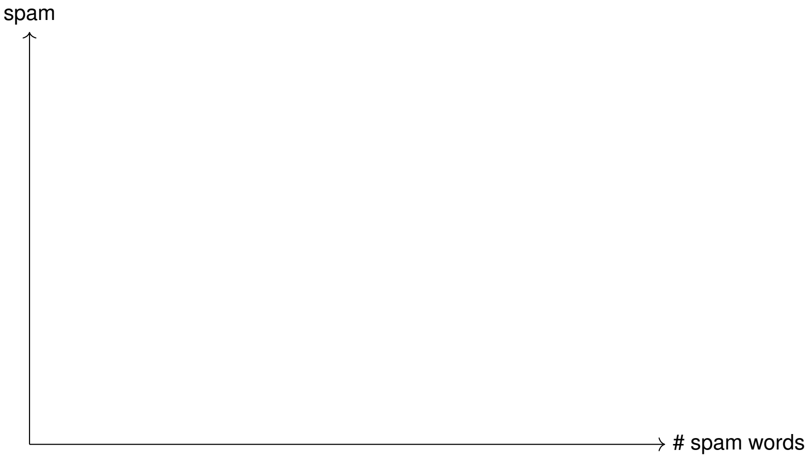
Intutivamente, podríamos aplicar un algoritmo de regresión lineal para resolver problemas de clasificación

Sin embargo, esto no parece ser una buena idea!

No tendría sentido que nuestro $h(x) > 1$ o < 0 siendo que $y \in 0, 1$

Además...

Intuición

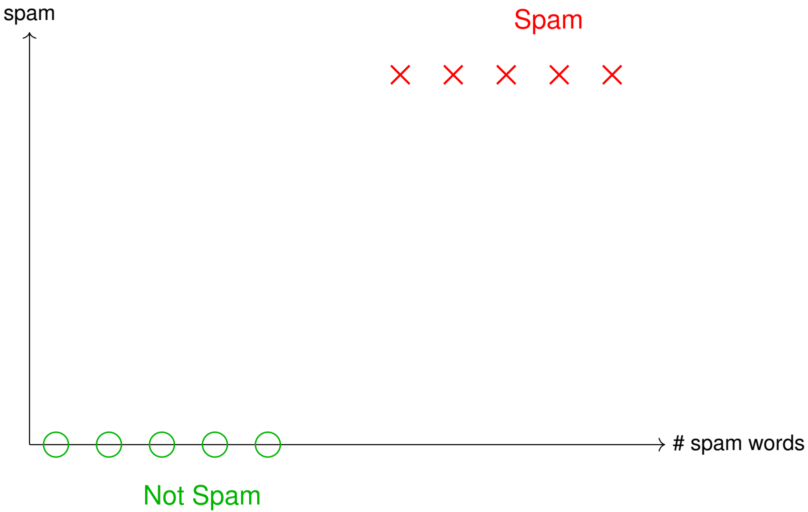


(*) Intuición adaptada de Andrew Ng

Intuición

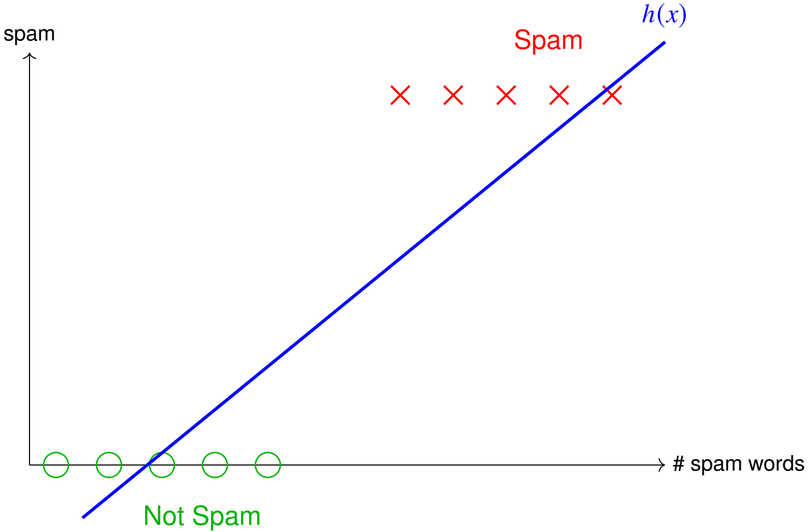
(*) Intuición adaptada de Andrew Ng

Intuición



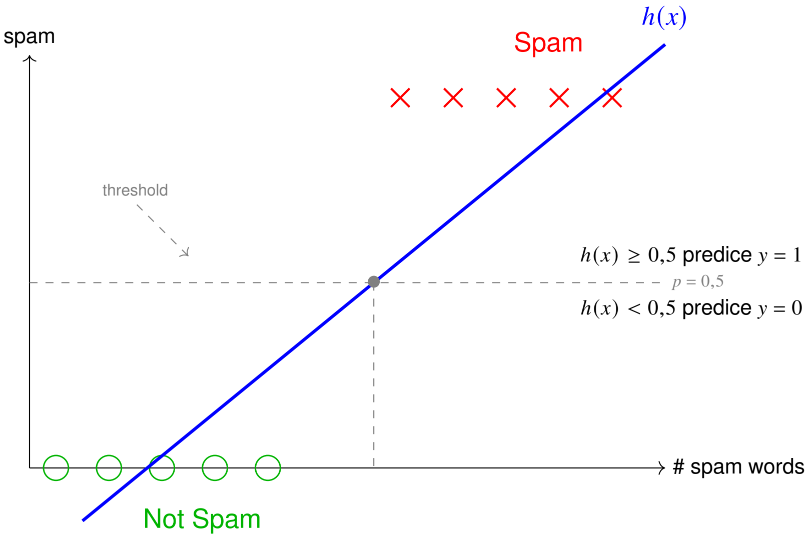
(*) Intuición adaptada de Andrew Ng

Intuición



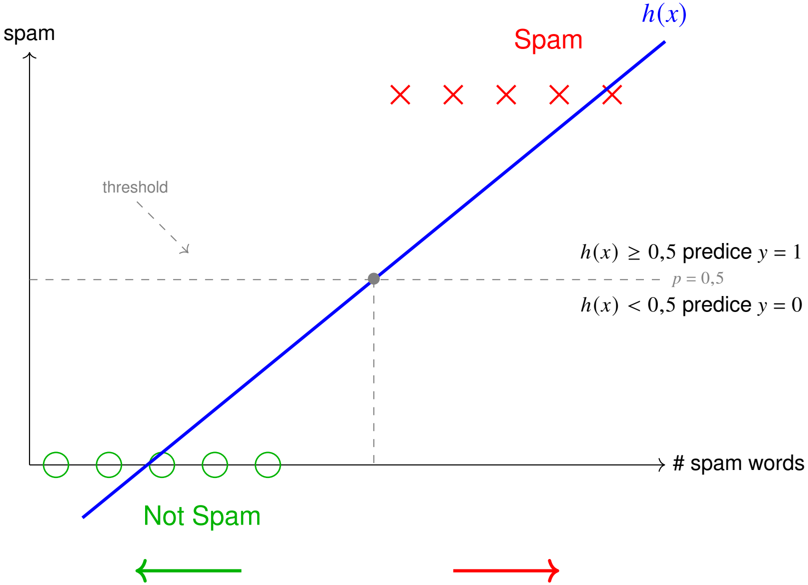
(*) Intuición adaptada de Andrew Ng

Intuición



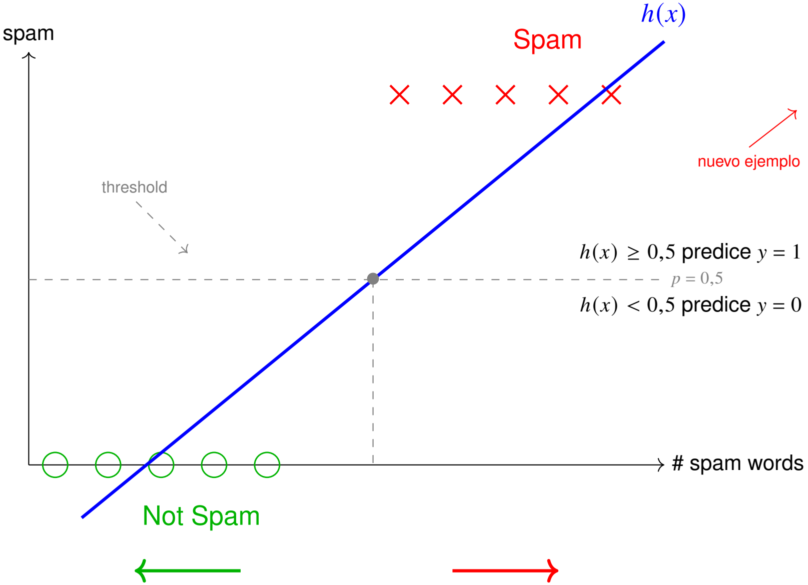
(*) Intuición adaptada de Andrew Ng

Intuición



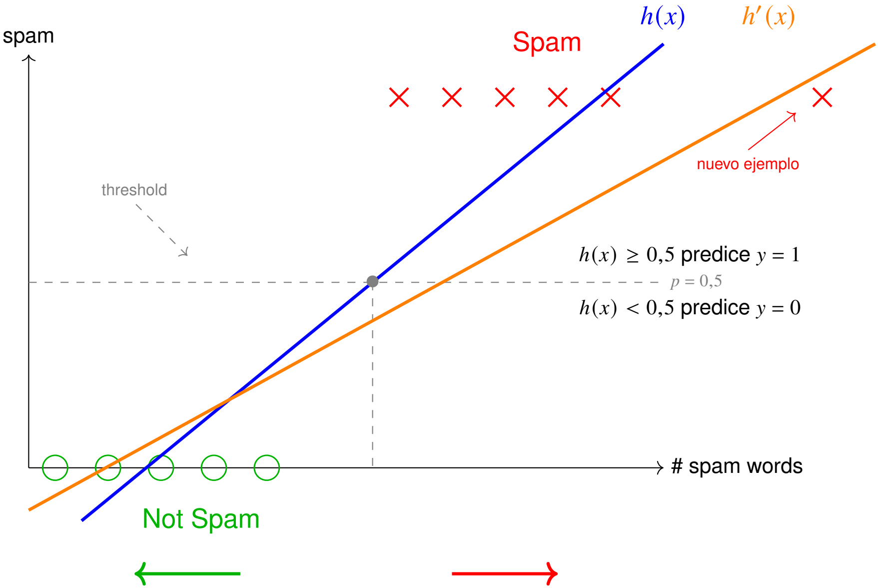
(*) Intuición adaptada de Andrew Ng

Intuición



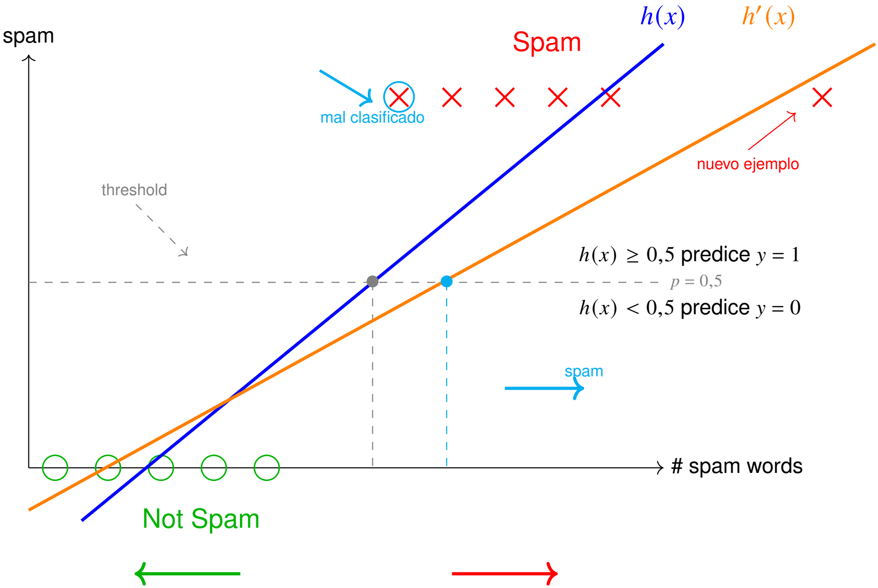
(*) Intuición adaptada de Andrew Ng

Intuición



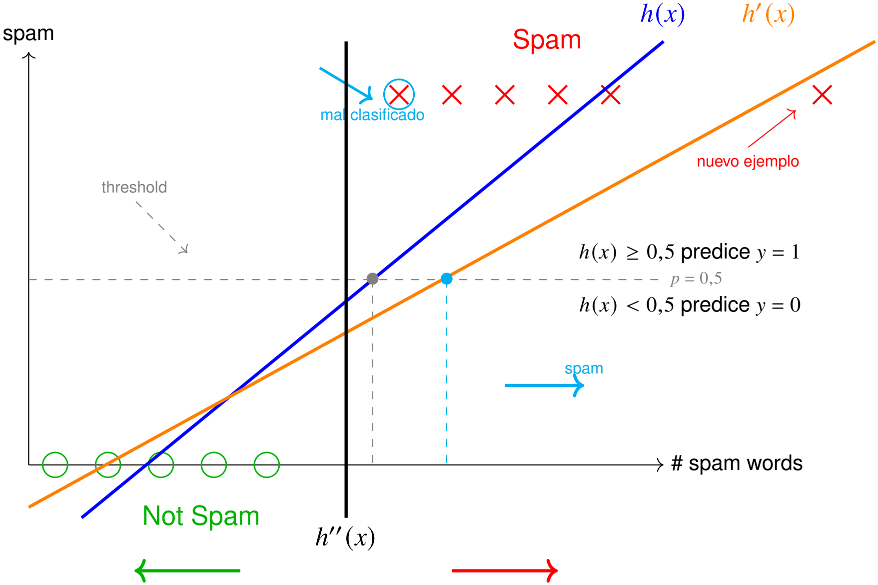
(*) Intuición adaptada de Andrew Ng

Intuición



(*) Intuición adaptada de Andrew Ng

Intuición



(*) Intuición adaptada de Andrew Ng

Regresión Logística

En la regresión logística,

$$0 \leq h(x) \leq 1$$

Regresión Logística

En la regresión logística,

$$0 \leq h(x) \leq 1$$

Hipótesis:

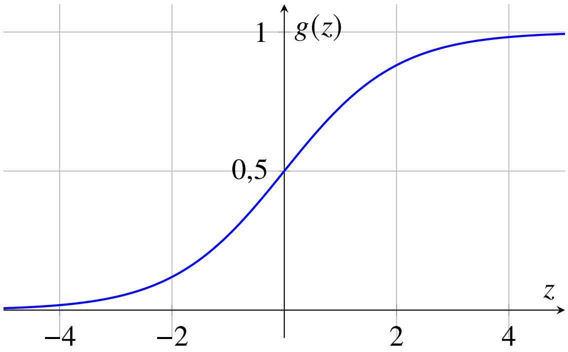
$$h(x) = g(\theta^\top x) = \frac{1}{1 + e^{-\theta^\top x}}$$

entonces si $z = \theta^\top x$, tenemos

$$g(z) = \frac{1}{1 + e^{-z}}$$

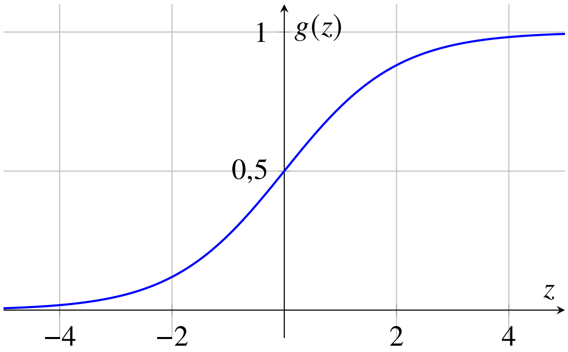
es llamada la función logística (o sigmoide)

Función Logística



- $g(z)$ tiene a 1 cuando $z \to \infty$
- $g(z)$ tiene a 0 cuando $z \to -\infty$
- $g(z)$ (y $h(x)$) están siempre entre 0 y 1

Función Logística

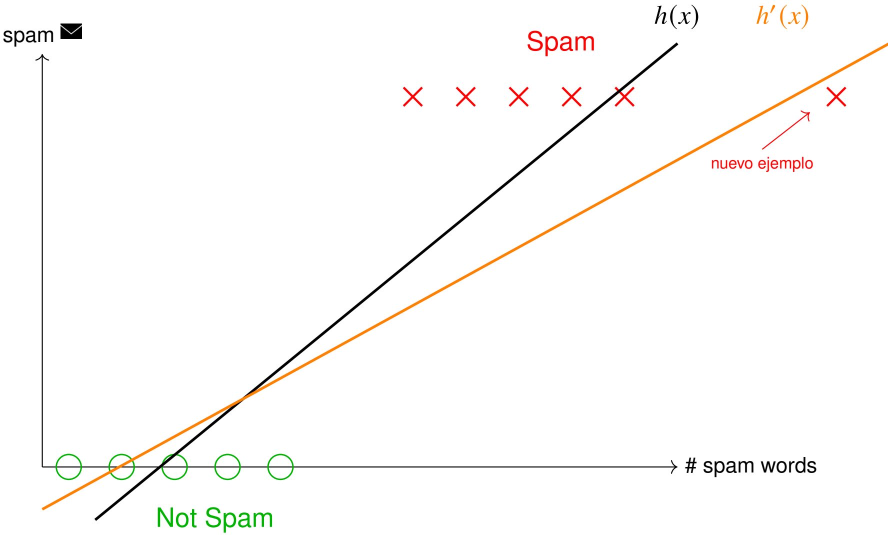


- $g(z)$ tiene a 1 cuando $z \to \infty$
- $g(z)$ tiene a 0 cuando $z \to -\infty$
- $g(z)$ (y $h(x)$) están siempre entre 0 y 1

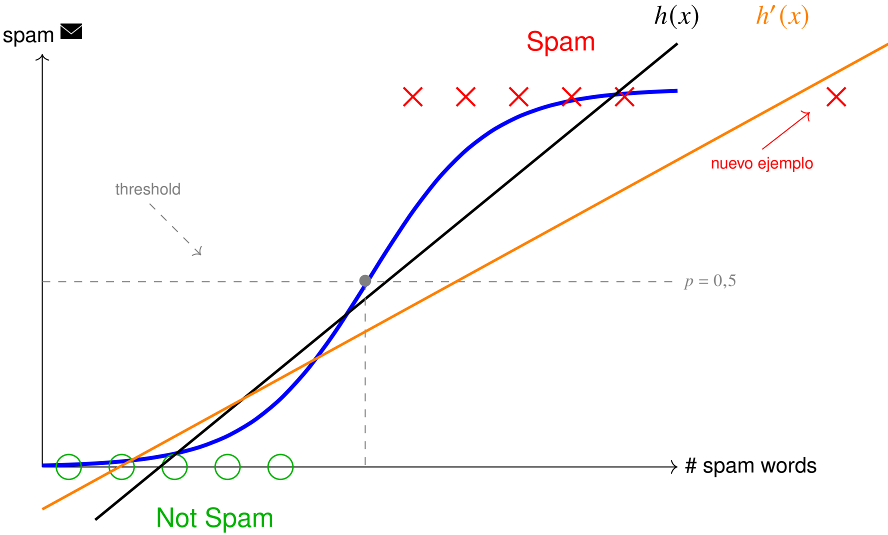
$h(x)$ es la **probabilidad** de $y = 1$ dada una input x parametrizado por θ

$$h(x) = P(y = 1|x; \theta)$$

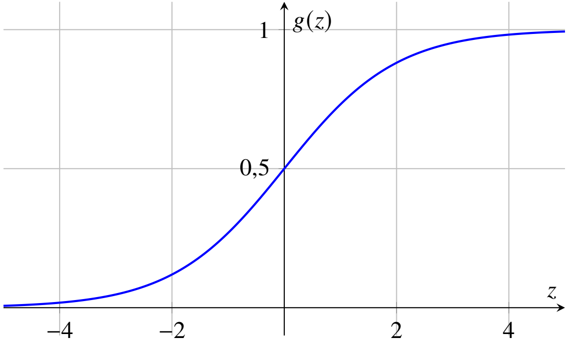
Spam - Revisemos intuición



Spam - Revisemos intuición



Supongamos que predecimos $y = 1$ si $h(x) \geq 0{,}5$, entonces $g(z) \geq 0{,}5$ cuando $z \geq 0$ (como muestra la siguiente figura)



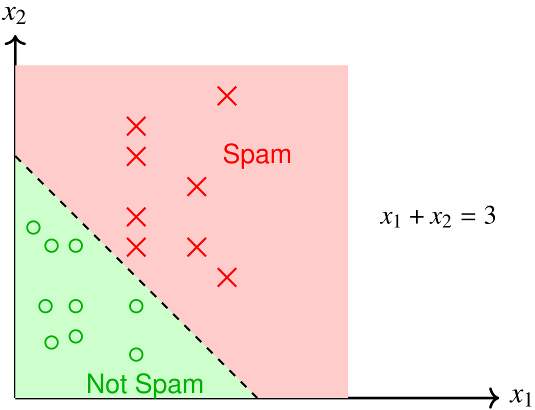
Limite de decisión (*Decision Boundary*) - cont'd

$y = 1 \rightarrow$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \geq 0$$

si $\theta_0 = -3, \theta_1 = \theta_2 = 1$

$$-3 + x_1 + x_2 \geq 0 \rightarrow x_1 + x_2 \geq 3$$



Función de costo

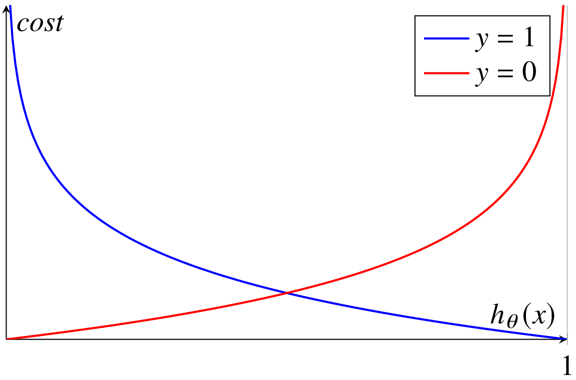
Función de costo:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y)$$

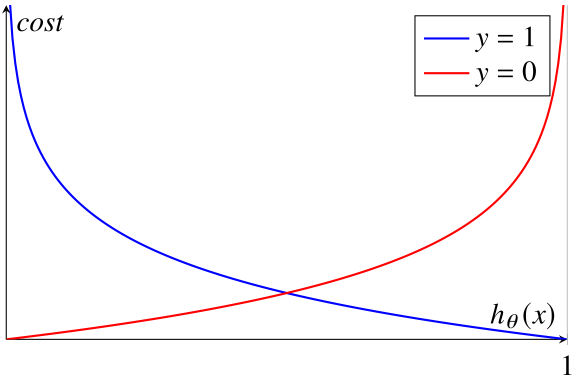
donde

$$cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right), & \text{si } y = 1 \\ -\log\left(1 - h_\theta(x)\right), & \text{si } y = 0 \end{cases}$$

Función de costo - interpretación

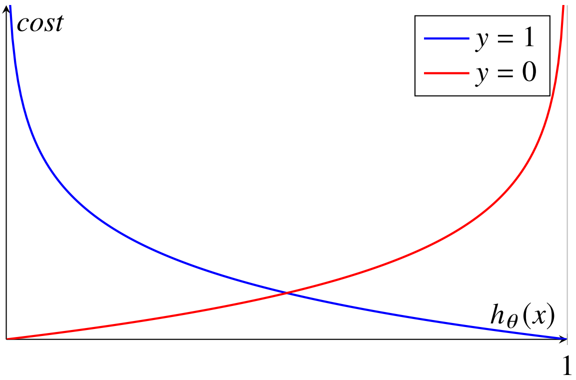


Función de costo - interpretación



- el costo es 0 si $y = 1$, y penaliza si predice $y = 0$ cuando $y = 1$ ($cost \rightarrow \infty$)

Función de costo - interpretación



- el costo es 0 si $y = 1$, y penaliza si predice $y = 0$ cuando $y = 1$ ($cost \to \infty$)
- el costo es 0 si $y = 0$

Método del gradiente

Objetivo:

$$min_{\theta} J(\theta)$$

Método del gradiente

Objetivo:

$$min_\theta J(\theta)$$

Función de costo:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y)$$

donde

$$cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right), & \text{si } y = 1 \\ -\log\left(1 - h_\theta(x)\right), & \text{si } y = 0 \end{cases}$$

$$J(\theta_0, \theta_1) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i))\right]$$

Método del gradiente

Objetivo:

$$min_\theta J(\theta)$$

Función de costo:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y)$$

donde

$$cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right), & \text{si } y = 1 \\ -\log\left(1 - h_\theta(x)\right), & \text{si } y = 0 \end{cases}$$

$$J(\theta_0, \theta_1) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial\theta_1} J(\theta_j)$$

Método del gradiente

Objetivo:

$$min_\theta J(\theta)$$

Función de costo:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y)$$

donde

$$cost(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right), & \text{si } y = 1 \\ -\log\left(1 - h_\theta(x)\right), & \text{si } y = 0 \end{cases}$$

$$J(\theta_0, \theta_1) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

$$\theta_j := \theta_j - \alpha\,\frac{\partial}{\partial\theta_1} J(\theta_j)$$

😎 mismo algoritmo que el gradiente para regresión lineal!

Algoritmo para Regresión Logística

Inicializar parámetros:
$\theta_0 \leftarrow 0,\ \theta_1 \leftarrow 0$ # parámetros iniciales
Definir hiperparámetros:
learning_rate $\leftarrow \alpha$, max_iter, ϵ
Cargar datos:
$X = [x_1, \ldots, x_m],\ Y = [y_1, \ldots, y_m],\ m \leftarrow |X|$
Definir hipótesis logística:
$h_\theta(x) = \sigma(\theta_0 + \theta_1 x)$, donde $\sigma(z) = \frac{1}{1+e^{-z}}$
Definir función de costo (log-loss):

$$J(\theta_0, \theta_1) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

for $iter = 1$ to max_iter **do**
    Calcular gradientes:
    $grad_0 = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i\right)$
    $grad_1 = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x_i) - y_i\right) \cdot x_i$
    Actualizar parámetros:
    $\theta_0 \leftarrow \theta_0 - learning\_rate \cdot grad_0$
    $\theta_1 \leftarrow \theta_1 - learning\_rate \cdot grad_1$
    Calcular costo actual: $cost = J(\theta_0, \theta_1)$
    if $|grad_0| < \epsilon$ **and** $|grad_1| < \epsilon$ **then**
        break
    end if
end for
Retornar: θ_0, θ_1, costo final

Consideraciones finales

- (además del método del gradiente) Otras técnicas de optimización avanzadas pueden ser *Conjugate gradient*, `BFGS` y `L-BFGS`

Consideraciones finales

- (además del método del gradiente) Otras técnicas de optimización avanzadas pueden ser *Conjugate gradient*, `BFGS` y `L-BFGS`
- El enfoque puede ser extendido para clasificación multi-clase, i.e. la variable de respuesta y puede tomar cualquier valor $k \in 0, 1, 2, ..., k$

Consideraciones finales

- (además del método del gradiente) Otras técnicas de optimización avanzadas pueden ser *Conjugate gradient*, `BFGS` y `L-BFGS`
- El enfoque puede ser extendido para clasificación multi-clase, i.e. la variable de respuesta y puede tomar cualquier valor $k \in 0, 1, 2, ..., k$
  - Clasificar emails en `Trabajo, Family, Work`

Consideraciones finales

- (además del método del gradiente) Otras técnicas de optimización avanzadas pueden ser *Conjugate gradient*, `BFGS` y `L-BFGS`
- El enfoque puede ser extendido para clasificación multi-clase, i.e. la variable de respuesta y puede tomar cualquier valor $k \in 0, 1, 2, ..., k$
  - Clasificar emails en `Trabajo, Family, Work`
  - Se denomina Regresión *Softmax*
  - En librerías tales como `scikit-learn` [**?**],la regresión multi-clase es implementada de manera similar definiendo el parámetro `multi_class` como `multinomial`

Próximas clases

- Máquinas de soporte vectorial
- Redes Neuronales
- Aprendizaje no supervisado

¡Gracias!

Créditos

- Íconos de https://icons8.com/icons/set/ai-machine-learning-icon/https://www.flaticon.com/free-icons/machine-learning

Bibliografía y material de referencia

- Harrington, Peter. Machine learning in action. *Simon and Schuster*, 2012.
- Alpaydin, Ethem. Introduction to machine learning. 3era Edición *MIT Press*, 2020.
- Brett Lantz. Machine Learning with R. *Packt Publishing*, 1997.
- Tom M. Mitchell. Machine Learning. *WCB McGraw-Hill*, 1997.
- Witten I., Frank E., Hall, M., Pal C.. Data Mining: Practical Machine Learning Tools and Techniques. 4th Edition*WMorgan Kaufmann. Elsevier*, 2017.
- Michael A. Nielsen. Neural Networks and Deep Learning. 4th Edition *Determination Press*, 2015. http://neuralnetworksanddeeplearning.com

Bibliografía y material de referencia

- Afshine Amidi, Shervine Amidi. CS 229 — Machine Learning. https://stanford.edu/~shervine/teaching/cs-229/
- Andrew Ng. Stanford CS229 - Machine Learning Course. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Andrew Ng. Deep Learning AI. https://www.deeplearning.ai/resources/
- Kilian Weinberger. Machine Learning for Intelligent Systems. https://www.cs.cornell.edu/courses/cs4780/2018fa/syllabus/