# Introducción Aprendizaje de Máquinas

Visión por Computador II

# Contenido

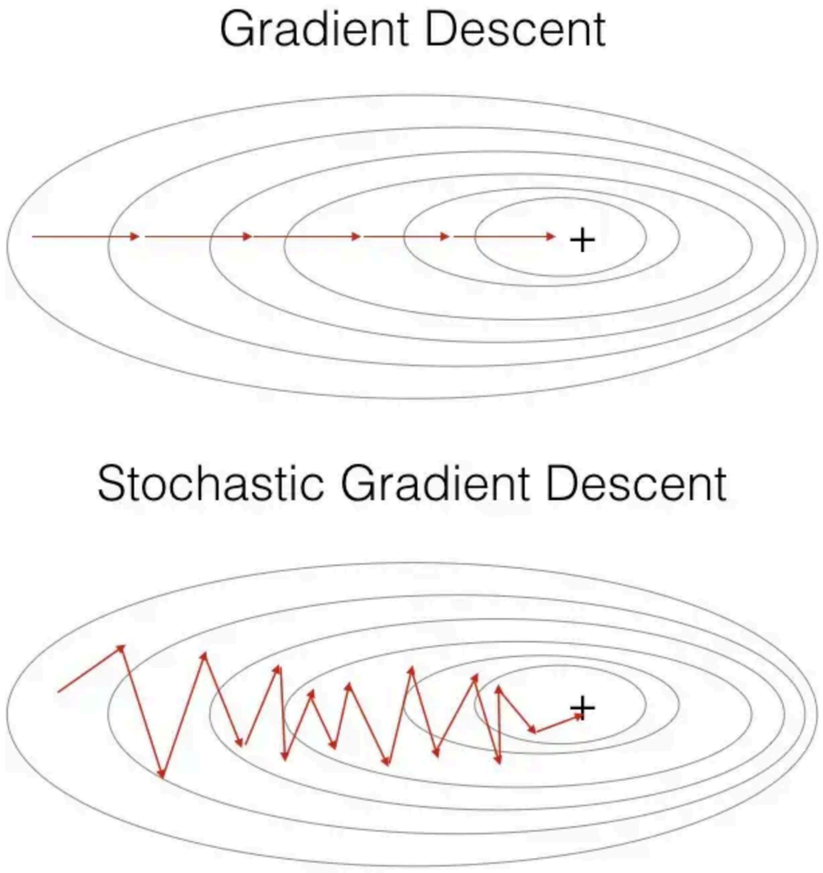
1. Stochastic Gradient Descent
2. Generalización
3. Regresión logística
4. Ejercicio

# Stochastic Gradient Descent

- Casi todo el aprendizaje profundo se basa en un algoritmo muy importante: el descenso de G**radiente Estocástico (SGD).**

- Para una buena generalización se necesitan grandes conjuntos de entrenamiento, pero también son más costosos desde el punto de vista informático.

- Tomar una muestra aleatoria de observaciones en cada iteración (Mini-batch o Batch)




*https://medium.datadriveninvestor.com/batch-vs-mini-batch-vs-stochastic-gradient-descent-with-code-examples-cd8232174e14*

# Stochastic Gradient Descent

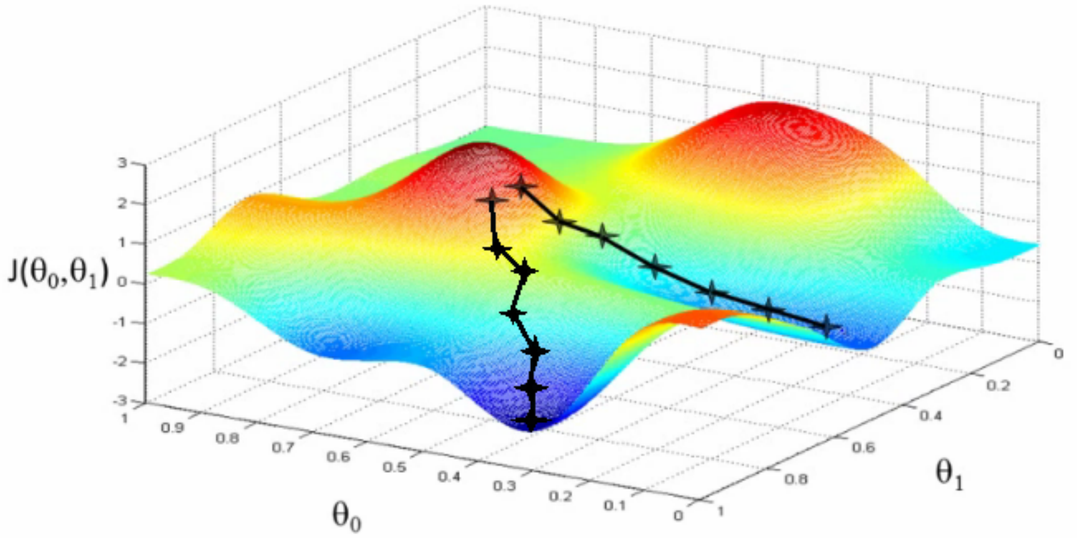
**En la práctica**

A. Se reorganiza de forma aleatoria el dataset cada Epoch

B. Dividimos el dataset en baches

C. Se pasan actualizan los parámetros del modelo con cada batch

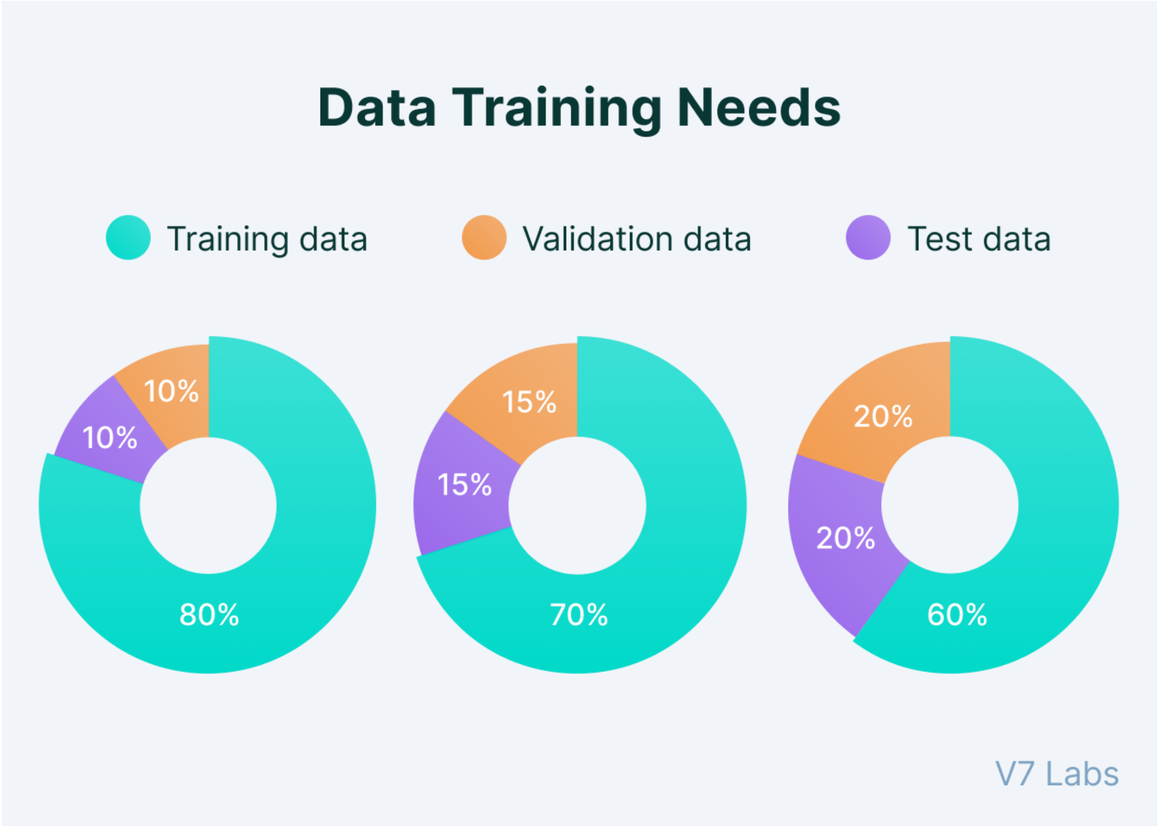
D. Repetimos




*https://metamug.com/article/groovy/stochastic-gradient-descent-tutorial-code-by-andrew-ng.html*

# Generalización

- El principal reto de ML es que nuestro algoritmo **funcione bien con entradas nuevas y desconocidas**, no sólo con las que se ha entrenado nuestro modelo.
- Esta capacidad se llama *generalización*
- Calculamos la medida de error en el conjunto de entrenamiento
- Denominada error de entrenamiento
- Reducimos este error de entrenamiento




*https://www.v7labs.com/blog/train-validation-test-set*

# Generalización

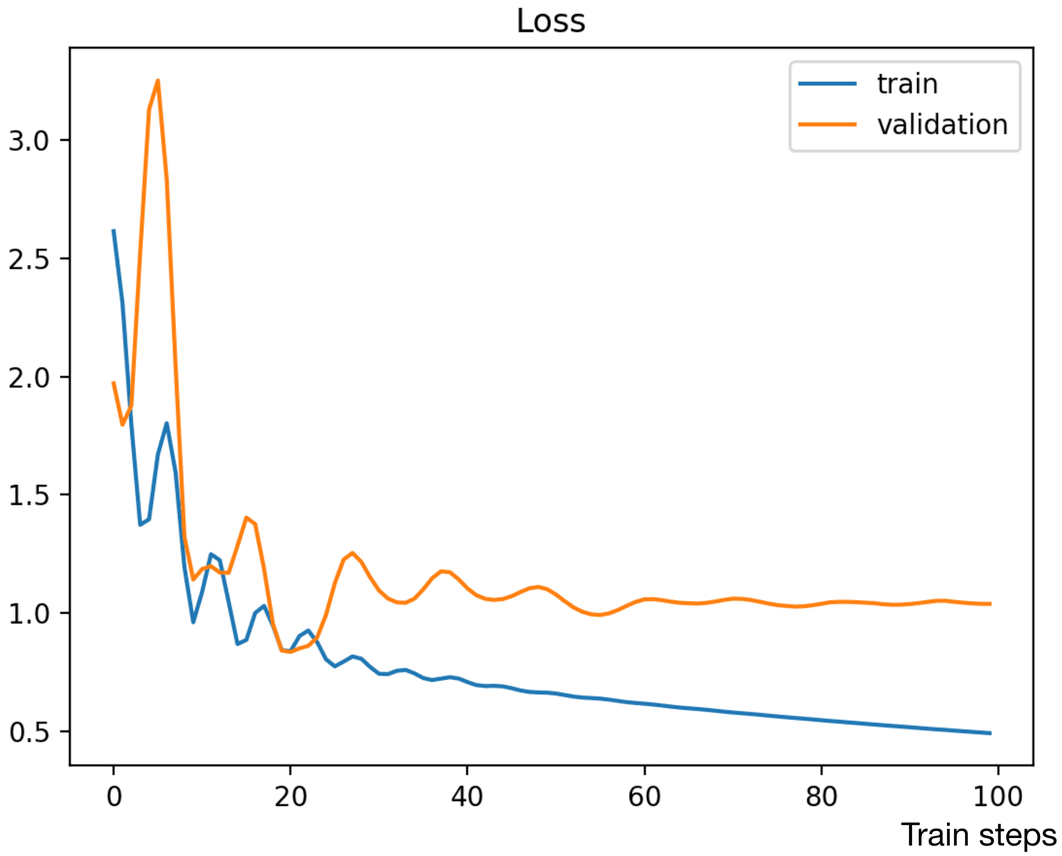
- La diferencia entre ML y optimización es que también queremos que el error de generalización (error de prueba) sea bajo.
- El error de generalización es el valor esperado del error en una nueva entrada
- El error de generalización es medido con el dataset de pruebas (test dataset)

$$e_{test} = \frac{1}{m^{(test)}} \sum_{i}^{m} \|X^{(test)}\theta - y^{(test)}\|_2^2$$

# Entrenamiento de los modelos

- Se asume que los dos conjuntos (train, test) son independientes y son muestras de una misma distribución.
- Los factores que determinan qué tan bien un modelo se desempeñará son:
  - Disminuir el error de entrenamiento
  - Disminuir la diferencia entre los errores de entrenamiento y pruebas

**Dos retos centrales: *underfitting* y *overfitting*.**

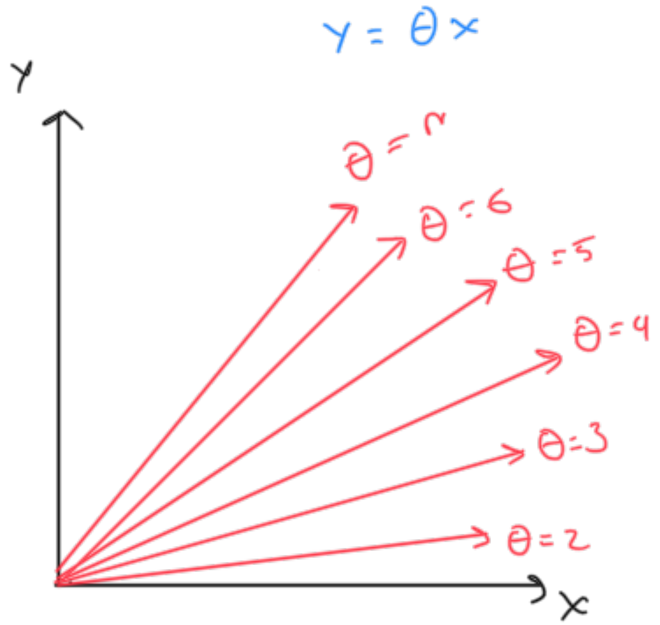



**Gráfica del error del modelo durante entrenamiento**

*https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/*

# Capacidad del modelo

- La diferencia entre el error de prueba y de entrenamiento está relacionada con la capacidad del modelo
- La capacidad representa el número de funciones que un modelo puede seleccionar como posible solución




**En el caso lineal se refiere a todas las posibles rectas que puede elegir el modelo**
**Como representación lineal el conjunto de datos**

# Capacidad del modelo

Por ejemplo, la regresión lineal tiene como espacio de hipótesis el conjunto de todas las funciones lineales de su entrada

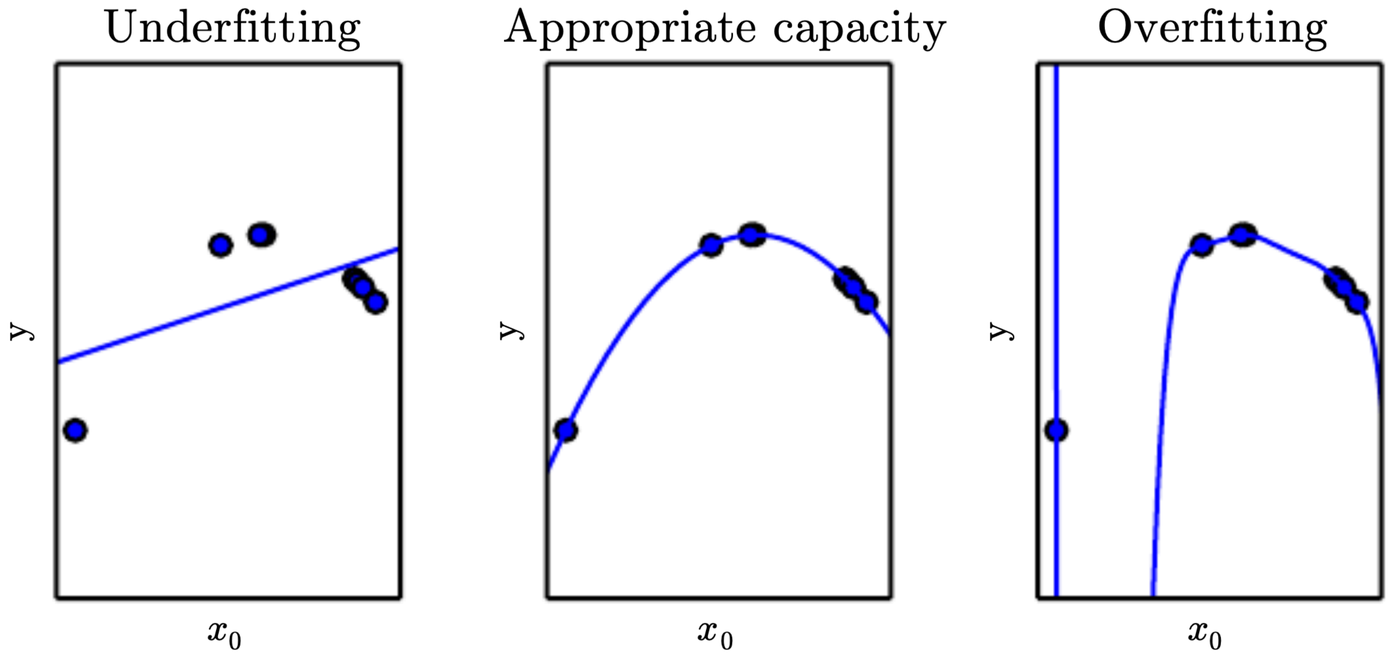
$$\hat{y} = b + xw$$

Si $x^2$ como otra característica, podemos aprender un modelo que sea cuadrático

$$\hat{y} = b + w_1 x + w_2 x^2$$

Podemos seguir añadiendo más potencias de x como características adicionales

$$\hat{y} = \sum_{i=1}^{n} b + w_i x^i + \ldots + w_n x^n$$




**Podemos controlar si un modelo tiene más probabilidades de overfitting o underfitting alterando su capacidad**

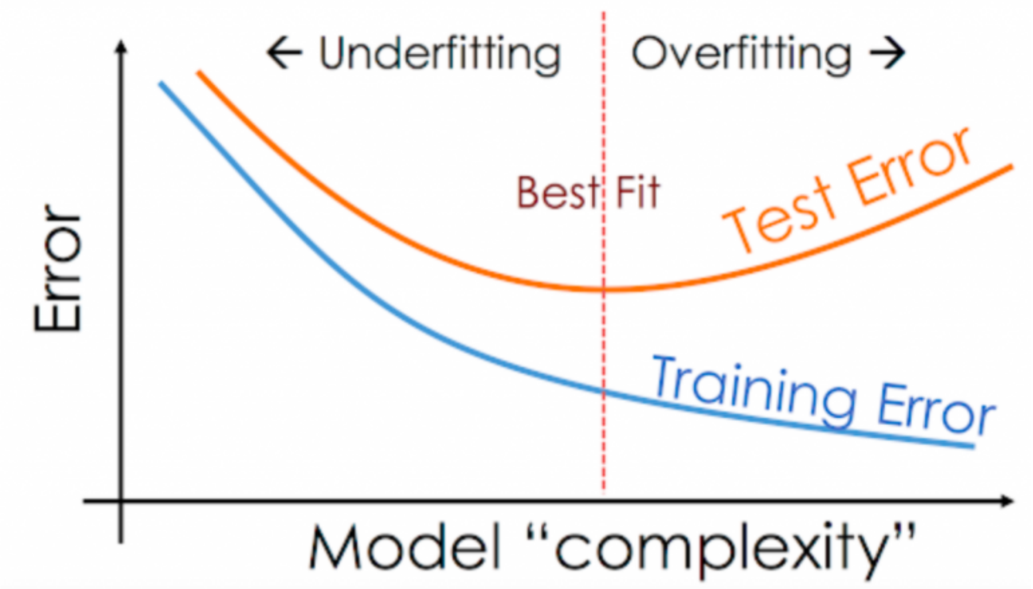
*Deep learning by Ian Goodfellow*

# Capacidad del modelo

**Underfit:** Capacidad insuficiente son incapaces de resolver tareas complejas.

**Overfit:** Capacidad superior a la requerida por el problema

- Hay varias formas de cambiar la capacidad de un modelo
  - **Capacidad de representación:** qué familia de funciones puede elegir el algoritmo de aprendizaje
  - **Capacidad effectiva:** capacidad real del modelo influenciada por el proceso de optimización.




*https://vitalflux.com/overfitting-underfitting-concepts-interview-questions/*

# *Overfitting* y *Underfitting*

## Underfitting

Ocurre cuando el modelo no es capaz de disminuir el error de entrenamiento

## Overfitting

Ocurre cuando la diferencia entre el error en el dataset de pruebas y de entrenamiento es muy algo



*https://histalk2.com/2018/12/05/machine-learning-primer-for-clinicians-part-7/*

# Addressing Overfitting

1. Reducir el número de características

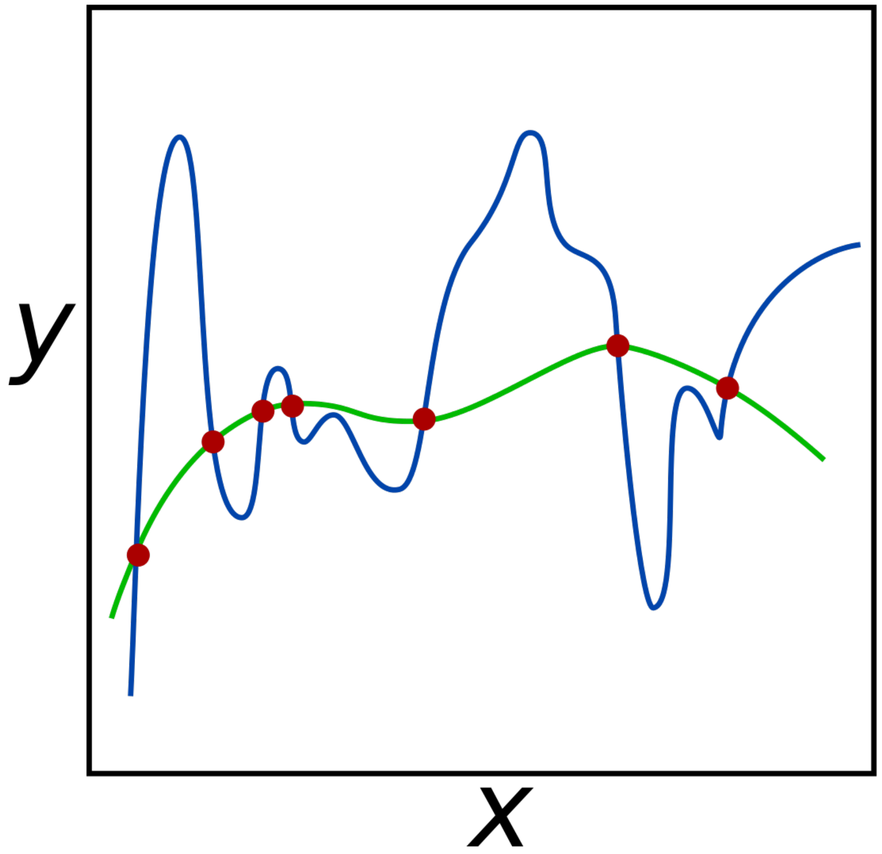
   “Manualmente” seleccionar cuales características mantener

2. Regularización :

   Mantener todas las características pero reducir la magnitud de los valores de $\theta_j$

3. Otros:

   Mas adelante los vemos



# Regularización L2

**Generalidades**

- Penalizar valores altos de los pesos $\theta_i$
- Mantener los pesos con valores pequeños
- Tiende menos a O*verfitting*

**Agregar un término al costo de las funciones**

En Regresión linea:

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2 + \frac{\lambda}{2}\sum_{j}^{n}\theta_j^2$$

# Regularización L2

La derivada del término de regularización:

$$\frac{\partial}{\partial\theta} = \lambda\theta$$

En el Gradient Descent:

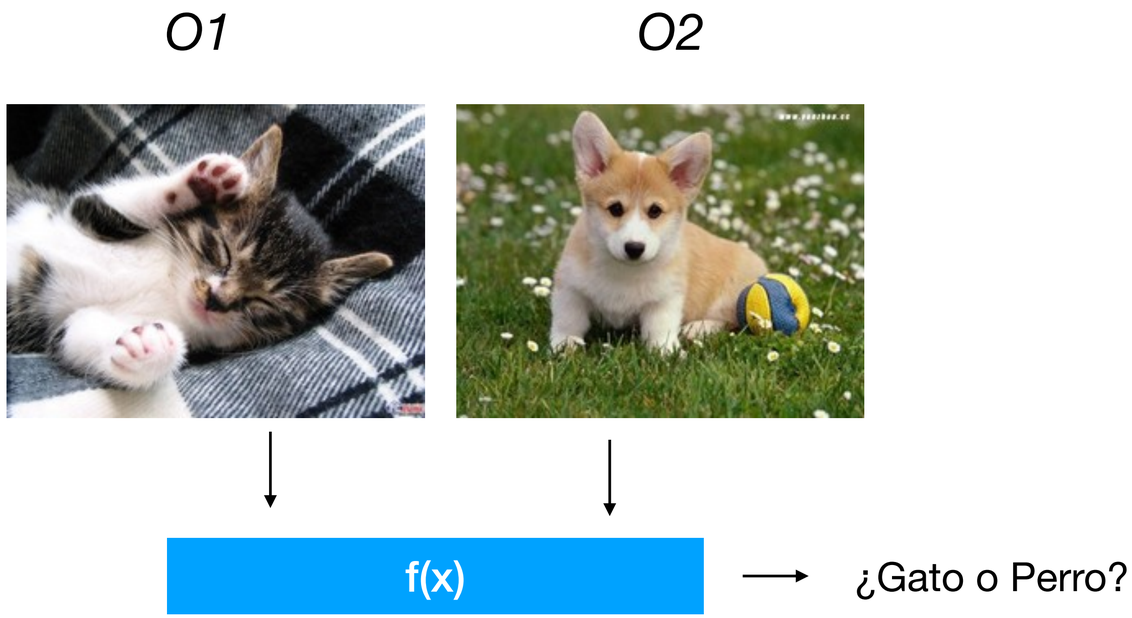
$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\hat{y}_i - y_i$$

El *bias* no contribuyen a la curvatura del modelo, por lo que no suele tener mucho sentido regularizarlos

$$\theta_j := \theta_j - \frac{\alpha}{m}\left[\sum_{i=1}^{m}(\hat{y}_i - y_i)x_j + \lambda\theta_j\right]$$

# Regresión logística

- Anteriormente aprendimos a predecir cantidades de valor continuo como una función lineal de valores de entrada
- Predecir valores discretos como:
  - Predecir si un correo es spam (1) o (0)
  - Predecir si un tumor es benigno (1) o no (0)
- Problema de clasificación binaria
- Aplica en general para clasificación múltiple

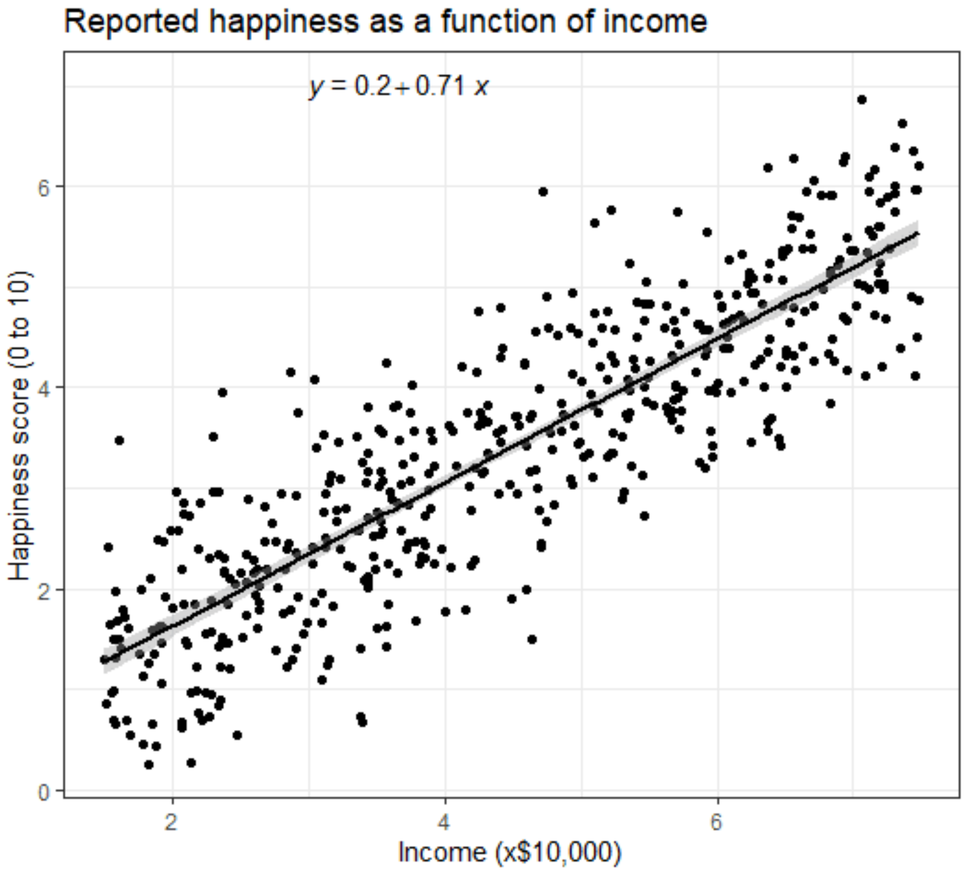



**Clasificación: se pide al programa que especifique a cuál de *k* categorías pertenece una entrada.**

# Notación

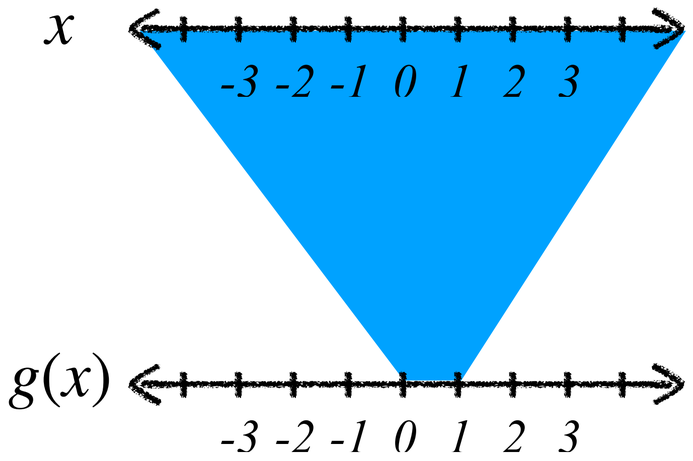
Por razones históricas la función de predicción es llamada hypothesis $h_\theta(x)$ con los parámetros $\theta$ de la observación $x$

$$\hat{y} = h_\theta(x) = \theta^T x$$



# Función de activación

- Podríamos usar la regresión lineal para predecir los valores
  - Tiene bajo desempeño
  - No tiene sentido que $h_\theta(x)$ tome valores menores que 0 o mayores que 1, ya que $y \in \{0,1\}$.
- Pasaremos el resultado de la función lineal por otra función $g(x)$ que la comprima el resultado de $\theta^T x$ en un rango $[0,1]$
- Interpretamos $h_\theta(x)$ como una probabilidad



# Sigmoid function

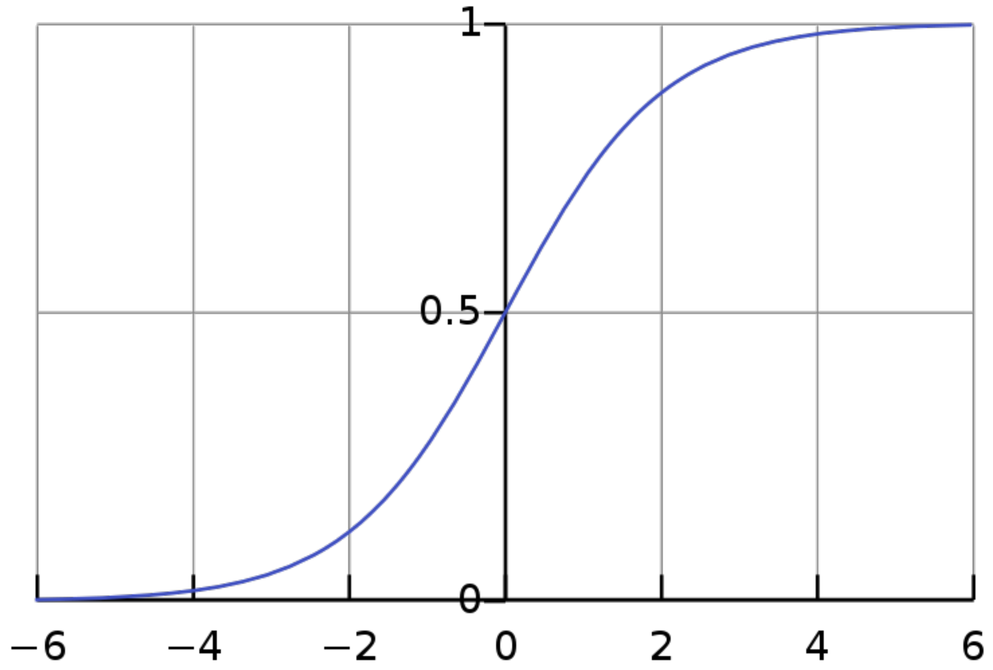
Cambiemos nuestra función de predicción por:

$$h_\theta(x) = g(\theta^T x)$$

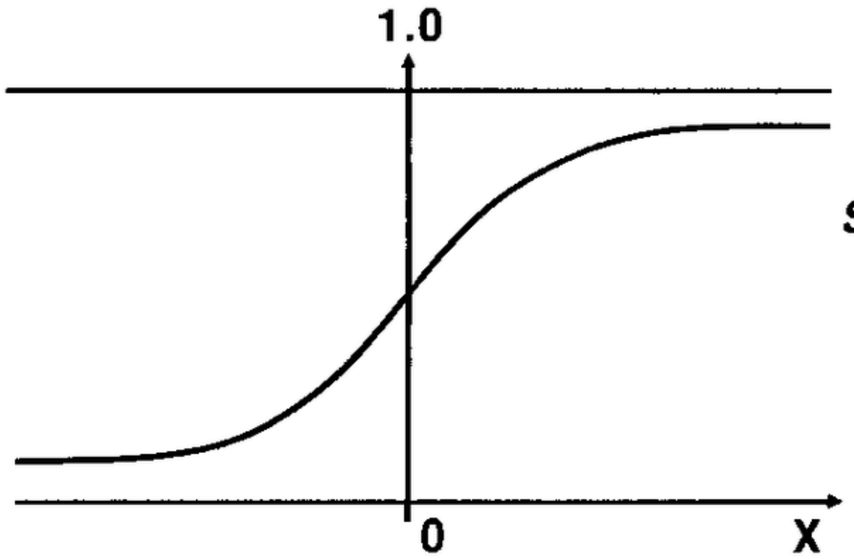
En donde

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

$g(z)$ Es llamada función sigmoid o logística



# Sigmoid function




*Sigmoid*

$$f(x) = \frac{1}{1+e^{-x}}$$

**Función sigmoid** $g(z)$

- Tiende a 1 a medida que $z \to \infty$
- Tiende a 0 a medida que $z \to -\infty$
- Siempre acotada entre 0 y 1.
- Se pueden utilizar otras funciones que aumentan suavemente de $[0,1]$

*1993 - Artificial Neural Networks in Multivariate Calibration*

# ¿Cómo sería la predicción en regresión logística?

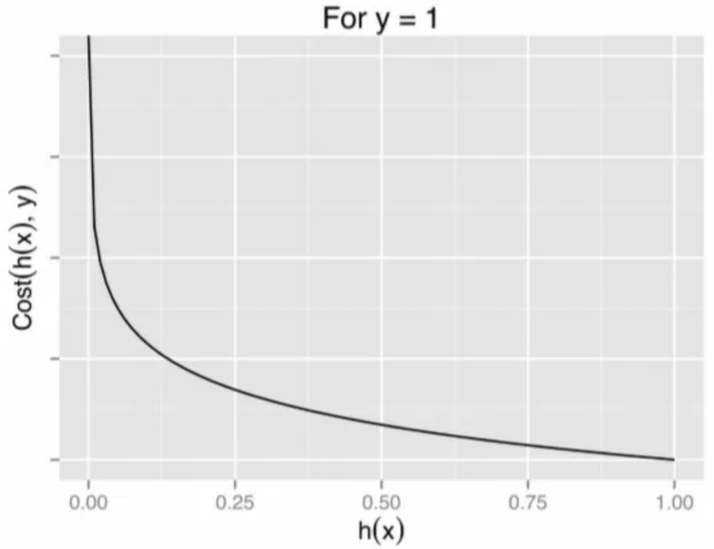
# Función de costo

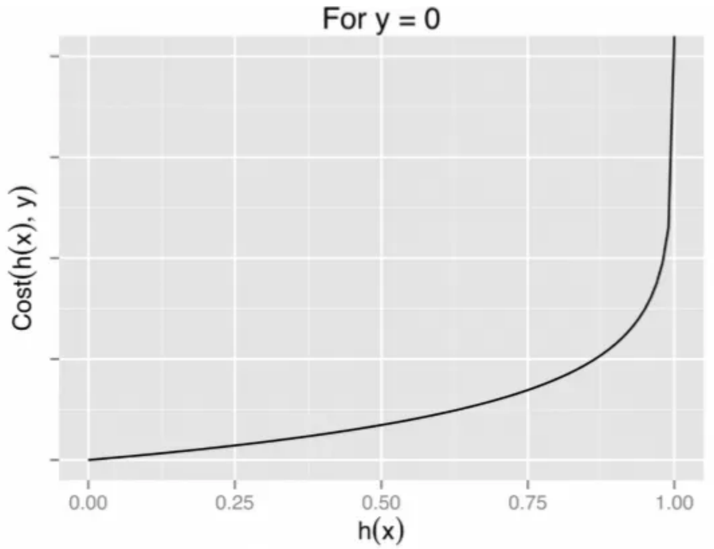
**Costo de la regresión logística:**

$$\ell(\theta) = \begin{cases} -\log(\hat{p}) & \text{si } y = 1 \\ -\log(1-\hat{p}) & \text{si } y = 0 \end{cases}$$

**Casos**:

- $y = 1 \begin{cases} \hat{y} = 1 \rightarrow \ell(\theta) = 0 \\ \hat{y} = 0 \rightarrow \ell(\theta) = \infty \end{cases}$
- $y = 0 \begin{cases} \hat{y} = 0 \rightarrow \ell(\theta) = 0 \\ \hat{y} = 1 \rightarrow \ell(\theta) = \infty \end{cases}$



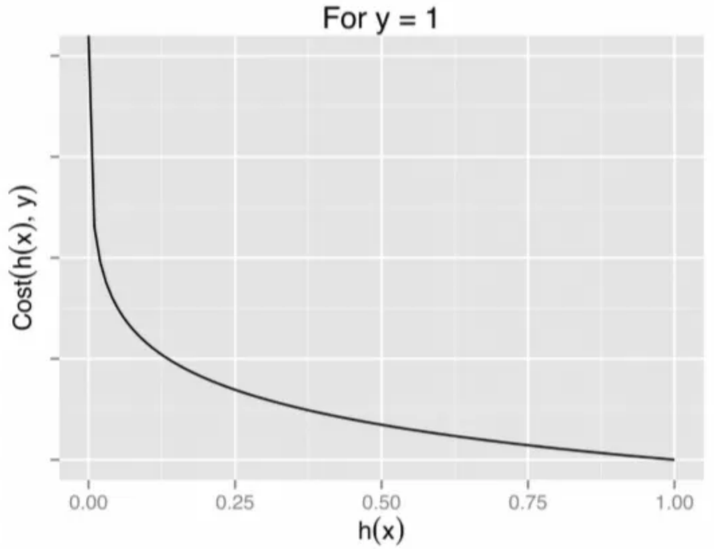



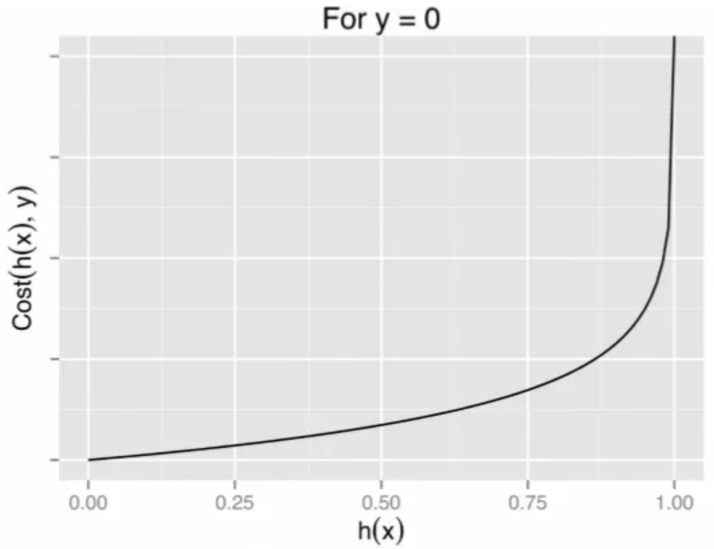

*https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148*

# Función de costo

Costo en términos de minimizar

$$J(\theta) = -\frac{1}{m}\sum_{1=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)})\log(1 - h(x^{(i)}))$$




*Más detalle ver Capítulo 2 de: https://cs229.stanford.edu/notes2022fall/main_notes.pdf*

# Optimización de función de costo

Gradientes de la función de costo

$$\nabla J(\theta) = y \log(\sigma(z))' + (1 - y)\log(1 - \sigma(z))'$$

Derivadas parciales:

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)}$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} (h_\theta(x^{(i)}) - y^{(i)})$$

Donde

$$h_\theta(x) = \sigma(\theta^T x)$$

*Más detalle ver Capítulo 2 de: https://cs229.stanford.edu/notes2022fall/main_notes.pdf*

# Gradient descent

Costo en términos de minimizar:

$$J(\theta) = -\frac{1}{m}\sum_{1=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 + y^{(i)})\log(1 - h(x^{(i)}))$$

Actualizar $\theta_j$ con cada paso de entrenamiento ($\alpha$: Learning Rate):

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

En donde:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)} - y^{(i)})x_j^{(i)}$$

# Ejercicio

Implementar las funciones necesarias para realizar un modelo de clasificación de dígitos (0 y 1)