



UNIVERSIDAD DE CHILE

# Minería de Datos

Welcome to the Machine Learning class

---

Valentin Barriere

Universidad de Chile – DCC

CC5205, Otoño 2024

# **Aprendizaje supervisado – Modelos Lineales**

# Recordatorios

---

# Outline : Recordatorios

## Recordatorios

Resumen

Optimización

Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

API : Scikit-learn

TP: Regresión Lineal

# Outline : Resumen

## Recordatorios

Resumen

Optimización

Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

API : Scikit-learn

TP: Regresión Lineal

# En resumen

- I Tener datos etiquetados
- II Extraer los descriptores = transformar documentos en vectores
- III Crear un modelo matemático  $f_\theta$
- VI Implementar una función de costo (error)  $\ell$  a minimizar
- V Encontrar los parámetros  $\theta^*$  de manera que  $\ell(f_{\theta^*}(\mathbf{X}_i), Y_i)$  sea pequeño
- VI Probar  $f_{\theta^*}$  en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

## III Crear un modelo matemático $f_\theta$

## VI Implementar una función de costo (error) $\ell$ a minimizar

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(\mathbf{X}_i), Y_i)$ sea pequeño

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

- $X$  es un vector de observaciones (por ejemplo: edad, sexo, salario)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## III Crear un modelo matemático $f_\theta$

## VI Implementar una función de costo (error) $\ell$ a minimizar

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

- $X$  es un vector de observaciones (por ejemplo: edad, sexo, salario)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## III Crear un modelo matemático $f_\theta$

- Modelo  $f_\theta$  tal que  $f_\theta(X)$  esté cerca de  $Y$  (para regresión)
- $\theta$  es el conjunto de parámetros del modelo matemático

## VI Implementar una función de costo (error) $\ell$ a minimizar

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

- $X$  es un vector de observaciones (por ejemplo: edad, sexo, salario)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## III Crear un modelo matemático $f_\theta$

- Modelo  $f_\theta$  tal que  $f_\theta(X)$  esté cerca de  $Y$  (para regresión)
- $\theta$  es el conjunto de parámetros del modelo matemático

## VI Implementar una función de costo (error) $\ell$ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

- $X$  es un vector de observaciones (por ejemplo: edad, sexo, salario)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## III Crear un modelo matemático $f_\theta$

- Modelo  $f_\theta$  tal que  $f_\theta(X)$  esté cerca de  $Y$  (para regresión)
- $\theta$  es el conjunto de parámetros del modelo matemático

## VI Implementar una función de costo (error) $\ell$ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

- $\theta^* = \arg \min_{\theta} \sum_i \ell(f_\theta(X_i), Y_i)$

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# En resumen

## I Tener datos etiquetados

- Conjunto de datos de tamaño  $n$ ,  $\mathcal{D}_n = \{(Doc_i, Y_i), i = 1..n\}$
- Doc es una muestra (por ejemplo: una persona)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## II Extraer los descriptores = transformar documentos en vectores

- $X$  es un vector de observaciones (por ejemplo: edad, sexo, salario)
- $Y$  son las etiquetas (por ejemplo: monto del préstamo concedido)

## III Crear un modelo matemático $f_\theta$

- Modelo  $f_\theta$  tal que  $f_\theta(X)$  esté cerca de  $Y$  (para regresión)
- $\theta$  es el conjunto de parámetros del modelo matemático

## VI Implementar una función de costo (error) $\ell$ a minimizar

- Cuanto más se equivoque el modelo, mayor será el costo
- En general, se desea tener un costo pequeño

## V Encontrar los parámetros $\theta^*$ de manera que $\ell(f_{\theta^*}(X_i), Y_i)$ sea pequeño

- $\theta^* = \arg \min_{\theta} \sum_i \ell(f_{\theta}(X_i), Y_i)$

## VI Probar $f_{\theta^*}$ en nuevos datos con una métrica de evaluación adecuada

# Outline : Optimización

## Recordatorios

Resumen

Optimización

Modelos de clasificación

Clasificador lineal

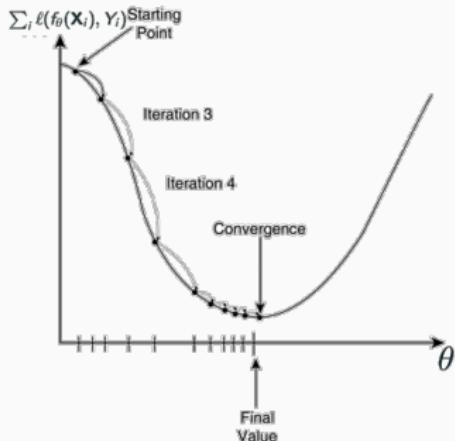
Regresión Lineal

Regresión Logística

API : Scikit-learn

TP: Regresión Lineal

# Optimización



## Optimización de la función de costo

- Sirve para converger al valor mínimo de la función de costo en el conjunto de datos de entrenamiento
- Mejor caso: rápido y preciso
- A menudo se hacen aproximaciones para ser más rápidos

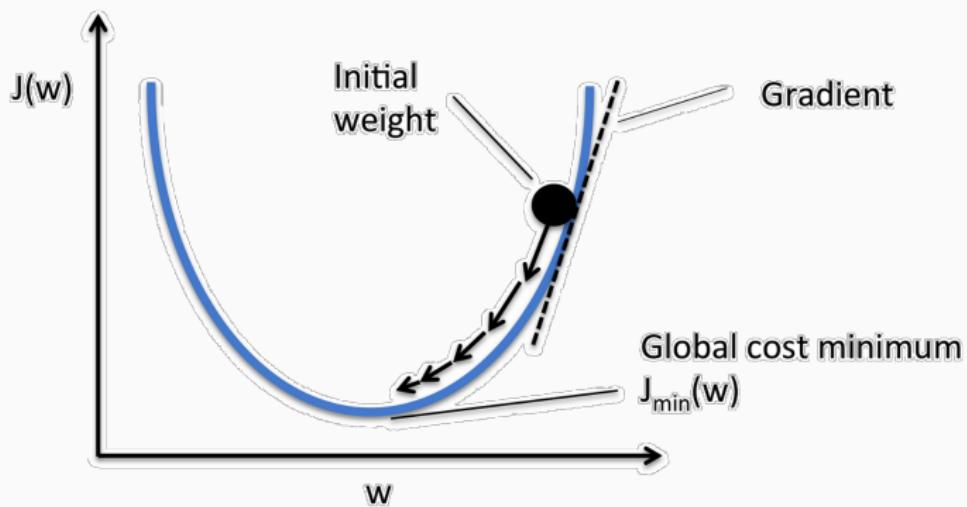
# Gradiente

Una función multivariante  $f(x)$  puede escribirse como una serie de Taylor:

$$f(x + \delta_x) = f(x) + \nabla_x f(x)\delta_x + \mathcal{O}(\|\delta_x^2\|)$$

## Definición

El gradiente de una función  $\nabla_x f(x) = (\frac{\partial f}{\partial_i})_{i=1..n}$  es su derivativa según cada dimensión. Es una **aproximación lineal de la función al nivel local**.



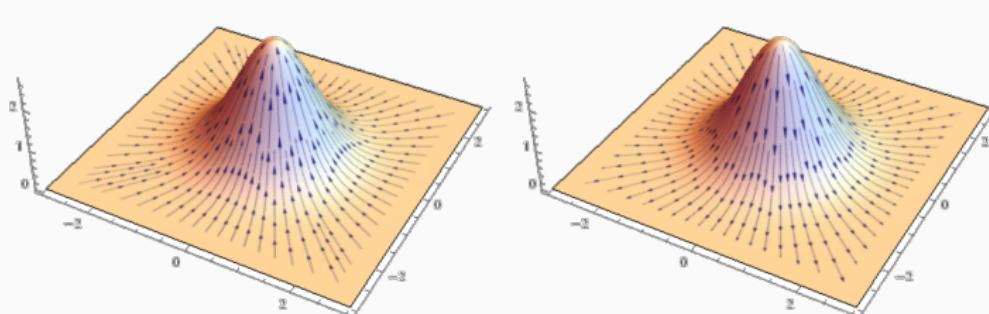
# Gradiente

Una función multivariante  $f(x)$  puede escribirse como una serie de Taylor:

$$f(x + \delta_x) = f(x) + \nabla_x f(x)\delta_x + \mathcal{O}(\|\delta_x^2\|)$$

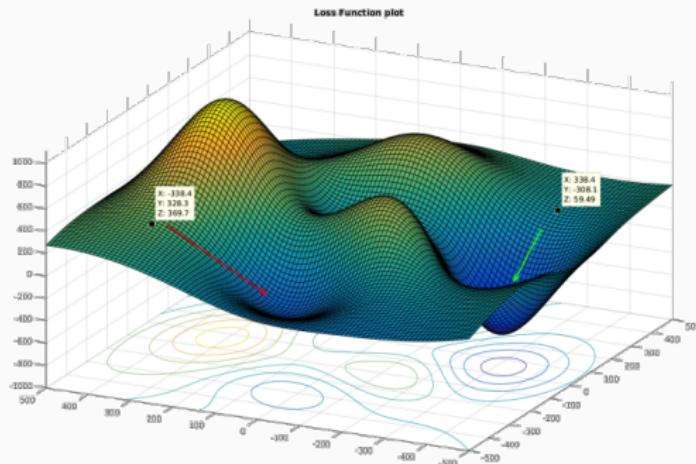
## Definición

El gradiente de una función  $\nabla_x f(x) = (\frac{\partial f}{\partial_i})_{i=1..n}$  es su derivativa según cada dimensión. Es una **aproximación lineal de la función al nivel local**.



Mas detalles [aca.](#)

# Visualización de la función de costo



Se puede visualizar el valor de la función de costo como una superficie:

- Los valores de los parámetros  $\theta$  varían en el plano, y el valor de la función  $\ell(\mathcal{D}_n; \theta)$  varía en altura.
- La convergencia se produce cuando se tienen parámetros que están en un hueco de esta superficie (mínimo local o global, dependiendo del modelo)

# Optimización : visualización

# Optimización : Descenso del Gradiente Estocástico

## Descenso del gradiente

Después de cada cálculo de la función de costo  $\ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$ , se calcula el gradiente de esta función para actualizar los parámetros  $\theta$ :

$$\theta \leftarrow \theta - \alpha * \nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$$

## Ejemplo

- Sean  $f_\theta(\mathbf{X}) = \theta^T \mathbf{X} = \sum_k^d \theta_k X^{(k)}$  y  $\ell(Y, f_\theta(\mathbf{X})) = \frac{1}{2}(Y - f_\theta(\mathbf{X}))^2$

$$\nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_d} \end{pmatrix} \cdot \ell(Y_i, f_\theta(\mathbf{X}_i); \theta) = \begin{pmatrix} X_i^{(1)}(Y - f_\theta(\mathbf{X}_i)) \\ \vdots \\ X_i^{(d)}(Y - f_\theta(\mathbf{X}_i)) \end{pmatrix}$$

# Optimización : Descenso del Gradiente Estocástico

## Descenso del gradiente

Después de cada cálculo de la función de costo  $\ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$ , se calcula el gradiente de esta función para actualizar los parámetros  $\theta$ :

$$\theta \leftarrow \theta - \alpha * \nabla_{\theta} \ell(Y_i, f_\theta(\mathbf{X}_i); \theta)$$

## Ejemplo

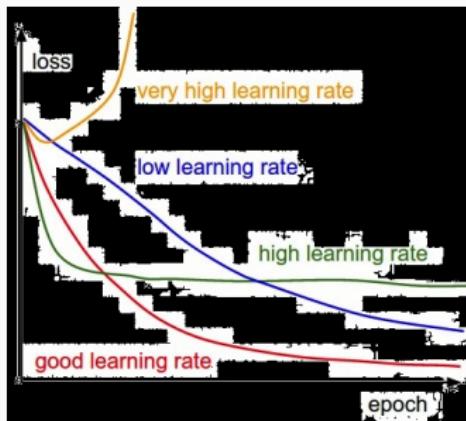
- Sean  $f_\theta(\mathbf{X}) = \theta^T \mathbf{X} = \sum_k^d \theta_k X^{(k)}$  y  $\ell(Y, f_\theta(\mathbf{X})) = \frac{1}{2}(Y - f_\theta(\mathbf{X}))^2$

$$\begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} \leftarrow \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} - \alpha * \begin{pmatrix} X_i^{(1)}(Y - f_\theta(\mathbf{X}_i)) \\ \vdots \\ X_i^{(d)}(Y - f_\theta(\mathbf{X}_i)) \end{pmatrix}$$

# Optimización : Importancia de la tasa de aprendizaje

## Learning rate

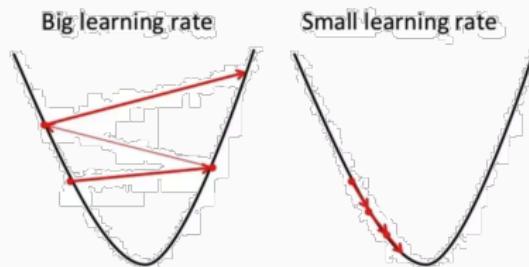
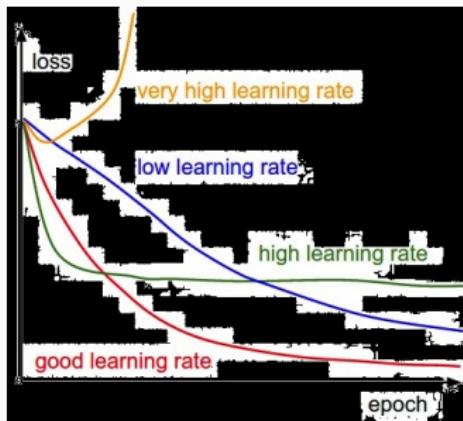
- Un  $\alpha$  demasiado pequeño no avanza en el aprendizaje
- Un  $\alpha$  demasiado pequeño alarga el tiempo de entrenamiento
- Un  $\alpha$  demasiado grande no permite alcanzar el mínimo (damos vueltas alrededor del agujero de la superficie)
- Un  $\alpha$  demasiado grande no permite nada
- Una solución: disminuir  $\alpha$  con el tiempo



# Optimización : Importancia de la tasa de aprendizaje

## Learning rate

- Un  $\alpha$  demasiado pequeño no avanza en el aprendizaje
- Un  $\alpha$  demasiado pequeño alarga el tiempo de entrenamiento
- Un  $\alpha$  demasiado grande no permite alcanzar el mínimo (damos vueltas alrededor del agujero de la superficie)
- Un  $\alpha$  demasiado grande no permite nada
- Una solución: disminuir  $\alpha$  con el tiempo



# Algoritmos de descenso del gradiente

Existen otros algoritmos existentes que se basan en un descenso del gradiente estocástico (SGD) con especificidades para mejorar su eficacia:

- Descenso del gradiente estocástico con momento
- Gradiente acelerado de Nestorov (NAG)
- Gradiente adaptativo (AdaGrad)
- Adam
- RMSprop

Finalmente, también existen otros métodos más clásicos de descenso: BFGS, L-BFGS, Quasi Newton, ...

# Algoritmos de descenso del gradiente

## **Modelos de clasificación**

---

# Outline : Modelos de clasificación

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

API : Scikit-learn

TP: Regresión Lineal

# Outline : Clasificador lineal

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

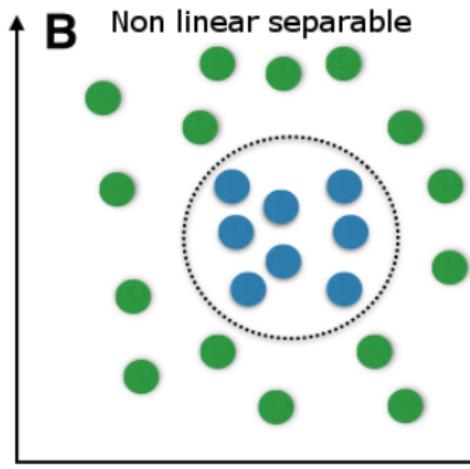
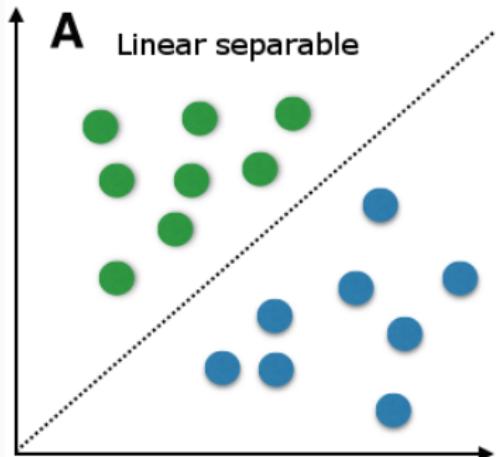
Regresión Lineal

Regresión Logística

## API : Scikit-learn

TP: Regresión Lineal

# Problemas lineales y no lineales



## Función lineal

Una función  $f : \mathcal{X} \rightarrow \mathcal{Y}$  es lineal si:

$$\forall (\mathbf{X}, \mathbf{X}') \in \mathcal{X} \text{ y } \lambda \in \mathbb{R} \text{ entonces } f(\lambda \mathbf{X} + \mathbf{X}') = \lambda f(\mathbf{X}) + f(\mathbf{X}')$$

Es el caso de las funciones de tipo  $f(\mathbf{X}) = \theta^T \mathbf{X} = \sum_i \theta_i X^{(i)}$

# Recordatorios de geometría: hiperplano I

## Definición

En un espacio de dimensión  $d$ , como  $\mathbb{R}^d$  un hiperplano  $\mathcal{H}$  es el

conjunto de puntos  $\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$  que satisfacen una ecuación del tipo:

$$\mathcal{H} : w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0 = 0, \quad \text{donde } \forall i \quad w_i \in \mathbb{R}$$

**Un hiperplano divide el espacio en 2 partes** (es un espacio de dimensión  $d - 1$ ). En  $\mathbb{R}^2$  (plano) es de dimensión 1 (es ??), en  $\mathbb{R}^3$  es de dimensión 2 (es ??)...

**El signo** de  $w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0$  define de **qué lado del espacio** separado por  $\mathcal{H}$  **se encuentra**  $\mathbf{X}$  : si  $> 0$  entonces  $\mathbf{X}$  está de un lado del hiperplano, sino está del otro lado.

# Recordatorios de geometría: hiperplano I

## Definición

En un espacio de dimensión  $d$ , como  $\mathbb{R}^d$  un hiperplano  $\mathcal{H}$  es el

conjunto de puntos  $\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$  que satisfacen una ecuación del tipo:

$$\mathcal{H} : w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0 = 0, \quad \text{donde } \forall i \quad w_i \in \mathbb{R}$$

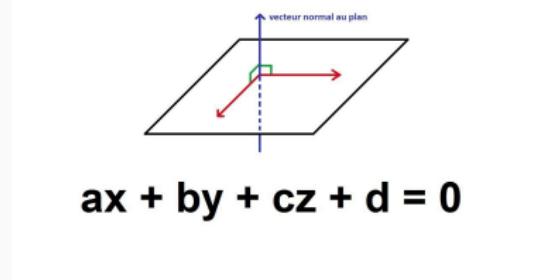
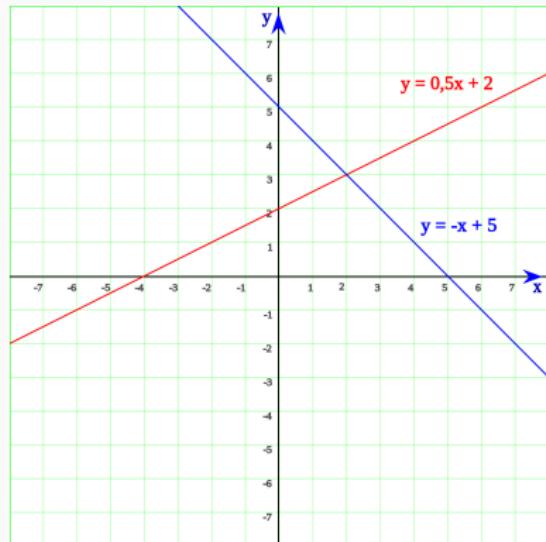
**Un hiperplano divide el espacio en 2 partes** (es un espacio de dimensión  $d - 1$ ). En  $\mathbb{R}^2$  (plano) es de dimensión 1 (es **una recta**), en  $\mathbb{R}^3$  es de dimensión 2 (es **un plano**)...

**El signo** de  $w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0$  define de **qué lado del espacio** separado por  $\mathcal{H}$  **se encuentra**  $\mathbf{X}$  : si  $> 0$  entonces  $\mathbf{X}$  está de un lado del hiperplano, sino está del otro lado.

## Recordatorios de geometría: hiperplano II

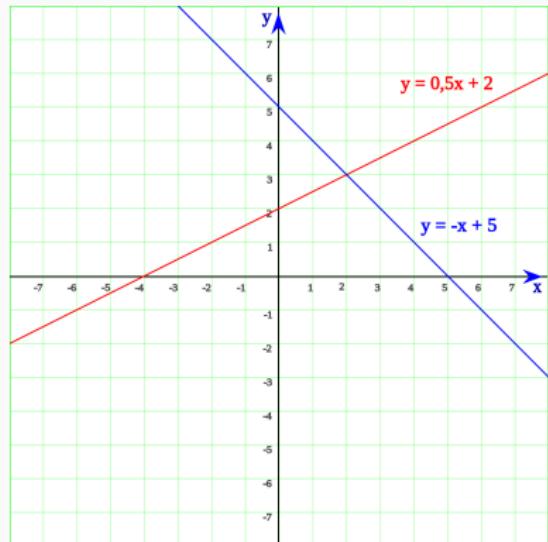
Un hiperplano en 2D (= ???) definido por un vector normal  $\vec{n} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$  y un sesgo  $w_0$ .

Un hiperplano en 3D (= ???) definido por un vector normal  $\vec{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$  y un sesgo  $d$

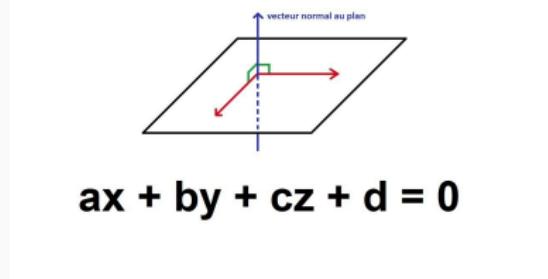


## Recordatorios de geometría: hiperplano II

Un hiperplano en 2D (= Recta) definido por un vector normal  $\vec{n} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$  y un sesgo  $w_0$ .



Un hiperplano en 3D (= Plano) definido por un vector normal  $\vec{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$  y un sesgo  $d$

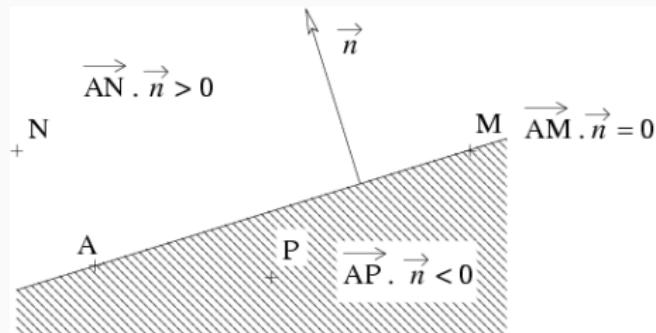


# Recordatorios de geometría: hiperplano III

## Ecuación

Para cualquier punto  $\mathbf{X} \in \mathbb{R}^d$ , el signo de  $w_1x_1 + \dots + w_dx_d$  delimita **de qué lado del espacio separado por el hiperplano** se encuentra el punto  $\mathbf{X}$

Tenemos lo que está de un lado del plano y lo que está del otro lado del plano según el valor del producto escalar con la normal  $\vec{n}$



## Recordatorios de geometría: hiperplano IV

Con  $\mathbf{X} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \\ 1 \end{pmatrix}$  y  $\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \\ \theta_0 \end{pmatrix}$  entonces se puede reducir la ecuación anterior a:

$$\mathcal{H} : w_1x_1 + w_2x_2 + \dots + w_dx_d + w_0 = w_0 + \sum_{i=1}^d w_i x_i = \langle \theta | \mathbf{X} \rangle = 0$$

Donde  $\langle \theta | \mathbf{X} \rangle$  es el producto escalar entre  $\theta$  y  $\mathbf{X}$ : una operación **lineal**.

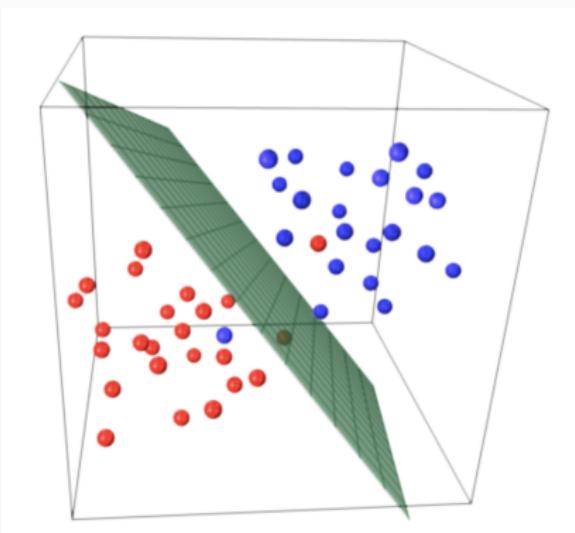
### Aumento del espacio para linealidad

Al aumentar el tamaño del espacio, se puede representar un hiperplano afín mediante la ecuación de un hiperplano lineal.

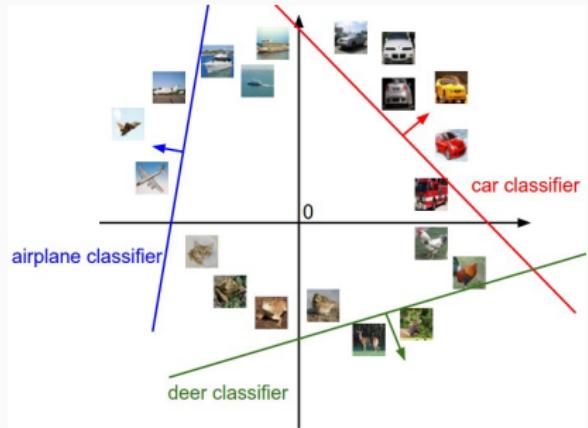
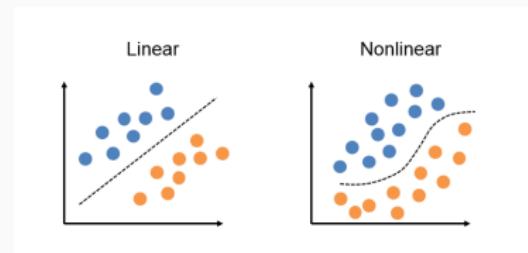
# Clasificador lineal: resumen

## Resumen

- $\mathbf{X} \in \mathbb{R}^d$  es el vector de descriptores
- La ecuación  $\mathbf{W}^T \mathbf{X} + b = 0$  define un hiperplano en  $\mathbb{R}^d$
- $f_{\mathbf{W}, b}(\mathbf{X}) = \text{signo}(\mathbf{W}^T \mathbf{X} + b)$  da la clase de  $\mathbf{X}$



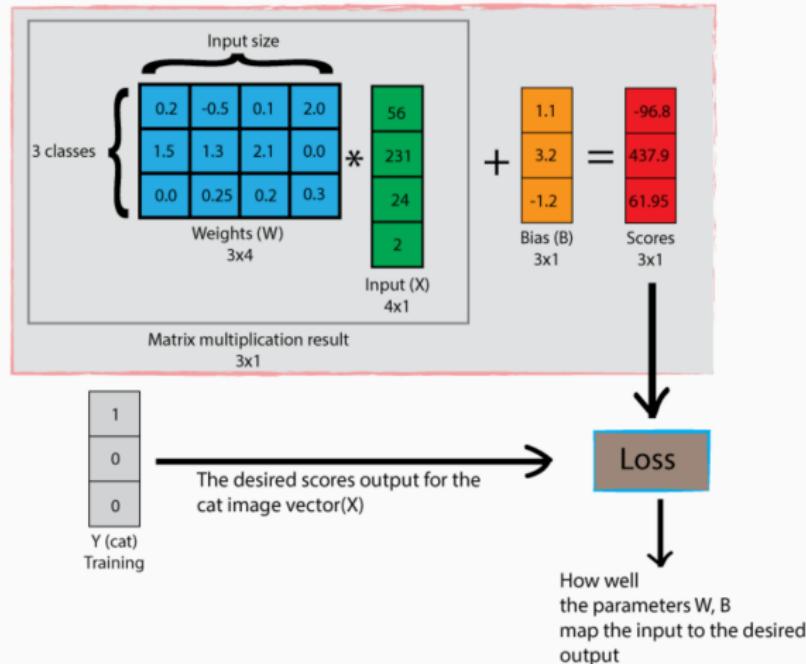
# Clasificador lineal



## Linealidad

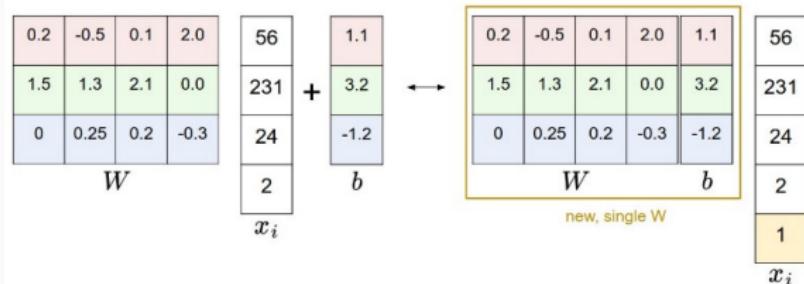
- Separa el espacio con un hiperplano
- Clasificador de la forma  $f_{W,b}(\mathbf{X}) = \text{signo}(\mathbf{W}^T \mathbf{X} + b)$
- Dos ejemplos anteriores: binario (*2 clases*) y multinomial (*n clases*)
- Operación matricial lineal

# Clasificador lineal multiclasses : operación matricial



Ya no tenemos un simple vector de parámetros  $\mathbf{W} = (w_1 \dots w_D)$  como antes, sino una matriz  $\mathbf{W} = (w_{cd})_{c=1..C, d=1..D}$

# Clasificador lineal : Integración del sesgo



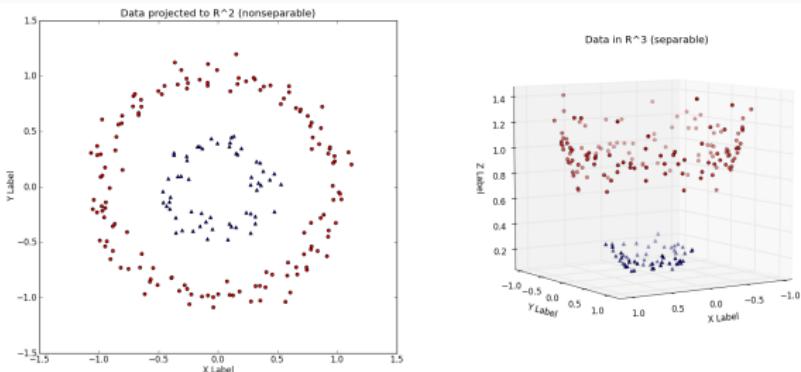
## Linealidad

Al aumentar la dimensión y concatenar un simple vector de 1 al final de los descriptores, obtenemos una operación lineal

- Separa el espacio con un hiperplano
- Clasificador de la forma  $f_{W,b}(\mathbf{X}) = \arg \max_{clases} (\mathbf{W}^T \mathbf{X} + b)$
- Si solo hay 2 clases, el argmax puede reemplazarse por una función de signo

# Clasificador lineal : Aumento del espacio

Al aumentar la dimensión y agregar nuevos descriptores, podemos hacer que nuestro modelo sea no lineal.



## Datos no linealmente separables

- Conjunto de datos no separables linealmente
- Tenemos descriptores de la forma  $\mathbf{X} = (x, y)$ .
- Al definir  $z = x^2 + y^2$ , obtenemos descriptores en un espacio de dimensión 3  $\mathbf{X} = (x, y, x^2 + y^2)$  y así podemos separar linealmente en el espacio de dimensión 3

# Outline : Regresión Lineal

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

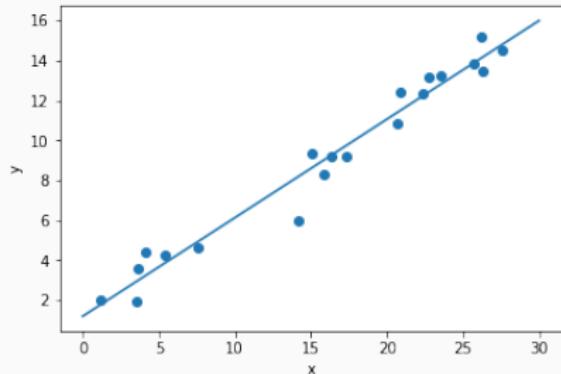
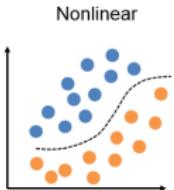
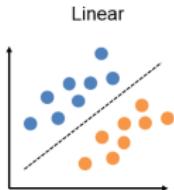
Regresión Lineal

Regresión Logística

API : Scikit-learn

TP: Regresión Lineal

# Regresión Lineal



## Linealidad

- Se modela con un hiperplano.
- Clasificador de la forma  $f_{\mathbf{W}, b}(\mathbf{X}) = \mathbf{W}^T \mathbf{X} + b = \sum_{i=1}^D w_i X_i + b$
- Operación matricial lineal.
- Puede adaptarse a datos no lineales mediante la adición de nuevas variables no lineales.

## Solucion simple

Para encontrar los parámetros que minimizan el error calculamos las derivadas parciales de SSE<sup>1</sup> respecto  $\beta_0$  y  $\beta_1$ . Luego igualamos las derivadas a cero y resolvemos la ecuación para despejar los parámetros.

$$\frac{\partial R}{\partial \beta_0} = -2 \sum_1^N (y_i - \beta_0 - \beta_1 x_i) = 0$$

$$\frac{\partial R}{\partial \beta_1} = -2 \sum_1^N (y_i - \beta_0 - \beta_1 x_i) x_i = 0$$

Lo que da:

$$\beta_1 = \frac{\sum_1^N (x_i - \bar{x})(y_i - \bar{y})}{\sum_1^N (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

---

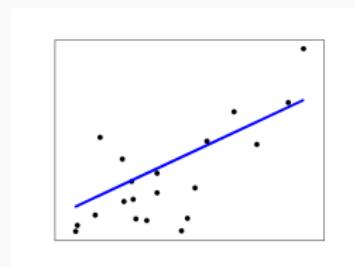
<sup>1</sup>sum of squared errors

# Evaluación de la predicción: distancia

## Diferentes funciones para calcular el rendimiento de una regresión

- Error medio absoluto:  $\frac{1}{n} \sum_{i=1}^n |Y_i - f(\mathbf{X}_i)|$
- Error cuadrático medio:  $\frac{1}{n} \sum_{i=1}^n (Y_i - f(\mathbf{X}_i))^2$
- Error medio mediano: mediana( $|Y_1 - f(\mathbf{X}_1)|, \dots, |Y_n - f(\mathbf{X}_n)|$ )
- Coeficiente de determinación  $R^2$ :

$$1 - \frac{\sum_{i=1}^n |Y_i - f(\mathbf{X}_i)|^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2} = \frac{\sum_{i=1}^n |f(\mathbf{X}_i) - \bar{Y}|^2}{\sum_{i=1}^n (Y_i - \bar{Y})^2}$$



$R^2$  representa la **proporción de la varianza explicada** por el modelo

# Regresión Lineal con 2 features: un plan

Regression Plane

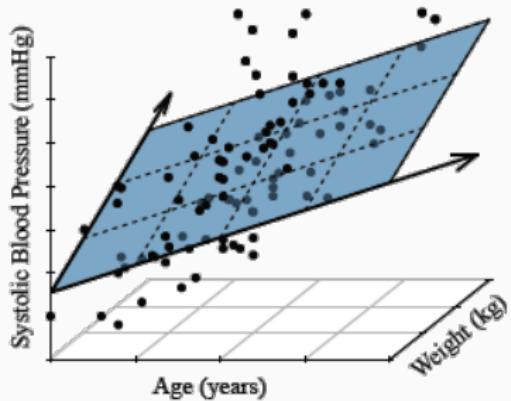


Figure 2.25: Systolic blood pressure linearly increases with age, but also with bodyweight. A line in two directions forms a plane.

Residuals

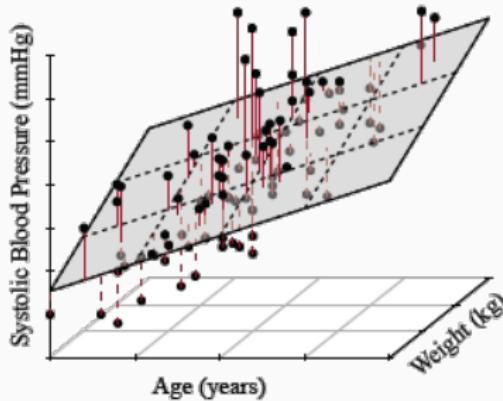


Figure 2.26: The residuals of figure 2.25 are the vertical distances to the plane. Negative residuals are indicated by dashed linepieces.

Figure 1: Los 2 atributos + la etiqueta son reales: 3 dimensiones

# Outline : Regresión Logística

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

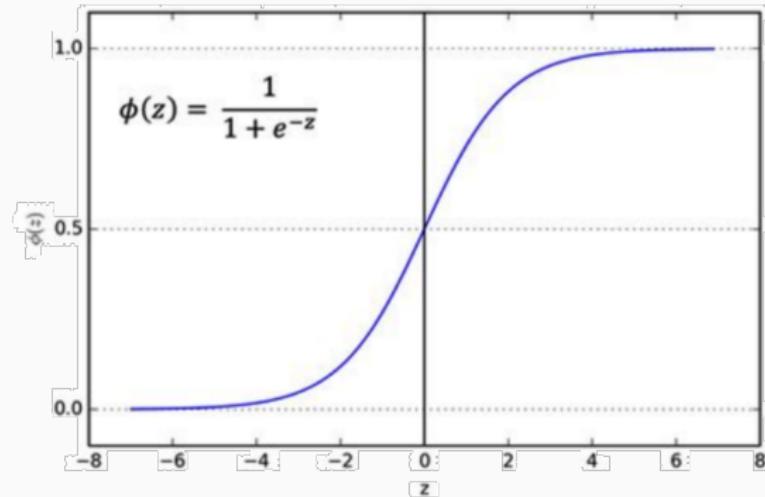
API : Scikit-learn

TP: Regresión Lineal

# Regresión Logística

Para transformar las "distancias"  $\mathbf{W}^T \mathbf{X} + b$  entre los vectores y el plan en probabilidades de classes tenemos que utilizar una función de normalización, la función sigmoidal o *softmax*:

$$\Phi(\text{dist}) = \frac{1}{1 + e^{-\text{dist}}}$$



# Regresión Logística

## Regresión logística binaria

- **Entrenamiento:** Costo:  $\ell(\mathbf{X}, Y) = \frac{1}{1+\exp^{Y<\theta|\mathbf{x}>}}$ , (aquí, ??)
- **Clasificación :**

$$P(Y=1) = \frac{1}{1 + \exp^{-<\theta|\mathbf{x}>}} = \frac{1}{1 + \exp^{-\sum_{k=1}^d \theta_k X^{(k)}}}$$

- Selección de etiqueta según el valor de  $<\theta|\mathbf{X}>$ :

$$P(Y=1) = \frac{1}{1 + \exp^{-<\theta|\mathbf{x}>}}$$

- Lo que da para la otra clase:

$$P(Y=-1) = 1 - P(Y=1) = \frac{\exp^{-<\theta|\mathbf{x}>}}{1 + \exp^{-<\theta|\mathbf{x}>}} = \frac{1}{1 + \exp^{+<\theta|\mathbf{x}>}}$$

- Clase final:  $\hat{c} = \arg \max_c P(Y=c)$

# Regresión Logística

## Regresión logística binaria

- Entrenamiento: Costo:  $\ell(\mathbf{X}, Y) = \frac{1}{1+\exp^{Y<\theta|\mathbf{x}>}}$ , (aquí, logística)
- Clasificación :

$$P(Y=1) = \frac{1}{1 + \exp^{-<\theta|\mathbf{x}>}} = \frac{1}{1 + \exp^{-\sum_{k=1}^d \theta_k X^{(k)}}}$$

- Selección de etiqueta según el valor de  $<\theta|\mathbf{X}>$ :

$$P(Y=1) = \frac{1}{1 + \exp^{-<\theta|\mathbf{x}>}}$$

- Lo que da para la otra clase:

$$P(Y=-1) = 1 - P(Y=1) = \frac{\exp^{-<\theta|\mathbf{x}>}}{1 + \exp^{-<\theta|\mathbf{x}>}} = \frac{1}{1 + \exp^{+<\theta|\mathbf{x}>}}$$

- Clase final:  $\hat{c} = \arg \max_c P(Y=c)$

# Regresión Logística Multinomial y Softmax

## De la regresión logística binaria a multinomial

- Clasificador lineal binario y función logística:

$$\begin{aligned} P(Y = 1) &= \frac{1}{1 + \exp^{-\langle \theta | \mathbf{x} \rangle}} = \frac{\exp^{\langle \theta | \mathbf{x} \rangle}}{\exp^{\langle \theta | \mathbf{x} \rangle} + 1} = \frac{\exp^{\langle \theta^{(1)} - \theta^{(-1)} | \mathbf{x} \rangle}}{\exp^{\langle \theta^{(1)} - \theta^{(-1)} | \mathbf{x} \rangle} + 1} \\ &= \frac{\exp^{\langle \theta^{(1)} | \mathbf{x} \rangle}}{\exp^{\langle \theta^{(1)} | \mathbf{x} \rangle} + \exp^{\langle \theta^{(-1)} | \mathbf{x} \rangle}} \end{aligned}$$

- RL para varias clases:  $P(Y = c) = \frac{\exp^{\langle \theta^{(c)} | \mathbf{x} \rangle}}{\sum_{j=1}^C \exp^{\langle \theta^{(j)} | \mathbf{x} \rangle}}$ , con :

$$\theta = \begin{pmatrix} | & | & | & | \\ \theta^{(1)} & \theta^{(2)} & \dots & \theta^{(C)} \\ | & | & | & | \end{pmatrix}$$

- Clase final:  $\hat{c} = \arg \max_c P(Y = c)$

# Regresión Logística – funciones de costo

## A tener en cuenta

- Función de costo logística, con  $Y \in \{0, 1\}$ :  $\ell(\mathbf{X}, Y) = \frac{1}{1 + \exp^{-Y\langle\theta|\mathbf{x}\rangle}}$
- Función de costo entropica, con  $Y \in \{0; 1\}$ :

$$\begin{aligned}\ell(\mathbf{X}, Y) &= -(Y \log\left(\frac{1}{1 + \exp^{-\langle\theta|\mathbf{x}\rangle}}\right) + (1 - Y) \log\left(\frac{1}{1 + \exp^{+\langle\theta|\mathbf{x}\rangle}}\right)) \\ &= -\sum_{c=1}^C \mathbb{1}_{\{Y=c\}} \log(P(Y=c))\end{aligned}$$

**Ejercicio:** ¿Cuál es la derivada  $\nabla_{\theta^{(k)}} \ell(\mathbf{X}, Y; \theta)$  para la función de costo entropica?

# Regresión Logística – funciones de costo

## A tener en cuenta

- Función de costo logística, con  $Y \in \{0, 1\}$ :  $\ell(\mathbf{X}, Y) = \frac{1}{1 + \exp^{Y - \theta' \mathbf{x}}}$
- Función de costo entropica, con  $Y \in \{0, 1\}$ :

$$\begin{aligned}\ell(\mathbf{X}, Y) &= -(Y \log\left(\frac{1}{1 + \exp^{-\theta' \mathbf{x}}}\right) + (1 - Y) \log\left(\frac{1}{1 + \exp^{\theta' \mathbf{x}}}\right)) \\ &= -\sum_{c=1}^C \mathbb{1}_{\{Y=c\}} \log(P(Y=c))\end{aligned}$$

Solución:

$$\nabla_{\theta(k)} \ell(\mathbf{X}, Y; \theta) = -\mathbf{X}(\mathbb{1}_{\{Y=k\}} - P(Y=k))$$

intuición:  $\sigma' = \sigma * (1 - \sigma)$  y  $\ln(u) = \frac{u'}{u}$

# Supuestos del Modelo Lineal

## Linealidad

La variable de respuesta se relaciona linealmente con los atributos.

## Normalidad

Los errores tienen distribución normal de media cero:  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

## Homocedasticidad

Los errores tienen varianza constante (mismo valor de  $\sigma^2$ )

## Independencia

Los errores son independientes entre sí

## **API : Scikit-learn**

---

# Outline : API : Scikit-learn

---

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

## API : Scikit-learn

TP: Regresión Lineal

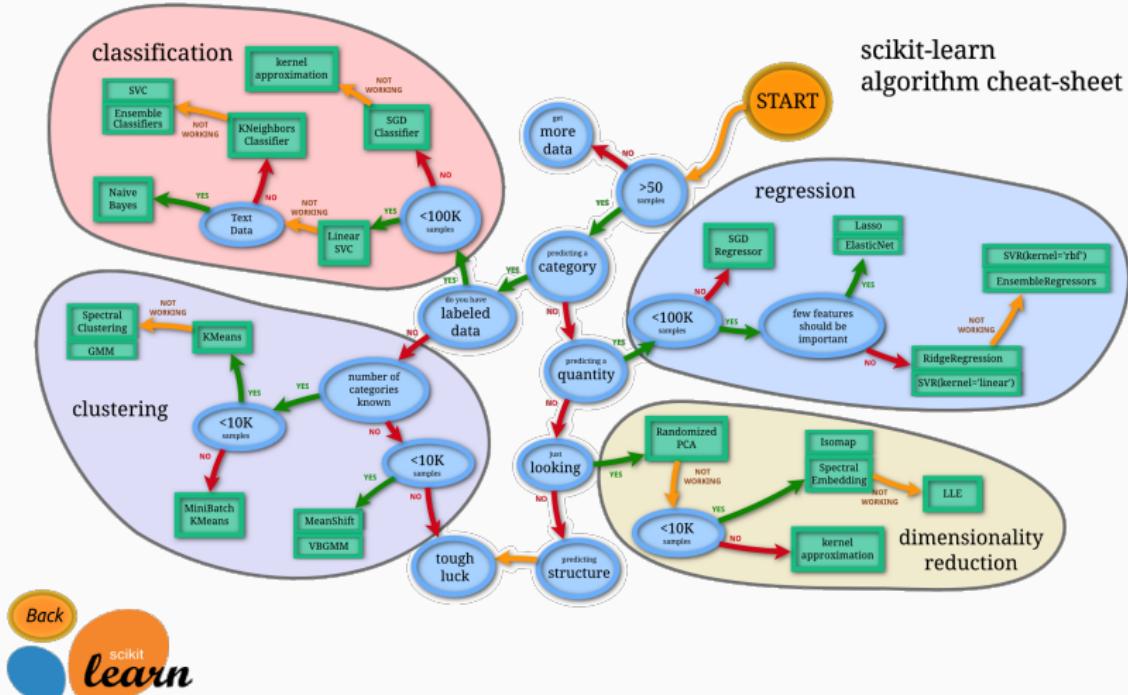
## Scikit-learn: biblioteca de Python

Biblioteca de Aprendizaje Automático muy fácil de usar:

<https://scikit-learn.org/>



# Scikit-learn: mapa de posibilidades



# Scikit-learn: funciones normalizadas

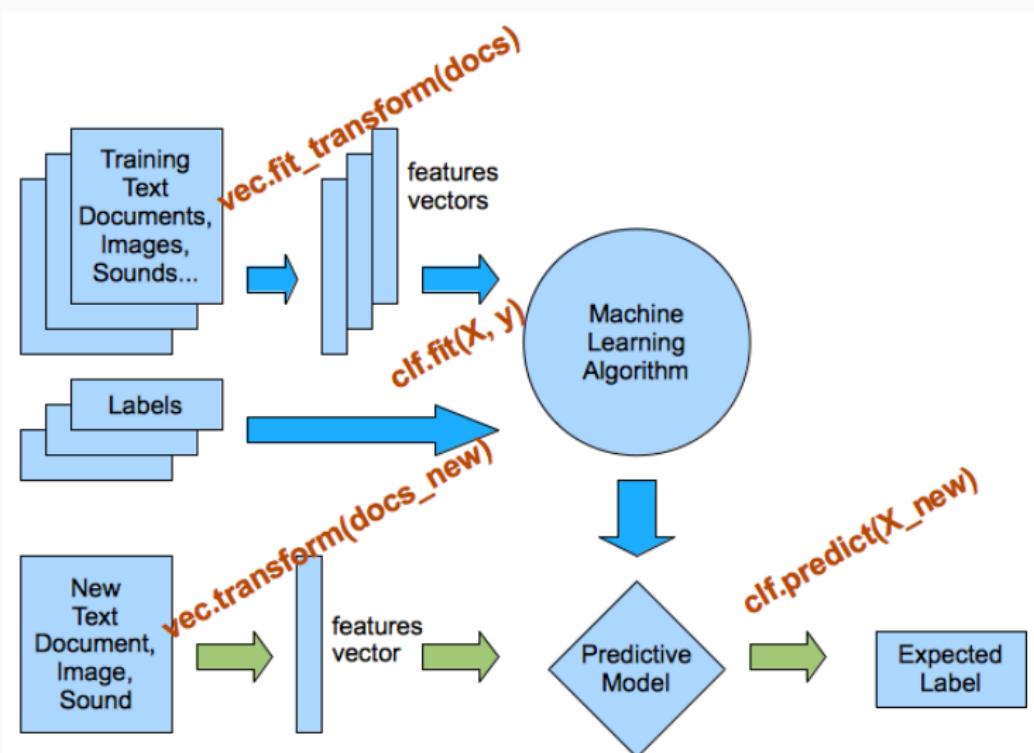
Funciones normalizadas para un aprendizaje rápido y claridad en el código:

- Los diferentes modelos (o conjuntos de datos) se implementan como clases (tipos de variables)
- Se pueden llamar métodos en estas variables para las diferentes etapas del algoritmo: Extracción de características, Aprendizaje de parámetros, Predicción,...

```
>>> from sklearn.datasets import load_iris
>>> data = load_iris()
>>> X, y = data.data, data.target
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(
...     iris.data, iris.target, test_size=0.4, random_state=0)
>>> from sklearn.linear_model import SGDClassifier
>>> clf = SGDClassifier(max_iter=1000, tol=1e-3)
>>> clf.fit(X_train, y_train)
>>> clf.score(X_test, y_test)
0.92...
```

# Scikit-learn: funciones normalizadas

Funciones normalizadas para un aprendizaje rápido y claridad en el código:



# Scikit-learn: Conjuntos de datos

## Múltiples conjuntos de datos disponibles :

- Conjuntos de datos "juguetes": precios de casas de Boston, plantas de Iris, Diabetes, reconocimiento de vinos, ...
- Conjuntos de datos reales: Caras Olivetti (imagen), Coberturas forestales (geográficas), Conjunto de datos RCV1 (texto), ...

```
>>> from sklearn.datasets import load_wine
>>> data = load_wine()
>>> data.keys()
['target_names', 'data', 'target', ...
'DESCR', 'feature_names']
```

# Scikit-learn: Extracción de características

## Funciones ya codificadas para vectorizar fácilmente características

- Desde diccionarios: `sklearn.feature_extraction`
- Desde imágenes: `sklearn.feature_extraction.image`
- Desde texto: `sklearn.feature_extraction.text`

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> corpus = [
...     'Este es el primer documento.',
...     'Este es el segundo segundo documento.',
...     'Y el tercero.',
...     '¿Este es el primer documento?',
... ]
>>> X = vectorizer.fit_transform(corpus).toarray()
array([[0, 1, 1, 1, 0, 0, 1, 0, 1],
[0, 1, 0, 1, 0, 2, 1, 0, 1],
[1, 0, 0, 0, 1, 0, 1, 1, 0],
[0, 1, 1, 1, 0, 0, 1, 0, 1]]...)
```

# Scikit-learn: Selección de características

## Técnicas para seleccionar características

- Eliminar características de baja varianza: VarianceThreshold
- Usar modelos para seleccionar características: SelectFromModel
- Eliminación recursiva de características: RFE

```
>>> from sklearn.svm import LinearSVC
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectFromModel
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> X.shape
(150, 4)
>>> lsvc = LinearSVC(C=0.01, penalty="l1", dual=False).fit(X, y)
>>> model = SelectFromModel(lsvc, prefit=True)
>>> X_new = model.transform(X)
>>> X_new.shape
(150, 3)
```

# Scikit-learn: Preprocesamiento de datos

Normalización, estandarización necesaria para un mejor aprendizaje

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
scaler.transform(X_test)
```

Valores faltantes

```
>>> from sklearn.impute import SimpleImputer
>>> imp = SimpleImputer(missing_values=np.nan, strategy='mean')
>>> imp.fit([[1, 2], [np.nan, 3], [7, 6]])
SimpleImputer(copy=True, fill_value=None, missing_values=nan, strategy=mean)
>>> X = [[np.nan, 2], [6, np.nan], [7, 6]]
>>> print(imp.transform(X))
[[4.          2.          ]
 [6.          3.666...]
 [7.          6.          ]]
```

# Scikit-learn: Validación cruzada

Funciones para la validación de modelos y separación en conjuntos de entrenamiento/prueba:

- De manera simple `train_test_split`, para una K-CrossVal KFold, en Leave-one-out `LeaveOneOut`, conservando las proporciones de las etiquetas `StratifiedKFold`
- Haciendo directamente la CrossVal: `cross_val_score`

```
>>> from sklearn.model_selection import train_test_split  
>>> X_train, X_test, y_train, y_test = train_test_split(  
...     iris.data, iris.target, test_size=0.4, random_state=0)  
  
>>> X_train.shape, y_train.shape  
((90, 4), (90,))  
>>> X_test.shape, y_test.shape  
((60, 4), (60,))
```

# Scikit-learn: Modelos de aprendizaje supervisado

Numerosos modelos supervisados normalizados para una fácil utilización:

- Regresión Logística/Lineal, Análisis Discriminativo Lineal/Quadrático, SVM, K-NN, Descenso de Gradiente Estocástico, Bayesiano Ingenuo, Árboles de decisión, Bosques Aleatorios, ...
- Cada uno de estos modelos se implementan como clases python con métodos que son comunes a todas las clases: fit, score, predict\_proba, predict, ...

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0, solver='lbfgs',
...                           multi_class='multinomial').fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

# Scikit-learn: Modelos de aprendizaje no supervisado

Numerosos modelos no supervisados normalizados para clustering, descomposición de señales en componentes, reducción de dimensiones:

- Clustering: K-medias, PCA, Factorización de Matrices no negativas, Asignación Latente de Dirichlet, t-sne,...
- Estos modelos se implementan como clases python con métodos que son comunes a todas las clases: fit, fit\_transform, predict,  
...

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...                 [4, 2], [4, 4], [4, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([0, 0, 0, 1, 1, 1], dtype=int32)
>>> kmeans.predict([[0, 0], [4, 4]])
array([0, 1], dtype=int32)
>>> kmeans.cluster_centers_
array([[1., 2.],
       [4., 2.]])
```

# Scikit-learn: Métricas

Múltiples métricas disponibles para la evaluación de modelos:

- de clasificación: Recuerdo, Precisión, F1, AUC, ROC, Matriz de confusión, ...
- de regresión:  $R^2$ , Error Cuadrático Medio, ..
- de clustering: Completitud, V-medida, ...

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
```

# Scikit-learn: Encontrar los hiperparámetros con una búsqueda

Funciones para encontrar los hiperparámetros óptimos realizando validaciones cruzadas para diferentes hiperparámetros.

- En una cuadrícula dada: GridSearchCV
- En relación con una distribución aleatoria de parámetros: RandomizedSearchCV
- Estos modelos se implementan como clases python con métodos que son comunes a todas las clases: fit, fit\_transform, predict,...

```
>>> from sklearn import svm, datasets
>>> from sklearn.model_selection import GridSearchCV
>>> iris = datasets.load_iris()
>>> parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
>>> svc = svm.SVC(gamma="scale")
>>> clf = GridSearchCV(svc, parameters, cv=5)
>>> clf.fit(iris.data, iris.target)
```

# Outline : TP: Regresión Lineal

## Recordatorios

Resumen

Optimización

## Modelos de clasificación

Clasificador lineal

Regresión Lineal

Regresión Logística

## API : Scikit-learn

TP: Regresión Lineal

# TP Introducción a Scikit-learn y regresión lineal

Estudio de la regresión mediante un plano lineal de una superficie cuadrática:

$$Y = 3X^{(1)} - 2X^{(2)^2} + \epsilon$$

Introducción a Scikit-learn:

- Separación en conjuntos de entrenamiento y prueba
- Creación, entrenamiento y prueba de modelos de regresión de Scikit-learn
- Ampliación del espacio de los descriptores para crear un modelo no lineal
- Comparación con otros modelos menos adecuados

**Questions?**

## References i