

# Regularización para regresión y clasificación

Mathieu Kessler

Departamento de Matemática Aplicada y Estadística  
Universidad Politécnica de Cartagena

Cartagena

# El problema del sobreajuste

- Si añadimos características adicionales (por ejemplo potencias de las características) podemos a veces mejorar el ajuste a nuestro conjunto de entrenamiento, o mejorar nuestra capacidad de clasificar sus observaciones.

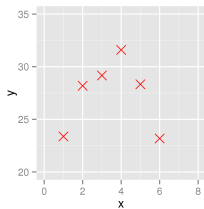
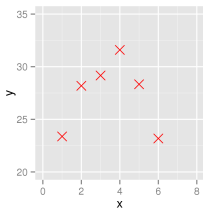
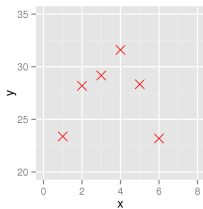
# El problema del sobreajuste

- Si añadimos características adicionales (por ejemplo potencias de las características) podemos a veces mejorar el ajuste a nuestro conjunto de entrenamiento, o mejorar nuestra capacidad de clasificar sus observaciones.
- Pero, puede ser que la mejora de este ajuste sea artificial

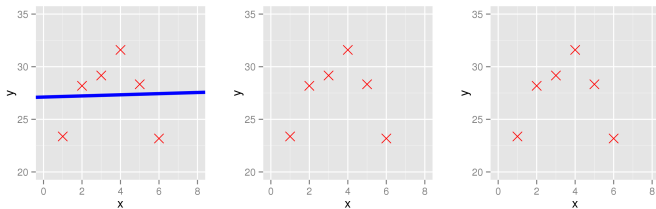
# El problema del sobreajuste

- Si añadimos características adicionales (por ejemplo potencias de las características) podemos a veces mejorar el ajuste a nuestro conjunto de entrenamiento, o mejorar nuestra capacidad de clasificar sus observaciones.
- Pero, puede ser que la mejora de este ajuste sea artificial
- Hemos mejorado aparentemente el ajuste para el conjunto de entrenamiento pero puede empeorar en el conjunto de test.

# Ejemplo: Ajuste con una característica $x$

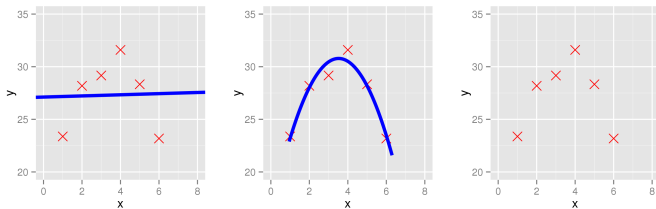


## Ejemplo: Ajuste con una característica $x$



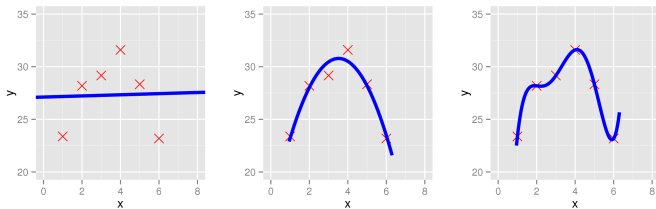
- Ajuste con una recta:  $y = \theta_0 + \theta_1 x$ , mal ajuste “high bias”

## Ejemplo: Ajuste con una característica $x$



- Ajuste con una recta:  $y = \theta_0 + \theta_1 x$ , mal ajuste “high bias”
- Parábola  $y = \theta_0 + \theta_1 x + \theta_2 x^2$ , ajuste adecuado.

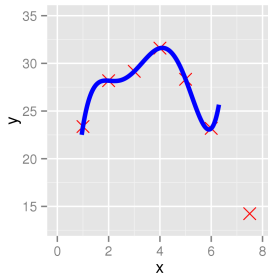
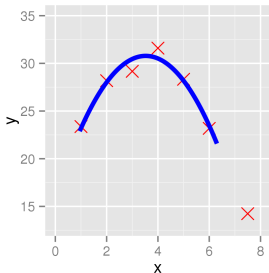
## Ejemplo: Ajuste con una característica $x$



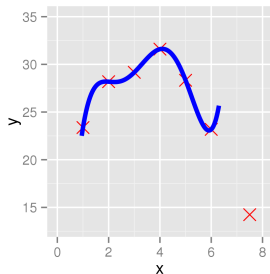
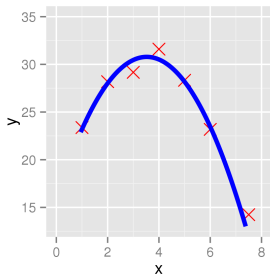
- Ajuste con una recta:  $y = \theta_0 + \theta_1 x$ , mal ajuste. “High bias.”
- Parábola  $y = \theta_0 + \theta_1 x + \theta_2 x^2$ , ajuste adecuado.
- Polinomio orden 5:  $y = \theta_0 + \theta_1 x + \dots + \theta_5 x^5$ , ajuste artificialmente bueno... “High variance.”



# Si usamos los modelos para predecir

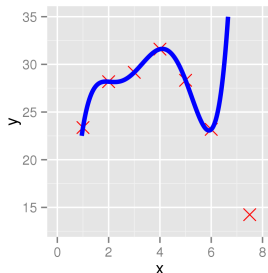
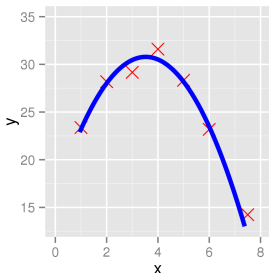


# Si usamos los modelos para predecir



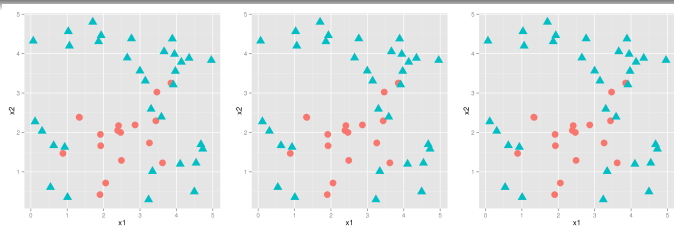
- Parábola  $y = \theta_0 + \theta_1x + \theta_2x^2$ , predicción correcta.

# Si usamos los modelos para predecir

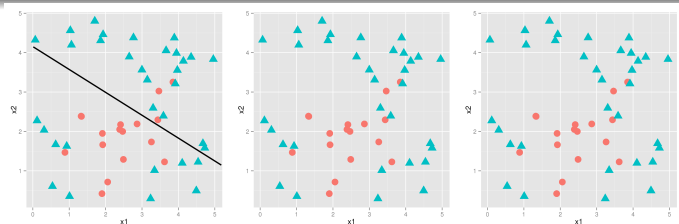


- Parábola  $y = \theta_0 + \theta_1x + \theta_2x^2$ , predicción correcta.
- Polinomio orden 5:  $y = \theta_0 + \theta_1x + \dots + \theta_5x^5$ , predicción muy mala.

# Ejemplo: Clasificación con dos características $x_1$ y $x_2$ .

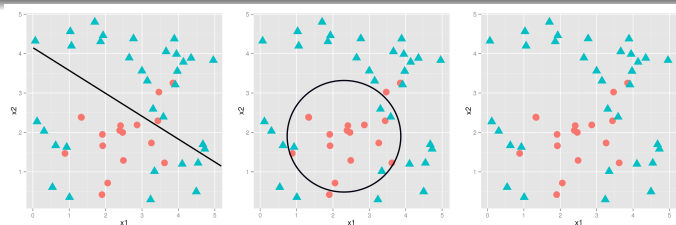


## Ejemplo: Clasificación con dos características $x_1$ y $x_2$ .



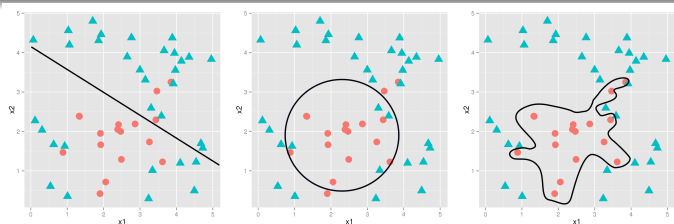
- Clasificación, frontera es una recta:  $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ .

## Ejemplo: Clasificación con dos características $x_1$ y $x_2$ .



- Clasificación, frontera es una recta:  $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ .
- Clasificación, frontera es un círculo:  
 $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \dots)$ .

## Ejemplo: Clasificación con dos características $x_1$ y $x_2$ .



- Clasificación, frontera es una recta:  $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ .
- Clasificación, frontera es un círculo:  
 $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \dots)$ .
- Clasificación, frontera con polinomios orden superior  
 $y = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \dots + \theta_6 x_1^6 + \dots)$ .

# Para evitar el sobre ajuste (overfitting)



# Para evitar el sobre ajuste (overfitting)

- Podríamos seleccionar manualmente las características que pensamos que pueden servir, sin pasarnos...

# Para evitar el sobre ajuste (overfitting)

- Podríamos seleccionar manualmente las características que pensamos que pueden servir, sin pasarnos...
- Podemos usar técnicas de selección de modelos: criterios estadísticos de elección de las características más informativas.

# Para evitar el sobre ajuste (overfitting)

- Podríamos seleccionar manualmente las características que pensamos que pueden servir, sin pasarnos...
- Podemos usar técnicas de selección de modelos: criterios estadísticos de elección de las características más informativas.
- Podemos introducir un término en la función coste que penaliza los ajustes con demasiadas variables...

# Para evitar el sobre ajuste (overfitting)

- Podríamos seleccionar manualmente las características que pensamos que pueden servir, sin pasarnos...
- Podemos usar técnicas de selección de modelos: criterios estadísticos de elección de las características más informativas.
- Podemos introducir un término en la función coste que penaliza los ajustes con demasiadas variables...  
Es lo que llamamos la **regularización**.

# Modificación de la función coste: regresión lineal

Consideramos la regresión lineal múltiple, la función coste es:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2,$$

donde  $\theta = (\theta_0, \dots, \theta_k)$  y  $x_{i\bullet}$  denota el vector de características para el individuo  $i$ .

# Modificación de la función coste: regresión lineal

Consideramos la regresión lineal múltiple, la función coste es:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2,$$

donde  $\theta = (\theta_0, \dots, \theta_k)$  y  $x_{i\bullet}$  denota el vector de características para el individuo  $i$ .

Regularización: primera opción,  $l_2$ .

Añadimos un término:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

donde  $\lambda$  es un número positivo que tendremos que fijar.

# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

- **Ojo:** el sumatorio  $\sum_{j=1}^k \theta_j^2$  empieza en  $j = 1$ , no incluye  $\theta_0$  (porque corresponde al término constante).



# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

- **Ojo:** el sumatorio  $\sum_{j=1}^k \theta_j^2$  empieza en  $j = 1$ , no incluye  $\theta_0$  (porque corresponde al término constante).
- Si  $\lambda$  es grande, para minimizar la función coste, se tenderá hacia una solución donde  $\theta_1^2 + \dots + \theta_k^2$  será pequeño, por ejemplo si varios son casi nulos.

# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

- **Ojo:** el sumatorio  $\sum_{j=1}^k \theta_j^2$  empieza en  $j = 1$ , no incluye  $\theta_0$  (porque corresponde al término constante).
- Si  $\lambda$  es grande, para minimizar la función coste, se tenderá hacia una solución donde  $\theta_1^2 + \dots + \theta_k^2$  será pequeño, por ejemplo si varios son casi nulos.
- Ajustaremos el valor de  $\lambda$ , según si queremos penalizar mucho las soluciones con muchos términos significativos o no...

# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

- Hablamos de regularización  $l_2$ , porque el término que se añade es proporcional a la norma  $l_2$  del vector de parámetros  $\theta[-1] = (0, \theta_1, \dots, \theta_k).$ ,

$$\|\theta[-1]\|^2 = \sum_{j=1}^k \theta_j^2.$$

# Regularización para regresión lineal

La nueva función coste:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2,$$

- Hablamos de regularización  $l_2$ , porque el término que se añade es proporcional a la norma  $l_2$  del vector de parámetros  $\theta[-1] = (0, \theta_1, \dots, \theta_k).$ ,

$$\|\theta[-1]\|^2 = \sum_{j=1}^k \theta_j^2.$$

Esta versión de la regresión regularizada se llama **"Ridge regression"**.

# Función de coste regularizada para la regresión múltiple.

Recordemos las notaciones para la regresión lineal múltiple.  
Para nuestro conjunto de entrenamiento que consta de  $n$  filas,  
habíamos introducido:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \quad \text{y la matriz} \quad \mathbf{X} = \begin{pmatrix} x_{10} & x_{11} & \cdots & x_{1k} \\ x_{20} & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nk} \end{pmatrix}$$

# Función de coste regularizada para la regresión lineal

La función de coste regularizada para la regresión lineal

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \left( y_i - x_{i\bullet}^T \theta \right)^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2 = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\theta\|^2 + \frac{\lambda}{2n} \|\theta[-1]\|^2,$$

donde  $\theta[-1] = (0, \theta_1, \dots, \theta_k)$ .

# Función de coste regularizada para la regresión lineal

La función de coste regularizada para la regresión lineal

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \left( y_i - x_{i\bullet}^T \theta \right)^2 + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2 = \frac{1}{n} \| \mathbf{y} - \mathbf{X}\theta \|^2 + \frac{\lambda}{2n} \| \theta[-1] \|^2,$$

donde  $\theta[-1] = (0, \theta_1, \dots, \theta_k)$ .

El gradiente es por lo tanto:

$$\nabla J(\theta) = \frac{2}{n} \mathbf{X}^T \cdot (\mathbf{X}\theta - \mathbf{y}) + \frac{\lambda}{n} \theta[-1].$$

# Ridge regression en scikit-learn

Para implementar Ridge regression en scikit-learn, se puede usar la clase Ridge del submódulo linear\_model.

```
from sklearn.linear_model import Ridge
```



## Ridge regression en scikit-learn

Para implementar Ridge regression en scikit-learn, se puede usar la clase Ridge del submódulo linear\_model.

```
from sklearn.linear_model import Ridge
```

A la hora de instanciar nuestro estimador, podemos fijar el valor de  $\lambda$  que multiplica la norma  $l_2$ , especificando el parámetro alpha.

```
ridge_reg = Ridge(alpha=1)
```

## Regularización: segunda opción, $l_1$ .

Añadimos un término:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{n} \sum_{j=1}^k |\theta_j|,$$

donde  $\lambda$  es un número positivo que tendremos que fijar.

## Regularización: segunda opción, $l_1$ .

Añadimos un término:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{n} \sum_{j=1}^k |\theta_j|,$$

donde  $\lambda$  es un número positivo que tendremos que fijar.

Esta versión de la regresión regularizada se llama *Least Absolute Shrinkage and Selection Operator Regression*, i.e. “**Lasso regression**”.

## Regularización: segunda opción, $l_1$ .

Añadimos un término:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + \frac{\lambda}{n} \sum_{j=1}^k |\theta_j|,$$

donde  $\lambda$  es un número positivo que tendremos que fijar.

Esta versión de la regresión regularizada se llama *Least Absolute Shrinkage and Selection Operator Regression*, i.e. “**Lasso regression**”.

Una característica importante de la regresión Lasso es que descarta completamente las características menos importantes (sus coeficientes son 0).

# Lasso regression en scikit-learn

Para implementar Lasso regression en scikit-learn, se puede usar la clase Lasso del submódulo `linear_model`.

```
from sklearn.linear_model import Lasso
```

# Lasso regression en scikit-learn

Para implementar Lasso regression en scikit-learn, se puede usar la clase Lasso del submódulo `linear_model`.

```
from sklearn.linear_model import Lasso
```

A la hora de instanciar nuestro estimador, podemos fijar el valor de  $\lambda$  que multiplica la norma  $l_1$ , especificando el parámetro `alpha`.

```
lasso_reg = Lasso(alpha=0.1)
```

## Regularización: tercera opción, Elastic net.

Se trata de una combinación de Lasso y de Ridge:

Añadimos dos términos:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + r \frac{\lambda}{n} \sum_{j=1}^k |\theta_j| + (1-r) \frac{\lambda}{2n} \sum_{j=1}^k (\theta_j)^2,$$

donde  $\lambda$  es un número positivo y  $r$  es un número entre 0 y 1 que tendremos que fijar.

# Regularización: tercera opción, Elastic net.

Se trata de una combinación de Lasso y de Ridge:

Añadimos dos términos:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\theta, x_{i\bullet}))^2 + r \frac{\lambda}{n} \sum_{j=1}^k |\theta_j| + (1 - r) \frac{\lambda}{2n} \sum_{j=1}^k (\theta_j)^2,$$

donde  $\lambda$  es un número positivo y  $r$  es un número entre 0 y 1 que tendremos que fijar.

## Recomendación

- Cuando tenemos bastantes características, es buena idea usar una regresión regularizada.
- Si sospechamos que hay varias características que no son relevantes, entonces se recomienda usar Lasso o Elastic Net.
- Elastic Net is preferible cuando hay más características que individuos en el conjunto de aprendizaje o cuando están las características muy correlacionadas.



# Elastic net regression en scikit-learn

Para implementar Elastic Net scikit-learn, se puede usar la clase ElasticNet del submódulo linear\_model.

```
from sklearn.linear_model import ElasticNet
```

# Elastic net regression en scikit-learn

Para implementar Elastic Net scikit-learn, se puede usar la clase ElasticNet del submódulo linear\_model.

```
from sklearn.linear_model import ElasticNet
```

A la hora de instanciar nuestro estimador, podemos fijar el valor de  $\lambda$  que multiplica las normas  $l_1$  y  $l_2$ , especificando el parámetro alpha, y el valor de  $r$  que indica el peso de la norma  $l_1$  frente a la norma  $l_2$ , especificando el parámetro l1\_ratio

```
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
```

# Función de coste regularizada para la regresión logística

Vimos en las transparencias de la clase anterior que, para la regresión logística, la función de coste sin regularizar era:

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\},$$

donde  $h_{\theta}(x_{i\bullet}) = g(x_{i\bullet}\theta) = 1/(1 + \exp(-x_{i\bullet}\theta))$ .

# Función de coste regularizada para la regresión logística

Además el gradiente es.

$$\nabla_{\theta} J(\theta) = \frac{1}{n} \sum_{i=1}^n \{x_{i\bullet} \cdot (h_{\theta}(x_{i\bullet}) - y_i)\}.$$

Si usamos la matriz de diseño  $\mathbf{X}$ , obtuvimos en forma compacta:

$$\nabla_{\theta} J(\theta) = \frac{1}{n} \mathbf{X}^T \cdot (\mathbf{H}_{\theta} - \mathbf{y}).$$

donde  $\mathbf{H}$  denota el vector columna:

$$\mathbf{H}_{\theta} = \begin{pmatrix} h_{\theta}(x_{1\bullet}) \\ h_{\theta}(x_{2\bullet}) \\ \vdots \\ h_{\theta}(x_{n\bullet}) \end{pmatrix}$$

# Función de coste regularizada para la regresión logística

Podemos regularizar la función de coste para la regresión logística igual que lo hicimos para la regresión lineal.

# Función de coste regularizada para la regresión logística

Podemos regularizar la función de coste para la regresión logística igual que lo hicimos para la regresión lineal.

- Podemos usar la regularización  $l_2$ :

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\} + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2$$

# Función de coste regularizada para la regresión logística

Podemos regularizar la función de coste para la regresión logística igual que lo hicimos para la regresión lineal.

- Podemos usar la regularización  $l_2$ :

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\} + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2$$

- o la regularización  $l_1$

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\} + \frac{\lambda}{n} \sum_{j=1}^k |\theta_j|$$

# Función de coste regularizada para la regresión logística

Podemos regularizar la función de coste para la regresión logística igual que lo hicimos para la regresión lineal.

- Podemos usar la regularización  $l_2$ :

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\} + \frac{\lambda}{2n} \sum_{j=1}^k \theta_j^2$$

- o la regularización  $l_1$

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \{y_i \log(h_{\theta}(x_{i\bullet})) + (1 - y_i) \log(1 - h_{\theta}(x_{i\bullet}))\} + \frac{\lambda}{n} \sum_{j=1}^k |\theta_j|$$

- o su combinación en Elastic net.



# Regresión logística en scikit-learn

Ya vimos que para implementar la regresión logística en scikit-learn, se puede usar la clase `LogisticRegression` del submódulo `linear_model`.

```
from sklearn.linear_model import LogisticRegression
```

# Regresión logística en scikit-learn

Ya vimos que para implementar la regresión logística en scikit-learn, se puede usar la clase `LogisticRegression` del submódulo `linear_model`.

```
from sklearn.linear_model import LogisticRegression
```

- `LogisticRegression` admite el parametro `penalty` que puede tomar el valor `'l2'`, `'l1'`, `'elasticnet'` o `'none'`.

# Regresión logística en scikit-learn

Ya vimos que para implementar la regresión logística en scikit-learn, se puede usar la clase `LogisticRegression` del submódulo `linear_model`.

```
from sklearn.linear_model import LogisticRegression
```

- `LogisticRegression` admite el parametro `penalty` que puede tomar el valor `'l2'`, `'l1'`, `'elasticnet'` o `'none'`.
- En lugar de implementar el parámetro `alpha`, usa el parámetro `C` que es la inversa de  $\lambda$  (`alpha`).

# Regresión logística en scikit-learn

Ya vimos que para implementar la regresión logística en scikit-learn, se puede usar la clase `LogisticRegression` del submódulo `linear_model`.

```
from sklearn.linear_model import LogisticRegression
```

- `LogisticRegression` admite el parametro `penalty` que puede tomar el valor `'l2'`, `'l1'`, `'elasticnet'` o `'none'`.
- En lugar de implementar el parámetro `alpha`, usa el parámetro `C` que es la inversa de  $\lambda$  (`alpha`).
- Por defecto, `LogisticRegression` usa la regularización `'l2'` con `C=1`!

# Regresión logística en scikit-learn

Ya vimos que para implementar la regresión logística en scikit-learn, se puede usar la clase `LogisticRegression` del submódulo `linear_model`.

```
from sklearn.linear_model import LogisticRegression
```

- `LogisticRegression` admite el parametro `penalty` que puede tomar el valor `'l2'`, `'l1'`, `'elasticnet'` o `'none'`.
- En lugar de implementar el parámetro  $\alpha$ , usa el parámetro `C` que es la inversa de  $\lambda$  ( $\alpha$ ).
- Por defecto, `LogisticRegression` usa la regularización `'l2'` con `C=1`!
- También admite el parámetro `l1_ratio` si `penalty='elasticnet'`.