# La tarea de clasificación en detalle (I)

## Índice



## Referencia principal

- Cap. 2 de "Hands-On Machine Learning with Scikit-Learn and TensorFlow"
- Cap. 3 de "Hands-On Machine Learning with Scikit-Learn and TensorFlow"
- Cap. 4 de "Hands-On Machine Learning with Scikit-Learn and TensorFlow"

`(Todas las imágenes pertenecen a dicha referencia salvo que se indique lo contrario)`

---

Al terminar este tema conoceremos los fundamentos matemáticos que hacen posible una solución ML así como las medidas de rendimiento.

# 1. Formulación del problema de optimización

En general, la mayor parte de las tareas de ML se pueden formular como un problema de optimización. Aprender consiste en encontrar el conjunto de parámetros que minimiza una cierta **función de coste o pérdida**, o bien maximiza una **función de utilidad**; es decir que optimiza una cierta función. Puesto que minimizar y maximizar son tareas equivalentes, en adelante utilizaremos los términos 'minimizar' y 'función de coste'.

Como se explicaba en la semana 2, dichos parámetros pertenecen al modelo elegido. Cada modelo suele tener su propio conjunto de parámetros. Por ese motivo, para poder explicar con un ejemplo-guía los conceptos de este tema, debemos elegir un método de clasificación. Por su sencillez, de nuevo utilizaremos el modelo lineal, que obtendremos mediante regresión.

## 1.1. Regresión lineal y logística

### 1.1.1. Regresión Lineal

La regresión lineal tiene como objetivo encontrar la recta (hiperplano si hay $n$ dimensiones) que mejor modela un conjunto de puntos dado. Dicho conjunto consiste en pares $(\mathbf{x}, y)$, y por tanto el objetivo es generar una predicción $\hat{y}$ con el menor error posible cuando recibamos un $\mathbf{x}$.

En la Figura 1, a la izquierda, se muestra un ejemplo típico, cuando $\mathbf{x}$ tiene 1 sola dimensión, o sea $\mathbf{x} = [x_1]$, donde los puntos azules son los ejemplos dados y la línea roja el modelo lineal aprendido. Dicho modelo es la ecuación de la recta, definida por dos parámetros: la 'pendiente' de la recta, que es el coeficiente que acompaña a la variable, y la 'intercepción' con el eje vertical, que es el término independiente, y en inglés se suele denominar *bias*. Formalmente:

$$\hat{y} = w_0 + w_1 x_1$$

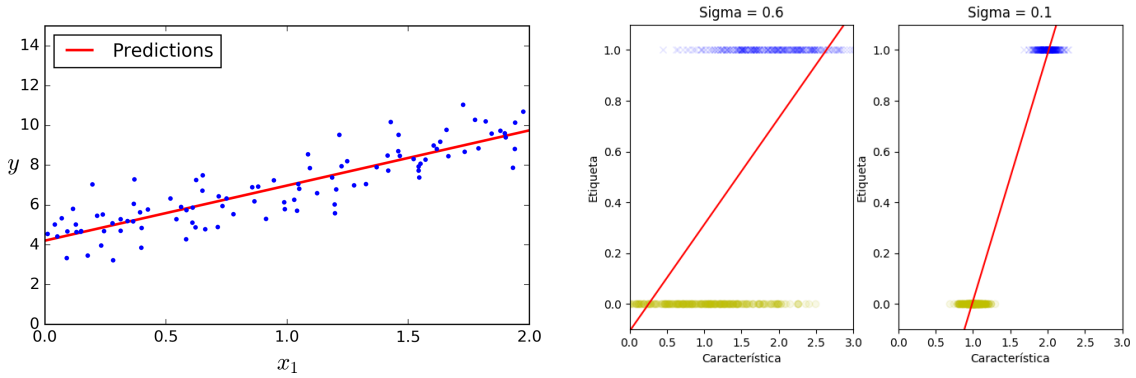


Figura 1: (Izq.) Regresión lineal con 1 sola dimensión. (Der.) ¿Tiene sentido la regresión lineal para hacer clasificación binaria? [Fuente: Original de A. Cuesta]

Si tratamos el problema de clasificación como el problema de regresión entonces $y$ sólo puede tomar dos valores posibles; y al resolverlo obtendríamos un resultado similar al mostrado en la Figura 1 a la derecha. En ella se muestra un ejemplo de juguete, en el cual los círculos amarillos están distribuidos con normalmente con media $\mu = 1$ y las aspas azules con media $\mu = 2$; y ambos con desviación $\sigma$. A simple vista, la linea roja no parece discriminar muy bien.

¿¿Cómo funciona entonces??

Debemos comprender que el modelo lineal (la línea roja) obtenido por regresión NO devuelve directamente la etiqueta en un problema de clasificación. Como en todo problema de regresión, para un ejemplo dado $\mathbf{x}$ el modelo predice (o estima) un valor $\hat{y} \in \mathbb{R}$. Es decir, necesitamos

construir una **función discriminante** que convierta $\hat{y}$ en una predicción de la etiqueta $\hat{t}$. Utilizamos la '$t$' por el término en inglés *tag* ya que ahora la '$y$' representa el resultado que devuelve el modelo lineal. Una vez se avanza en el curso y se comprende la diferencia bien, o cuando se consultan libros, es usual ver que la '$y$' se utiliza para las etiquetas.

El discriminante se construye añadiendo un un umbral $\theta$, por debajo del cual asignaríamos una etiqueta y por encima la otra. Así, en el ejemplo de la Figura 1, la asignación de la etiqueta 0 ó 1 se decidiría según la siguiente expresión:

$$\hat{t} = \begin{cases} \text{“etiqueta 0”} & \text{si } \hat{y} < \theta \\ \text{“etiqueta 1”} & \text{si } \hat{y} \geq \theta \end{cases} \quad \text{donde } \hat{y} = w_0 + w_1 x_1 = [w_0, w_1] \begin{bmatrix} 1 \\ x_1 \end{bmatrix} = \mathbf{w}^T \mathbf{x} \tag{1}$$

El resultado con dicho discriminante se muestra en la Figura 2, donde se han resaltado los fallos que se producen cuando seleccionamos un umbral $\theta$ demasiado bajo y como, según lo modificamos, logramos reducirlos o incluso eliminarlos, ya que este conjunto es linealmente separable.

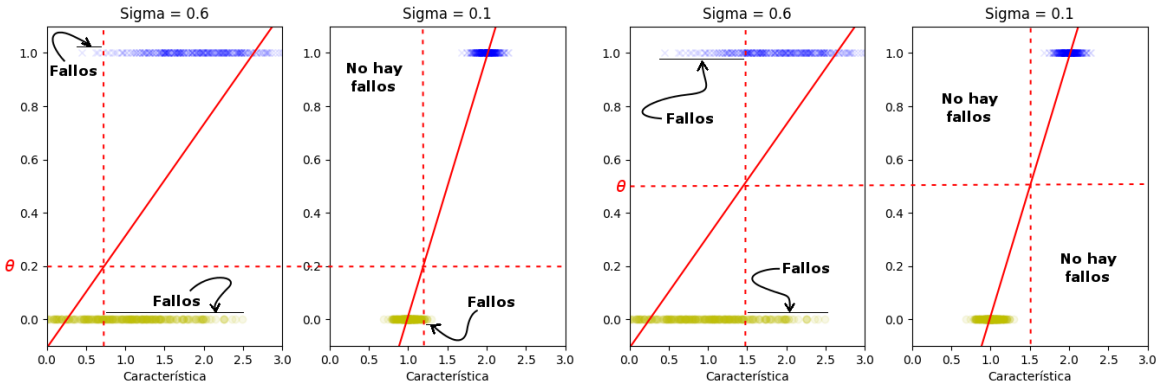


Figura 2: Variando el umbral logramos separar mejor las dos clases, aunque el conjunto con $\sigma$ =0.6 no es linealmente separable porque las dos clases se solapan. [Fuente: Original de A. Cuesta]

En resumen, hemos comprendido que la regresión lineal también sirve para clasificar, transformando la ecuación de la superficie de separación (la recta) en la función discriminante (1).

Si el número de etiquetas es dos (o finito) no tiene mucho sentido utilizar el eje vertical de las figuras para representar la etiqueta. Es mejor utilizar diferentes marcadores o colores para los ejemplos de diferentes clases. Así podemos el eje vertical queda libre y lo podemos usar para visualizar dos características. De este modo, las Figuras 1(Der.) y 2 se podrían representar sólamente en el eje horizontal, tal y como se muestra en la Figura 3.

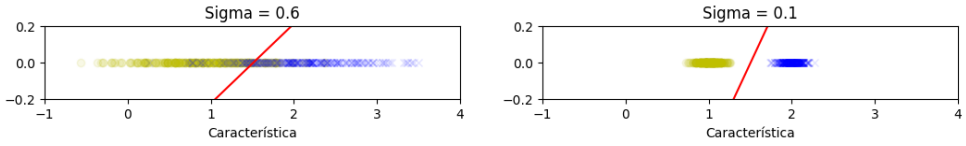


Figura 3: Representación de ambas clases sobre una sóla dimensión, ya que sólo hay una característica. El discriminante ha sido obtenido con regresión lineal. [Fuente: Original de A. Cuesta]

### 1.1.2. Regresión Logística

El discriminante de la expresión (1) nos devuelve la etiqueta estimada para un ejemplo, pero ¿cuánto podemos confiar en que dicha estimación sea correcta?. En otras palabras, un valor de $\hat{y} \gg 0$ nos da más seguridad que $\hat{y} = 10^{-10}$ a la hora de afirmar que $\hat{t}$ = “etiqueta 1” porque está más lejos de la superficie de decisión.

Podemos 'normalizar' la confianza al intervalo $[0, 1]$ si lo modelamos como la probabilidad de dicha etiqueta. La regresión logística nos devuelve precisamente dicha probabilidad. Después nosotros ajustamos el umbral de incertidumbre para decidir cuando una predicción es suficientemente 'segura', o sea, probable.

Para ello, se utiliza la función sigmoide $S(z) = 1/\left(1 + e^{-z}\right)$, que modula la respuesta del clasificador lineal entre 0 y 1.

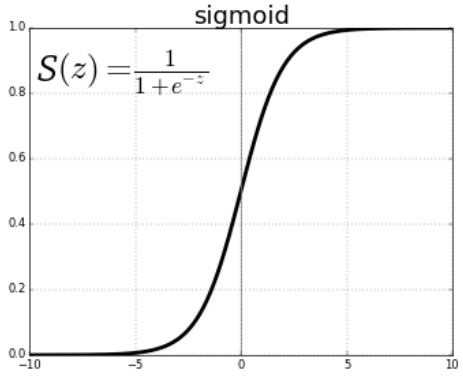


Figura 4: Función sigmoide. [Fuente: Anónimo @ Internet]

Por tanto, en regresión logística, el resultado es una estimación de la probabilidad, $\hat{p}$, que da lugar a la siguiente función discriminante:

$$\hat{t} = \begin{cases} \text{"etiqueta 0"} & \text{si } \hat{p} < \theta \\ \text{"etiqueta 1"} & \text{si } \hat{p} \geq \theta. \end{cases} \quad \text{donde } \hat{p} = S\left(\hat{y}\right) = S\left(\mathbf{w}^T\mathbf{x}\right) = \frac{1}{1 + \exp\left(-\mathbf{w}^T\mathbf{x}\right)} \tag{2}$$

Si $\hat{y} \gg 0$, entonces $\hat{p} \simeq 1$, y asignamos la etiqueta "etiqueta 1" con mucha seguridad porque es muy probable. Pero si $\hat{y} \ll 0$, entonces $\hat{p} \simeq 0$, es decir la etiqueta "etiqueta 1" es muy improbable. Como se trata de una clasificación binaria, esto significa que la etiqueta contraria, "etiqueta 0", es muy probable.

Finalmente, es importante resaltar que tanto en (1) como en (2) aparece un umbral $\theta$, pero donde cobra un sentido más universal (es decir que sirve para cualquier problema) es en (2). La interpretación probabilística de los métodos de ML añade dificultad pero tiene grandes ventajas.

### 1.1.3. Generalización a $n$ dimensiones

Las expresiones (1) y (2) son generales para cualquier dimensión de $\mathbf{x}$, pero conviene recordar brevemente cual es el aspecto de estos vectores 'desplegados'.

En $n$ dimensiones, el vector de características 'ampliado' es $\mathbf{x} = [1, x_1, x_2, \ldots, x_n]^T$, mientras que el vector de pesos es $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]^T$. El superíndice $\cdot^T$ indica la trasposición, es decir que tanto $\mathbf{x}$ como $\mathbf{w}$ son vectores columna.

Si, en vez de un sólo ejemplo, tenemos un conjunto de datos, como ya hemos explicado, normalmente los ejemplos se apilan en las filas de una matriz. Entonces el ejemplo $i$-esimo, ampliado, sería $\mathbf{x}^{(i)} = [1, x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T$, y el conjunto de $m$ ejemplos sería $\mathbf{X} = [\mathbf{x}^{(1)T}, \ldots, \mathbf{x}^{(m)T}]^T$.

Es decir, los vectores son siempre vectores columnas. Sin embargo, cuando representamos el conjunto de datos, como hemos dicho varias veces, cada vector de características es una fila.

En resumen, con la notación generalizada a $n$ dimensiones podemos resumir la regresión lineal y logística en la siguiente tabla.

Tabla 1: Tabla resumen de clasificación mediante regresión.

| **Reg. Lineal** | **Reg. Logística** |
|---|---|
| Conjunto de $m$ ejemplos $n$ dimensionales etiquetados | |
| $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n]^T$ | |
| $\mathbf{x}^{(i)} = [1, x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T$ | |
| $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} = \begin{bmatrix} 1, & x_1^{(1)}, & x_2^{(1)}, & \ldots & x_n^{(1)} \\ 1, & x_1^{(2)}, & x_2^{(2)}, & \ldots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1, & x_1^{(m)}, & x_2^{(m)}, & \ldots & x_n^{(m)} \end{bmatrix}$ | |
| $\hat{y}^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)}$ | $\hat{p}^{(i)} = 1/(1+\exp(-\mathbf{w}^T\mathbf{x}^{(i)})$ |
| $\hat{t}^{(i)} = \begin{cases} \text{“etiqueta 0”} & \text{si } \hat{y} < \theta \\ \text{“etiqueta 1”} & \text{si } \hat{y} \geq \theta \end{cases}$ | $\hat{t}^{(i)} = \begin{cases} \text{“etiqueta 0”} & \text{si } \hat{p}^{(i)} < \theta \\ \text{“etiqueta 1”} & \text{si } \hat{p}^{(i)} \geq \theta \end{cases}$ |

## 1.2. Función de coste

En la Tabla 1 los elementos del vector de pesos $\mathbf{w}$ son los parámetros del modelo que debemos encontrar dado el conjunto de datos etiquetados $\{\mathbf{X}, \mathbf{t}\}$ para que el error cometido en las predicciones sea lo menor posible.

Dicho error se define a partir de una función de coste $J(\mathbf{w}, \mathbf{X}, \mathbf{t})$, que depende del vector de pesos $\mathbf{w}$, del conjunto de datos $\mathbf{X} = \{\mathbf{x}^{(i)}\}$ y de las etiquetas $\mathbf{t} = \{t^{(i)}\}$, con $i = 1, 2, \ldots, n$.

En problemas de regresión hay tres funciones típicas: el error absoluto medio (MAE), el error cuadrático medio (MSE) y la raíz cuadrada positiva del anterior (RMSE).

$$\text{MAE} = \left| \frac{1}{m} \sum_{i=1}^{m} \left( \hat{y}^{(i)} - y^{(i)} \right) \right| , \; \text{MSE} = \frac{1}{m} \sum_{i=1}^{m} \left( \hat{y}^{(i)} - y^{(i)} \right)^2 , \; \text{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left( \hat{y}^{(i)} - y^{(i)} \right)^2} ,$$

donde $y^{(i)}$ es el valor real, conocido porque el conjunto de datos es etiquetado, asociado al ejemplo $i$-esimo; mientras que $\hat{y}^{(i)}$ es la predicción.

Para clasificación mediante regresión lineal podríamos sustituir la $y^{(i)}$ por la etiqueta $t^{(i)}$. Pero si utilizamos la regresión logística es preferible utilizar la función *log-loss* $L(\mathbf{w}, \mathbf{X}, \mathbf{t})$,

$$L(\mathbf{w}, \mathbf{X}, \mathbf{t}) = -\frac{1}{m} \sum_{i=1}^{m} (t^{(i)}) \log(\hat{p}^{(i)}) + (1 - t^{(i)}) \log(1 - \hat{p}^{(i)}).$$

En este punto es matemáticamente conveniente que la “etiqueta 0” $= 0$ y que “etiqueta 1” $= 1$. Así, la función *log-loss* es el promedio del logaritmo de la probabilidad de observar cada una de las etiquetas bajo la distribución estimada de manera independiente unas de otras (Anexo 1).

En cualquier caso, se utilice la función de coste que se utilice, el objetivo es siempre el mismo: Encontrar el vector de pesos óptimo $\mathbf{w}^*$ que minimiza la función de coste $J(\mathbf{w}, \mathbf{X}, \mathbf{t})$. Formalmente:

$$\mathbf{w}^* = \arg\min_{w} \left( J(\mathbf{w}, \mathbf{X}, \mathbf{t}) \right).$$

## 1.3. Método de optimización

El problema de optimización planteado es

$$\mathbf{w}^* = \arg\min_w \left( J(\mathbf{w}, \mathbf{X}, \mathbf{t}) \right).$$

El modo 'teórico/analítico' general de resolverlo consiste en derivar respecto de la variable argumento e igualar a cero para buscar óptimos, es decir

$$\frac{\partial J(\mathbf{w}, \mathbf{X}, \mathbf{t})}{\partial w_j} = 0, \quad \text{con} \quad j = 1, 2, \ldots, n.$$

Sin embargo, lo normal es NO encontrar una fórmula cerrada que nos resuelva el problema, por lo que debemos acudir a métodos numéricos.

### 1.3.1. Fórmula cerrada

En regresión lineal SÍ es posible encontrar una fórmula cerrada.

Encontrar el vector de pesos es tan sencillo como resolver el siguiente sistema lineal:

$$\begin{bmatrix} 1, & x_1^{(1)}, & x_2^{(1)}, & \ldots & x_n^{(1)} \\ 1, & x_1^{(2)}, & x_2^{(2)}, & \ldots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1, & x_1^{(m)}, & x_2^{(m)}, & \ldots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} w_0^* \\ w_1^* \\ w_2^* \\ \vdots \\ w_n^* \end{bmatrix} = \begin{bmatrix} t^{(1)} \\ t^{(2)} \\ \\ \vdots \\ t^{(m)} \end{bmatrix}; \quad \text{es decir,} \quad \mathbf{X}\mathbf{w}^* = \mathbf{t}.$$

Despejando $\mathbf{w}^*$ tenemos que:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{t}$$

### 1.3.2. Descenso del gradiente

Cuando no es posible encontrar una fórmula cerrada, o requiere mucho esfuerzo, o el número de óptimos locales es muy grande, debemos recurrir a métodos numéricos que devuelvan o el óptimo $\mathbf{w}^*$, o una solución 'bastante buena'.

Los métodos numéricos funcionan iterativamente, actualizando sucesivamente el valor del vector de pesos hasta que se cumpla un criterio de parada. Actualizar significa añadir o quitar (o sea añadir algo 'negativo') algo, representado por $\mathbf{g}$, a lo que tenemos; de modo que, si representamos por $\mathbf{w'}$ al nuevo vector de pesos respecto del actual $\mathbf{w}$, entonces $\mathbf{w'} = \mathbf{w} + \mathbf{g}$. Y una vez se detenga el método, $\mathbf{w}^* = \mathbf{w'}$.

En otras palabras, en cada iteración $\mathbf{g}$ debería mejorar un poco el vector $\mathbf{w}$. Pero esto sólo se logra si vamos recorriendo la superficie de $J(\mathbf{w})$ en sentido descendente[1] según modificamos $\mathbf{w}$. $\mathbf{X}$ y $\mathbf{t}$ se han quitado de $J$ porque son datos. No es que la función $J$ no dependa de ellas, sino que, una vez dados, nosotros ya sólo podemos modificar $\mathbf{w}$. Para hacer más explícita esta diferencia, a partir de ahora escribiremos $J(\mathbf{w}; \mathbf{X}, \mathbf{t})$.

El gradiente de una función $f$ indica la dirección en la cual $f$ varía más rápidamente, y se construye con la derivada de $f$ en cada una de sus componentes. Es decir,

$$\nabla f(x_1, \ldots, x_m) = \left[ \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m} \right]$$

Por tanto, si nos movemos en la dirección $-\nabla J(\mathbf{w})$ nos aseguramos el descenso ($\nabla J(\mathbf{w})$ apunta 'hacia arriba') más rápido. En la Figura 5 se muestra un ejemplo de como se avanza por la función de coste $J$ siguiendo el gradiente.

[1] Recordar que, salvo que se diga lo contrario, estamos minimizando la función de coste.

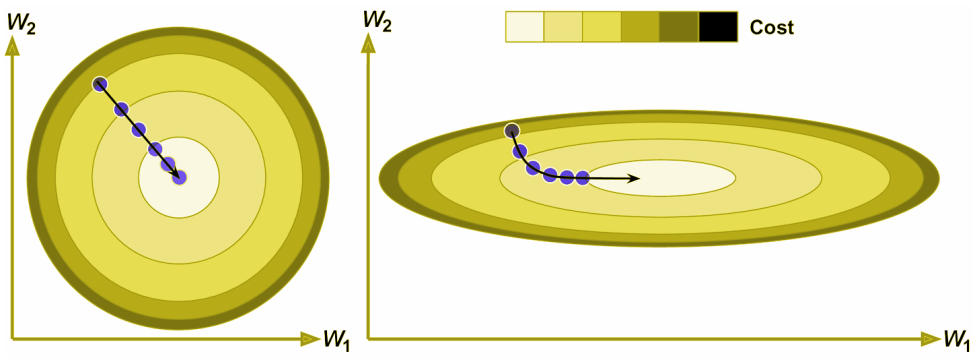


Figura 5: Los puntos azules son sucesivas actualizaciones de pesos $\mathbf{w} = [w_1, w_2]$ por descenso del gradiente en dos funciones de coste $J(\mathbf{w})$ distintas, representadas por sus curvas de nivel.

La cantidad de avance en una dirección vendrá dada por el módulo del gradiente en dicho punto, pero se matiza con el **ratio de aprendizaje** $\eta$, cuyo efecto se muestra en la Figura 6. Como vemos, es difícil ajustar el ratio. No se puede dar una norma general salvo ejecutar varias veces la optimización con varios arranques diferentes, lo cual es recomendable este y cualquier otro procedimiento iterativo.

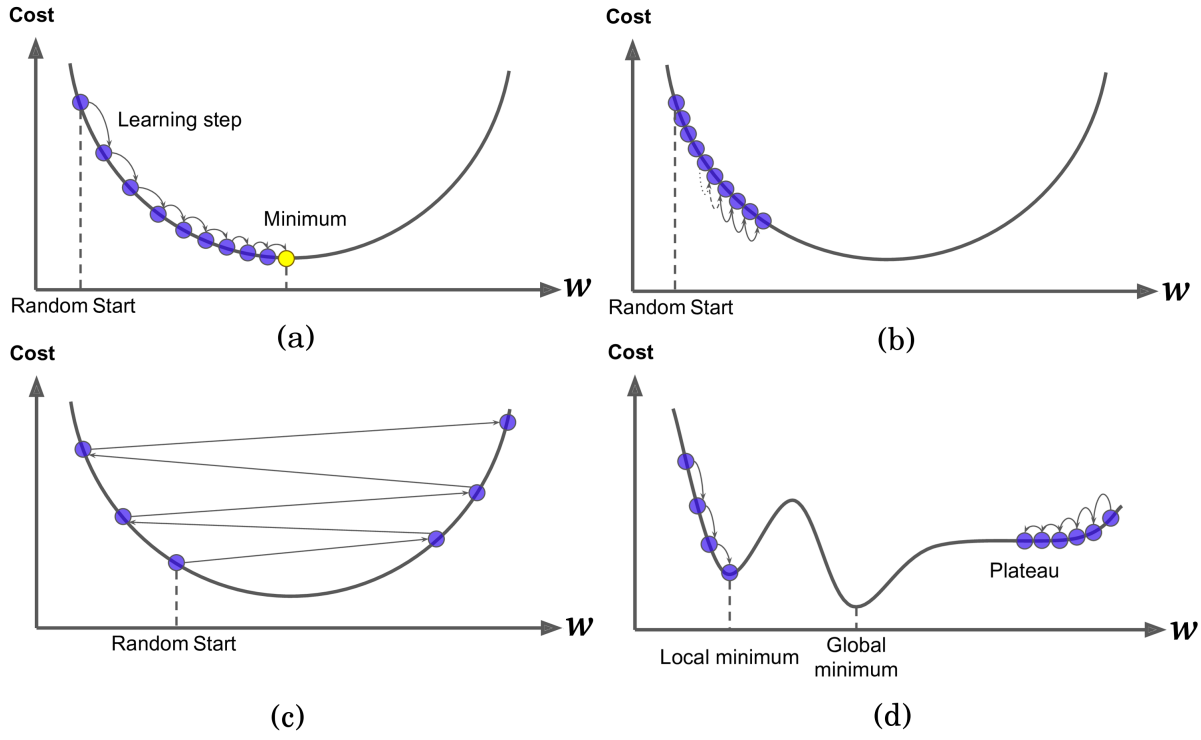


Figura 6: Efecto del ratio de aprendizaje $\eta$ en el descenso del gradiente para $f(x)$. (a) Con un buen ratio, (b) con un ratio demasiado pequeño, (c) con uno demasiado grande, y (d) cuando $J$ tiene varios mínimos locales y mesetas (*plateau*).

En definitiva, el descenso de gradiente proporciona la siguiente regla de actualización de pesos

$$\mathbf{w}' = \mathbf{w} - \eta \nabla J(\mathbf{w})$$

## 1.4. Criterio de parada

El más sencillo es aquel en el que se cumple alguno de las dos condiciones siguientes:

- Alcanzar un cierto número de iteraciones prefijado.

  → Esta condición se impone como medida de seguridad, para no quedar atrapados en un bucle infinito.

- La distancia media entre los últimos $N$ pesos actualizados es menor de un cierto valor de tolerancia.

  → Si se cumple esta condición entonces se dice que hemos convergido a un mínimo.

La segunda condición puede plantear problemas cuando, al recorrer $J$ entramos en una meseta (*plateau*) como la mostrada en la Figura 6(d), donde el gradiente es casi 0 en una vecindad.

Una alternativa muy importante consiste en detener el proceso de aprendizaje en el momento en el que obtengamos el modelo que menos errores comete en el conjunto de validación. Como se puede ver en la Figura 7, según avanza el proceso de aprendizaje y validación es esperable que se reduzcan los errores. Sin embargo llegará un punto en el que, de tanto usar los mismos datos, empecemos a aprender también el conjunto de validación. A partir de ahí estaremos sobreajustando el modelo, por lo que debemos retroceder a aquel modelo con el que obtuvimos los mejores resultados en el conjunto de validación. Esta técnica se denomina *early stopping*.

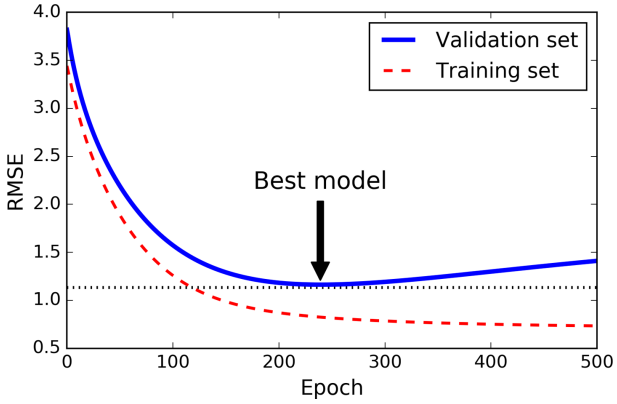


Figura 7: Utilización del conjunto de validación para decidir cuando debemos parar de aprender.

## 1.5. Regularización

A la función de coste, a veces, se le añaden términos que dependen sólo del vector de pesos $\mathbf{w}$, denominados **términos de regularización** .

Al añadirse a la función que estamos minimizando lo que logramos es aprender al mismo tiempo a clasificar los datos y a mantener los pesos lo más pequeños posibles.

Junto con el *early stop*, la regularización también sirve para evitar el sobreajuste de los modelos.

Ahora bien, ¡¡ sólo se utiliza el término de regularización con el conjunto de entrenamiento !! Una vez obtenidos los pesos, tanto la validación como el test se realizan con la función de coste sin los términos de regularización.

A continuación se detallan tres términos muy comunes.

### 1.5.1. *Ridge*

$$J_{\mathrm{R}}(\mathbf{w};\mathbf{X},\mathbf{t}) = J(\mathbf{w};\mathbf{X},\mathbf{t}) + \frac{\alpha}{2}\sum_{j=1}^{n} w_j^2, \quad \text{con} \quad \alpha \geq 0.$$

Una ventaja de la regularización *ridge*, sobre los dos términos siguientes, es que tiene hay fórmula cerrada para obtener el vector de pesos óptimo, dada por la expresión:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{t}$$

### 1.5.2. *Lasso*

$$J_{\mathrm{L}}(\mathbf{w};\mathbf{X},\mathbf{t}) = J(\mathbf{w};\mathbf{X},\mathbf{t}) + \alpha\sum_{j=1}^{n} |w_j|, \quad \text{con} \quad \alpha \geq 0.$$

*Lasso* tiende a eliminar completamente los pesos de las características menos 'importantes'. Es decir que realiza una selección de características y devuelve un modelo disperso, o sea que sólo unos pocos pesos distintos de 0.

### 1.5.3. *Elastic Net*

$$J_{\mathrm{EN}}(\mathbf{w};\mathbf{X},\mathbf{t}) = J(\mathbf{w};\mathbf{X},\mathbf{t}) + (r)J_{\mathrm{L}}(\mathbf{w};\mathbf{X},\mathbf{t}) + (1-r)J_{\mathrm{R}}(\mathbf{w};\mathbf{X},\mathbf{t}), \quad \text{con} \quad 0 \leq r \leq 1.$$

*Elastic Net* combina un cierto porcentaje de *Lasso* y el resto, hasta llegar al 100 %, de *Ridge*.

En general es preferible elegir *Elastic Net* frente a *Lasso* porque *Lasso* se puede volver errático varias características están fuertemente correlacionadas.

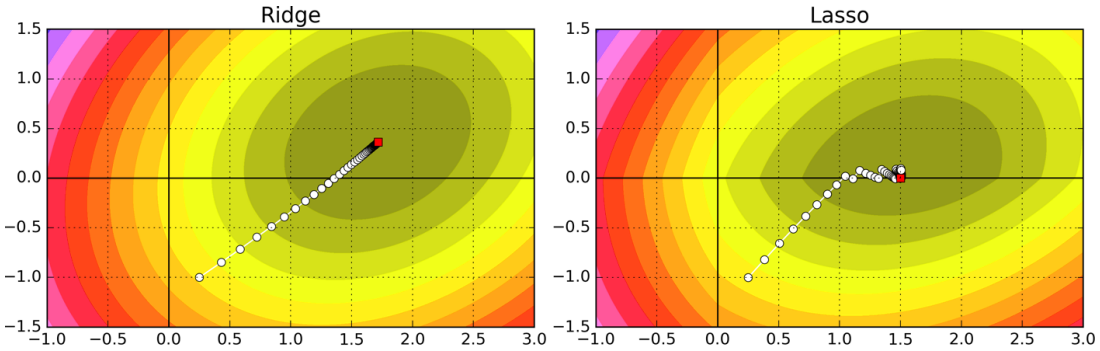


Figura 8: Descenso del gradiente en 2 dimensiones con regularización *Ridge* y *Lasso*

## 2. Ajuste fino de los modelos

### 2.1. Parámetros vs. Hiperparámetros

- Los **parámetros** son los 'elementos' que configuran el modelo.

  *Ej.*: En el ejemplo-guía de la semana pasada teníamos los pesos, $\{w_0, w_x, w_y\}$, que determinaban el clasificador lineal.

  *Ej.*: En el ejemplo unidimensional de esta semana, sólo hay dos pesos, que a partir de ahora se representan como los componentes del vector $\mathbf{w} = [w_0, w_1]$.

- Los **hiperparámetros** son los 'elementos' que configuran el proceso de aprendizaje.

  *Ej.*: Parámetros de algoritmo, que configuran el proceso de aprendizaje, como el valor de $\alpha$ o de $r$ cuando se añade regularizacion.

  *Ej.*: Parámetros que configuran la tarea de clasificación, como el número de *folds* que hacemos en validación cruzada, o que configuran el proceso de ingeniería de características, como el umbral que se utilizaba para construir los atributos y posteriormente las características en el ejemplo-guía de la semana pasada.

En resumen, los hiperparámetros configuran el problema de optimización e incluso los datos que va a recibir; y los parámetros son la solución que obtenemos al resolverlo. Entonces, ¿cómo obtenemos el valor óptimo de los hiperparámetros?. Intuitivamente, se podría pensar en incluirlos también en el problema de optimización o bien en montar un problema de optimización para ellos. Ambas soluciones son actualmente fuente de investigación[2]. En la práctica podemos recurrir a una búsqueda ordenada (*grid search*) o una búsqueda aleatoria (*randomized search*)

### 2.2. Búsqueda de hiperparámetros

**Búsqueda ordenada** El término 'ordenado' intenta traducir *grid*, que es más descriptivo. Se trata de construir un bucle `for` barriendo un conjunto de valores por cada hiperparámetro que deseamos optimizar.

**Búsqueda aleatoria** La búsqueda ordenada puede funcionar cuando tenemos uno o dos hiperparámetros, pero evidentemente escala muy mal. Si barremos $N$ valores por cada hiperparámetros, y tenemos $K$ hiperparámetros necesitamos realizar $N^K$ entrenamientos y validaciones para elegir la mejor combinación.

Alternativamente podemos fijar un cierto número $M$ de combinaciones aleatorias con los $K$ hiperparámetros. De esta manera tenemos un coste computacional limitado a $M$.

**Búsqueda heurística** Parte de la investigación en esta área ha propuesto hace tiempo utilizar métodos heurísticos (colonias de hormigas, algorítmos genéticos, busqueda en vecindades variables, ...) para encotrar buenas combinaciones de hiperparámetros, pero quedan fuera de esta asignatura.

**Métodos de Ensamblado** Se trata de combinar los modelos que funcionan mejor. Son un metodo muy potente en el cual se han basado algunos de los ganadores de retos de Kaggle, y que estudiaremos más adelante.

## 3. Medidas de bondad de los resultados

Cuando se trata de clasificación binaria, como hasta ahora, pertenecer a una clase implica no pertenecer a la otra. Por tanto, en vez de predecir si un ejemplo pertenece a una clase u otra, podemos predecir si pertenece a una clase o no pertenece. Cuando lo afirmamos decimos que la predicción es **positiva**, y cuando lo negamos es **negativa**. Ahora bien, una cosa es nuestra predicción y otra la verdad, que conocemos pues se tratan de ejemplos etiquetados.

De este modo, hay dos tipos de aciertos: Positivos verdaderos (*True Positives*, TP) y Negativos verdaderos (*True Negatives*, TN). Y del mismo modo, hay dos tipos de fallos: Falsos positivos (*False Positives*, FP) y Falsos negativos (*False Negatives*, FN).

[2]Si somos capaces de plantear un problema de optimización que devuelve el conjunto óptimo de hiperparámetros, el proceso de ML sería prácticamente automático a partir de los datos 'en crudo'.

## 3.1. Tabla de confusión

Calcular TP, TN, FP y FN es tan sencillo como contar el número de veces que ocurre cada caso con el conjunto de validación mientras se aprende, y con el de test cuando hemos llegado a un modelo y lo queremos comparar con otros.

Con estos valores se forma la matriz de confusión y se pueden calcular una serie de medidas derivadas de ellos. La Figura 9 muestra un resumen gráfico.

| | | **Condition (as determined by "Gold standard")** | | | |
|---|---|---|---|---|---|
| | Total population | Condition positive | Condition negative | Prevalence = Σ Condition positive / Σ Total population | |
| **Test outcome** | Test outcome positive | **True positive** | **False positive** (Type I error) | Positive predictive value (PPV, Precision) = Σ True positive / Σ Test outcome positive | False discovery rate (FDR) = Σ False positive / Σ Test outcome positive |
| | Test outcome negative | **False negative** (Type II error) | **True negative** | False omission rate (FOR) = Σ False negative / Σ Test outcome negative | Negative predictive value (NPV) = Σ True negative / Σ Test outcome negative |
| | Positive likelihood ratio (**LR+**) = TPR/FPR | True positive rate (TPR, Sensitivity, Recall) = Σ True positive / Σ Condition positive | False positive rate (FPR, Fall-out) = Σ False positive / Σ Condition negative | Accuracy (ACC) = Σ True positive + Σ True negative / Σ Total population | |
| | Negative likelihood ratio (**LR−**) = FNR/TNR | False negative rate (FNR) = Σ False negative / Σ Condition positive | True negative rate (TNR, Specificity, SPC) = Σ True negative / Σ Condition negative | | |
| | Diagnostic odds ratio (**DOR**) = LR+/LR− | | | | |

Figura 9: Tabla de confusión en clasificación binaria y medidas derivadas. [Fuente: Wikipedia]

## 3.2. Otras medidas derivadas

Como hemos visto, el umbral determina la función discriminante y, en consecuencia, los aciertos y fallos que se producen. Ahora que sabemos clasificar estos fallos vamos a ver otras medidas derivadas de ellos

### 3.2.1. *Precision*, *Recall* y *F-score*

**Precision** Es el término en inglés que representa el porcentaje de aciertos entre todos los etiquetados como positivos. En términos de las celdas de la matriz de confusión:

$$\text{Precision} = \frac{TP}{TP + FP}$$

**Recall** Es el término en inglés que representa el porcentaje de aciertos entre todos los ejemplos que están etiquetados como positivos. En Español el término usado es 'Sensitividad'; y su expresión, en términos de la matriz de confusión es:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Un clasificador capaz de lograr un 100 % de *precision* o de *recall* no significa necesariamente que sea bueno. En la Figura 10 se muestra un ejemplo de cada caso. En el dibujo del centro podemos

ver que los 2 ejemplos verdes etiquetadas serán etiquetados como positivos porque están en el lado 'positvo' y además no hay ningún circulo rojo.

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{2}{2+0} = 1.$$

En el dibujo de la derecha podemos ver que todos los ejemplos verdes serán etiquetados como verdes, aunque algunos rojos también lo serán. Entonces:

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{6}{6+0} = 1.$$

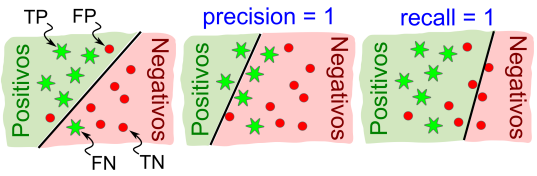


Figura 10: Un clasificador puede lograr un *precision* o un *recall* del 100 %. Basta con que no haya ningún FP o ningún FN respectivamente. [Fuente: Original de A. Cuesta]

**F-score** Es una medida de compromiso entre ambas, calculada como su media harmónica.

$$\text{F} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

**¿Qué medida se debe usar?** El F-score favocere aquellos clasificadores que tengan valores de *precision* y *recall* similares. Sin embargo habrá ocasiones en las que prefiramos favorecer el *precision* y otras en las que prefiramos el *recall*. Por ejemplo, si queremos filtrar videos inapropiadas para niños, preferimos no tener ningún FP a tener algún FN. Es decir no nos importa equivocarnos en algún video apto si nos aseguramos de que no pasa ninguno no-apto. Por otro lado, si queremos detectar actividades sospechosas en videos, no queremos que ningún video sospechos sea etiquetado erroneamente. El objetivo de este clasificador será eliminar el mayor número posible de videos no-sospechosos para facilitar la tarea del humano (policía, jefe de seguridad...)

Otra manera de llegar a un compromiso entre *precision* y *recall* consiste pintar sus valores según modificamos $\theta$, tal y como se muestra en la Figura 11.

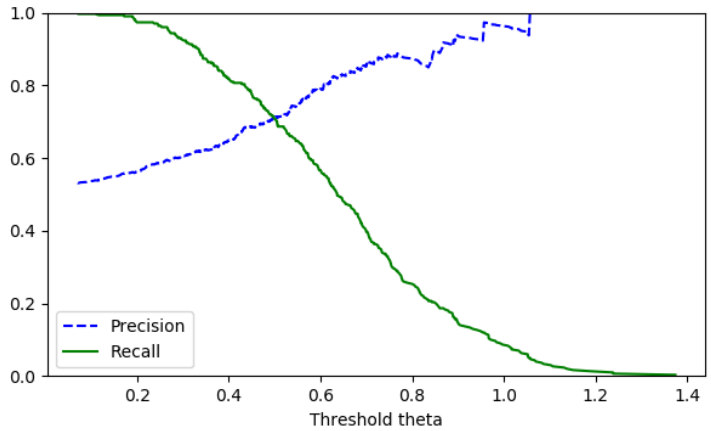


Figura 11: Curvas de *precision* y *recall* para el conjunto de datos de la Figura 1 con $\sigma = 0{,}6$. [Fuente: Original de A. Cuesta]

### 3.2.2. Curva ROC

La curva ROC (del inglés *Receiver Operation Characteristic*, ya que fue desarrollada en el ámbito de la electrónica y las comunicaciones) se forma del siguiente modo:

1. Calcular el valor de TPR y FPR, cuya expresión está en la Figura 9, para una serie de valores del umbral.

   *Ej.*: Si estamos utilizando regresión logística, sabemos que el umbral variará entre 0 y 1, así que podemos hacer un barrido de 100 muestras.

2. Pintar los pares (FPR,TPR) obtenidos.

El resultado es una curva similar a la de la Figura 12. Si la curva se aproxima a la diagonal significa que el clasificador tiene la misma eficacia que decidir al azar lanzando una moneda.

La curva ROC será mejor cuanto más vertical suba al punto (0,1). Cuantitativamente hay dos medidas típicas:

- Area bajo la curva (*area under curve*, AUC):

  En el caso ideal la curva ROC subiría verticalmente desde el (0,0) hasta el (0,1) y luego seguiría horizontalmente hasta el (1,1). Por tanto el límite superior para el AUC = 1, y cuanto más próximo mejor.

- Distancia desde el punto (0,1) a la curva.

  Razonando con el caso ideal igual que antes, el mejor valor posible es 0. Por tanto cuanto más pequeña sea esta distancia mejor.

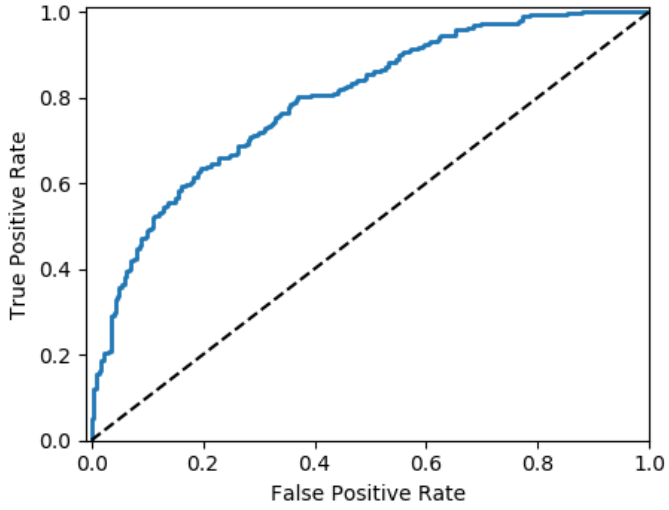


Figura 12: Curvas ROC para el conjunto de datos de la Figura 1 con $\sigma = 0{,}6$. [Fuente: Original de A. Cuesta]

En el tema siguiente ampliaremos los métodos vistos en este tema.

## Anexo 1. Diseño de la función *log-loss*

La función de coste *log-loss* introducida en clasificación binaria con regresión logística es:

$$L(\mathbf{w}; \mathbf{X}, \mathbf{t}) = -\frac{1}{m}\sum_{i=1}^{m}(t^{(i)})\log(\hat{p}^{(i)}) + (1-t^{(i)})\log(1-\hat{p}^{(i)}),$$

donde:

- $\mathbf{w}$ es el vector de pesos.
- $\mathbf{X}$ es una matriz $m \times n$ cuya fila $i$ es el vector de características 'expandido' $\mathbf{x}^{(i)T}$, y con la etiqueta asociada $t^{(i)}$
- $\mathbf{t}$ es el vector de etiquetas.
- $\hat{p}^{(i)}$ es el valor de probabilidad asignado por la función sigmoide al ejemplo ponderado por el vector de pesos.

¿Por qué? ¿De dónde sale?

Si un ejemplo $i$ tiene dos posibles etiquetas, y tenemos una función que me devuelve la probabilidad de una de ellas, $\hat{p}^{(i)}$, entonces la otra etiqueta tiene probabilidad $1-\hat{p}^{(i)}$.

Si no está claro, podemos pensar en que ocurre cuando lanzamos una moneda al aire. Cuando la moneda no está trucada, la probabilidad de obtener cara es del 50 %, igual que la de obtener cruz. O sea que si lanzamos la moneda 100 veces es esperable obtener (casi) 50 caras y 50 cruces. Es decir $p(\text{cara}) = p(\text{cruz}) = 0{,}5$ , y por tanto $p(\text{cara}) + p(\text{cruz}) = 1$. Pero si la moneda estuviera trucada para que salga una cara por cada 9 cruces, entonces si lanzamos la moneda 100 veces es esperable obtener (casi) 10 caras y 90 cruces. Es decir, $p(\text{cara}) = 0{,}1$ y $p(\text{cruz}) = 0{,}9$, y se sigue cumpliendo que $p(\text{cara}) + p(\text{cruz}) = 1$.

La distribución de probabilidad que modela este tipo de sucesos es la **distribución Bernoulli**,

$$p_{\text{Ber}}(k) = p^k(1-p)^{(1-k)} = \begin{cases} (1-p) & \text{si} \quad k=0 \\ p & \text{si} \quad k=1 \end{cases}$$

En este punto es matemáticamente conveniente que la "etiqueta 0" $= 0$ y que "etiqueta 1" $= 1$, de manera que la etiqueta hace las funciones de $k$ en la expresión de arriba.

Es decir, dado un ejemplo $i$, la distribución de probabilidad sobre las dos etiquetas posibles es

$$p_{\text{Ber}}(t^{(i)}) = \left(\hat{p}^{(i)}\right)^{t^{(i)}} \left(1-\hat{p}^{(i)}\right)^{(1-t^{(i)})}.$$

Si tomamos logaritmos en la expresión anterior convertimos el producto en suma y las exponenciales en productos, lo cual es muy ventajoso desde el punto de vista computacional, quedando:

$$\log\left(p_{\text{Ber}}(t^{(i)})\right) = \left(t^{(i)}\right)\log\left(\hat{p}^{(i)}\right) + \left(1-t^{(i)}\right)\log\left(1-\hat{p}^{(i)}\right).$$

El valor negativo de esta función sería el *log-loss* de un ejemplo, pero como hay $m$ en el conjunto calculamos su promedio sumando el *log-loss* de cada uno y dividiendo por $m$. Finalmente,

$$L(\mathbf{w}; \mathbf{X}, \mathbf{t}) = -\frac{1}{m}\sum_{i=1}^{m}(t^{(i)})\log(\hat{p}^{(i)}) + (1-t^{(i)})\log(1-\hat{p}^{(i)}).$$

□

# Índice alfabético