Lab 7. Clasificación con Regresión Logística

Objetivos

En esta práctica vamos a trabajar con el método de clasificación de la regresión logística (Logistic Regression). Vamos a simular un experimento de clasificación y calcular algunas métricas de error del mismo, dividiendo el conjunto de datos en subconjuntos de entrenamiento y test como en la práctica anterior.

El script general para esta práctica será `main_lab7.m`, y en él se irá completando el código para realizar los ejercicios planteados en la práctica.

Para esta práctica emplearemos un *dataset* con datos de masas detectadas en mamografías para determinar si las masas que se detectaron son benignas o malignas. El dataset está descrito en https://archive.ics.uci.edu/ml/datasets/Mammographic+Mass. En este dataset se han eliminado los patrones con *missing values* (patrones en los que no existían todos los datos) y sus datos se normalizaron para que cada variable del conjunto tuviese media 0 y desviación típica 1. Cada elemento del *dataset* (variable de Matlab `X`) se define mediante 5 variables. Las clases se indican en la variable de Matlab `Y` (`Y(i)==0` indica que la masa descrita por el patrón `X(i,:)` es benigna; `Y(i)==1` indica que la masa descrita por el patrón `X(i,:)` es maligna). Estas variables de Matlab se obtienen tras cargar el archivo `mammographic_data_norm.mat`.

Como convención, una vez más los patrones estarán contenidos en las filas de las matrices de datos (es decir, cada patrón está en una fila distinta de la variable `X`).

1) Preliminar: División entre conjuntos de entrenamiento y de test

En la primera parte del script se cargan los datos del dataset y se divide el conjunto en subconjuntos de entrenamiento y de test aleatoriamente.

Al diseñar un clasificador necesitamos conocer el error de la clasificación para evaluar si es bueno (y se puede poner en producción) o malo (y es necesario rediseñarlo o cambiar parámetros del mismo). Esta evaluación consistirá en clasificar una serie de patrones con el clasificador ya entrenado y comparar las predicciones que realiza para ellos con sus clases reales.

Si utilizásemos el conjunto de datos con los que se ha entrenado el clasificador, favoreceríamos las hipótesis que se centran demasiado en los datos de entrenamiento, de modo que podría ocurrir que el clasificador no fuese capaz de generalizar predicciones para patrones desconocidos (nuevos).

Por ello, es necesario evaluar el rendimiento con datos que el clasificador no haya "visto" antes (es decir, que no se hayan utilizado para el entrenamiento). Por ello se da la necesidad de contar con dos conjuntos completamente separados: De entrenamiento y de test.

1) Parte 1: Entrenamiento y clasificación

En esta primera parte se entrena el clasificador con el conjunto de entrenamiento y a continuación se evalúa con el conjunto de test.

El objetivo del **entrenamiento** del clasificador de regresión logística es el de estimar una serie de parámetros $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]$ utilizados para estimar la probabilidad de que un determinado patrón $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]$ pertenezca a la clase positiva. En regresión logística, la **función de hipótesis** será la función sigmoidal, que acota cualquier entrada entre 0 y 1, como se indica en la Figura . La ecuación de dicha función es:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Durante la fase de entrenamiento, se van a modificar los parámetros $\boldsymbol{\theta}$ de manera que se minimice una **función de coste**, que indicará el error medio entre las salidas de la hipótesis para los patrones de entrenamiento y su clase real. En Regresión Logística esta función de coste viene dada por:

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \ln(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \ln\left(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)\right]$$

Esta optimización se realiza mediante el algoritmo de descenso del gradiente, que consiste en un procedimiento iterativo en el que se actualizan cada uno de los componentes del vector $\boldsymbol{\theta}$:

$$\theta_j^{(k)} = \theta_j^{(k-1)} - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

Ten en cuenta que el elemento x_0 siempre vale 1. En la práctica se obtendrá añadiendo 1 al inicio del vector del patrón.

Para simplificar la comprensión, en esta práctica la **condición de parada** del algoritmo del descenso del gradiente es que se llegue al **número máximo de iteraciones** (indicadas en el parámetro `max_iter`). Además, en cada iteración del algoritmo del descenso del gradiente en la función de entrenamiento se realizan **tres pasos**:

1. Se calcula el valor de la **salida de la función de hipótesis (h)** de cada elemento del conjunto de entrenamiento.
2. Actualización de los componentes de $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]$.
3. Cálculo del **valor de la función de coste** (J) con los nuevos parámetros $\boldsymbol{\theta}$.

Si el entrenamiento se realiza correctamente, el coste tiene que ser más pequeño en cada iteración.

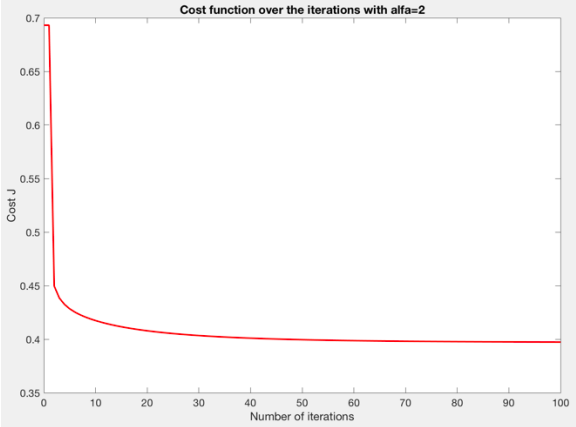

Figura 1. Función de coste en el entrenamiento a lo largo de 100 iteraciones

Una vez finalizado el entrenamiento, estos parámetros se utilizarán para estimar la probabilidad de pertenencia de un patrón cualquiera a la clase positiva, es decir, para clasificarlo.

a. Entrenamiento

El entrenamiento se implementa en la función `fEntrena_LogisticReg`. Esta función recibe los siguientes parámetros:

- **`X_train`**: Matriz de dimensión `(n x m)` con los datos de entrenamiento, donde `m` es el número de patrones de entrenamiento y `n` es el número de características (la longitud del vector de características que definen el patrón.

- **`Y_train`**: Vector que contiene las clases de los patrones de entrenamiento. Su longitud será `n`.

- **`alpha`**: Tasa de aprendizaje del algoritmo de descenso del gradiente.

Y como parámetro de salida devuelve **`theta`**, un vector de longitud `n` (es decir, el número de características de cada patrón con los parámetros `theta` de la hipótesis estimada tras el entrenamiento.

Ábrela, estudia el código y completa las partes del código indicadas con:

```
% ======================= YOUR CODE HERE ======================


% =============================================================
```

Tendrás que usar las funciones **`fun_sigmoidal`** para calcular el resultado de la hipótesis, que es el valor de la función sigmoidal dada como entrada la combinación lineal de los `theta` y los `x` y **`fCalculaCosteRegLog`** para calcular el coste de una salida particular de la función de hipótesis. También tendrás que completar estas dos funciones.

b. Clasificación

Una vez entrenado el clasificador (es decir, obtenido el vector $\boldsymbol{\theta}$ para los que el coste es mínimo), se realiza la clasificación con dicho vector $\boldsymbol{\theta}$. Esta clasificación se realiza en la función `fClasifica_LogisticReg`, la cual devuelve la probabilidad de cada patrón del conjunto de test de pertenecer a la clase positiva. Sus parámetros de entrada son:

- **`X_test`**: Matriz de dimensión (n x m_t) con los datos de entrenamiento, donde m_t es el número de patrones de entrenamiento y n es el número de características (la longitud del vector de características que definen el patrón.

- **`theta`**: Vector de longitud n (es decir, el número de características de cada patrón con los parámetros theta de la hipótesis estimada tras el entrenamiento.

Ábrela y completa las partes del código indicadas con:

```
% ====================== YOUR CODE HERE ======================


% ============================================================
```

2) Parte 2: Estimación del rendimiento del clasificador

El clasificador de regresión logística devuelve la probabilidad de un patrón determinado de pertenecer a la clase positiva. Sin embargo, para asignar un patrón a una clase es necesario determinar un valor denominado **umbral de decisión**, tal que cuando la salida del clasificador es mayor que dicho umbral, la clase asignada será la positiva. Normalmente este umbral vale 0.5.

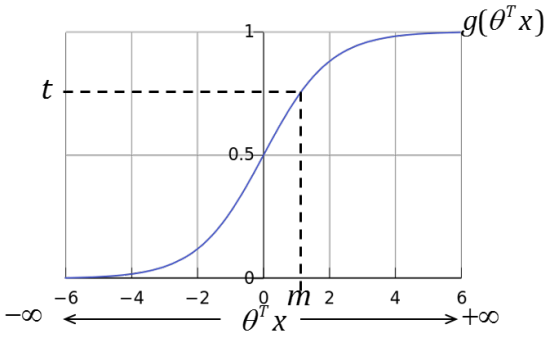

Figura 2. Función sigmoidal

Esto permite obtener métricas de rendimiento de la clasificación como las que vimos en la práctica anterior, basadas en la matriz de confusión del clasificador.

La **tasa de acierto** (*accuracy*) es el **porcentaje de elementos bien clasificados**. Es decir, el número de elementos que el clasificador clasificó bien entre el número total de elementos clasificados. Sin embargo, esta no siempre es una buena medida de rendimiento del clasificador, sobre todo cuando el conjunto de datos de test está muy desequilibrado (es decir,

hay muchos más elementos de una clase que de otra). Supongamos que queremos construir un clasificador para diagnosticar una enfermedad de la que se estima que solo la contrae el 1% de la población. Si el clasificador predijera que todo el mundo está enfermo, tendría un 99% de tasa de acierto. Sin embargo, **esto no quiere decir que el clasificador sea bueno**.

Existen otras medidas más "justas" para medir el rendimiento del clasificador, como la *precision*, *recall* o el *F1 Score*. Dada la matriz de confusión del clasificador:

		Clase predicha	
		1	0
Clase real	1	TP	FN
	0	FP	TN

Recordemos que TP, TN, FP, FN corresponden respectivamente, al número de **verdaderos positivos**, **verdaderos negativos**, **falsos positivos** y **falsos negativos**. Así, por ejemplo, si FN=10, significa que el clasificador ha clasificado como negativos (clase 0) 10 elementos que realmente eran positivos (pertenecían a la clase 1).

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- ***Precision*:** Es la fracción de verdaderos positivos entre el **total de elementos que el clasificador ha predicho como positivos**.

$$precision = \frac{TP}{pred\_positive} = \frac{TP}{TP + FP}$$

- ***Recall*:** Fracción de los verdaderos positivos entre los **elementos que son realmente positivos**.

$$recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- ***F-Score*:** Es la media armónica entre la *precision* y el *recall*. De este modo, da una idea del equilibrio entre ambas medidas.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$