Análisis de modelos de machine learning. Regresión lineal y regresión logística.

Pablo Vargas Ibarra

Trabajo de fin de grado

Grado en Ingeniería Matemática

Tutor: Javier Yañez

Dpto. Estadística e Investigación Operativa

Abstract

Linear regression and logistic regression are two classical models of supervised learning. The aim of this project is to analyze both with mathematical rigorous. Firstly, the strong statistical assumptions and the optimization problem derived from them. Secondly, regularization techniques and the obtaining of the optimal parameters using the algorithm gradient descent. Finally, some model evaluation metrics are explained, also one methodology is proposed in order to find models that generalize correctly with new data. A Github repository has been developed with Python implementations where the theory is applied to real data [8].

Resumen

La regresión lineal y la regresión logística son dos modelos clásicos de aprendizaje supervisado. El objetivo de este proyecto es analizar ambos matemáticamente de manera rigurosa. En primer lugar, las fuertes hipótesis estadísticas y el problema de optimización asociado. En segundo lugar, técnicas de regularización y la obtención de parámetros óptimos utilizando el algoritmo del descenso del gradiente. Finalmente, se exponen distintas métricas de evaluación de modelos y una metodología para encontrar modelos que generalicen correctamente con nuevos datos. Se ha desarrollado un repositorio en GitHub con implementaciones en Python dónde se aplica la teoría sobre datos reales [8].

Índice general

Capítulo 1

Introducción

El Machine Learning, también llamado aprendizaje automático, es un campo de la **Inteligencia Artificial**. En la década pasada, unos ejemplos de aplicaciones que el Machine Learning permiten la búsqueda web efectiva, el reconocimiento de voz y la traducción de texto automatizada.

1.1. Definiciones

Actualmente dos definiciones del Machine Learning han tomado mayor peso. La primera definición es más antigua y menos técnica:

"Campo de estudio que ofrece a los ordenadores la habilidad de aprender sin estar explícitamente programados". Arthur Samuel.

La segunda entra más en detalle utilizando más conceptos:

"Un ordenador aprende de la experiencia E, respecto a una clase de tarea T y una medida de mejora P si ésta mejora en la tarea T, medida por P, mejora gracias a la experiencia E". Tom Mitchell

Un ejemplo de está definicion podría ser jugar a un juego J, siendo :
E = Experiencia jugando a J
T = Jugar a J
P = Probabilidad de que el ordenador consiga la victoria

1.2. Datos como experiencia

La experiencia a partir de la cuál nuestro algoritmo va a "aprender" se traduce en los datos. Los datos son el alimento que sustenta cualquier algoritmo de Machine Learning, normalmente, conforme más datos se disponen mayor será su precisión, tal y como la definamos. Gracias al fenómeno llamado **Big Data**, el aprendizaje automático está creciendo considerablemente tanto en el mundo empresarial como en la investigación.

La notación que se va a seguir en este proyecto es la clásica, desde el punto de vista computacional tendremos:

$$X^{m\times n} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad y^{m\times 1} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

input data **output data**

Durante este proyecto se complementará el análisis computacional junto con un análisis estadístico. Esto suele ayudar a la interpretación de resultados de cara a la toma de decisiones en el mundo real.
El conjunto de datos son una muestra de **m** individuos, de cada uno de ellos hemos medido **n** características/variables/atributos. El "output" es la característica que queremos predecir.

$$\boldsymbol{x_j^{(i)}} = \textbf{ valor del atributo j en el individuo i}$$

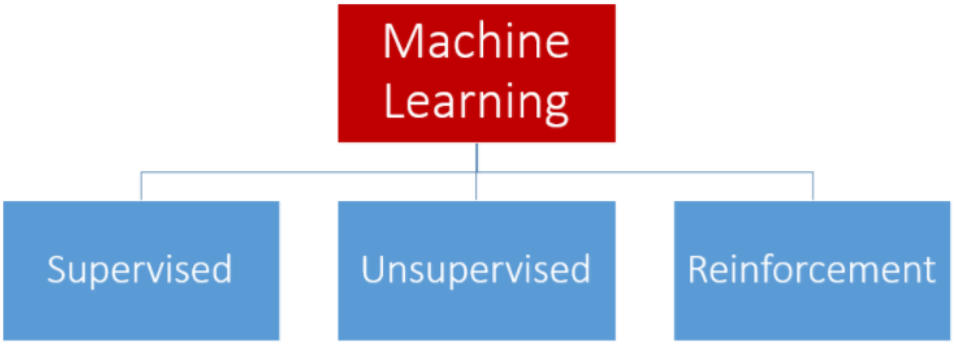
$$\boldsymbol{y^{(i)}} = \textbf{característica que queramos predecir en el individuo i}$$

No todos los problemas tienen una variable que queramos predecir. También cabe destacar que no siempre el hecho de disponer de más datos mejora la capacidad predictiva. Todo esto será argumentado posteriormente en el capitulo donde se trate en profundidad.

1.3. Tipos de aprendizaje

Existen tres grandes grupos en los cuales podemos dividir la forma de abordar los problemas de Machine Learning.

Figura 1.1: Tipos de Aprendizaje



1.3.1. Supervisado

En este tipo de aprendizaje, conocemos la relación entre los datos de entrada (input) y los de salida (output). De hecho, conocemos exactamente cuál debe de ser la salida correcta en nuestro algoritmo para estos datos.
El aprendizaje supervisado, a su vez, se puede dividir en regresión y clasificación. El problema de regresión intenta predecir un valor real continuo, mientras que los problemas de clasificación un valor discreto, es decir, en que clase o grupo se debe clasificar cada individuo.

Ejemplos de regresión:

(a) Dada una imagen de hombre o mujer, predecir la edad en base a la imagen.

(b) Dado el tamaño y zona de un inmueble, predecir su precio.

(c) Dado el historial médico de un paciente, predecir sus años restantes de vida.

Ejemplos de clasificación:

(a) Dada una imagen de hombre o mujer, predecir si se encuentra en la universidad, trabajando o jubilado.

(b) Dado el tamaño y zona de un inmueble, predecir si está habitada o desocupada.

(c) Dado el historial médico de un paciente, predecir si padece cierta enfermedad.

1.3.2. No Supervisado

El aprendizaje no supervisado, sin embargo, nos permite afrontar problemas con poca o ninguna idea sobre el resultado. Buscamos patrones en los datos sin la necesidad de conocer los efectos de las variables. Una posible manera de afrontar el problema es localizando grupos o clusters definidos por relaciones entre las variables.
Una de las principales diferencias con el aprendizaje supervisado es que no existe un "output", lo cual implica, que no podemos obtener un feedback basado en las predicciones, es decir, no existe una métrica clara que ayude a conocer cómo de bien funciona nuestro algoritmo.

Ejemplo :

(a) Un ejemplo de **clustering** podría ser segmentar a los clientes de un producto en función de ciertas características con el objetivo de realizar una diferente campaña publicitaria.

1.3.3. Reforzado

El aprendizaje reforzado tiene como objetivo aprender qué hacer maximizando una medida de recompensa. Al sistema no se le indica que acciones tomar, en cambio, debe descubrir cuál es la que conlleva mayor beneficio. En los casos más complejos e interesantes, las acciones pueden no implicar una recompensa máxima de manera instantánea, sino estar prevista para situaciones posteriores (más de un paso temporal).

1.4. Notación Aprendizaje Supervisado

El aprendizaje supervisado actualmente es una de las ramas del Machine Learning más investigada e utilizada. Trata de inferir una función sobre los datos de entrenamiento, estos datos están etiquetados y sabemos con anterioridad qué solución debemos obtener. La función inferida tiene como objetivo principal etiquetar nuevos datos con la mayor precisión posible. Formalmente, en este tipo de algoritmos tenemos valores de entrada (input) y salida (output). Clásicamente por convenio asumimos:

$$\boldsymbol{x_0^{(i)} = 1 \ \forall i \in (1, ..., m)}$$

$$X^{m\times(n+1)} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad y^{m\times 1} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

input data **output data**

$$\boldsymbol{x^{(i)} \text{ está etiquetado por } y^{(i)} \ \forall i \in (1, ..., m)}$$

La variable $x_0^{(i)}$ es constantemente uno, multiplicará al término constante, independiente del valor de las **n** variables. Esta nomenclatura nos permitirá simplificar nuestros cálculos, en especial, a la hora de implementarlos de manera vectorizada. El hecho de implementar algoritmos de manera vectorizada permite reducir complejidad computacional.

1.5. Hipótesis principal

La hipótesis principal que asumimos en el aprendizaje supervisado es que el valor que tomen las variables independientes, X, es capaz de explicar la variable dependiente y. Esta afirmación **no implica** que las variables independientes ***causen*** los distintos valores de y, sólo que existe algún tipo de correlación que será utilizada para inferir su valor.

Capítulo 2

Regresión Lineal Múltiple

La regresión lineal es un modelo de aprendizaje supervisado. Su objetivo es estimar la relación lineal entre una variable dependiente real (y) y sus regresores (X). En el caso en el que sólo haya un regresor o variable explicativa, se trata de una regresión lineal simple.

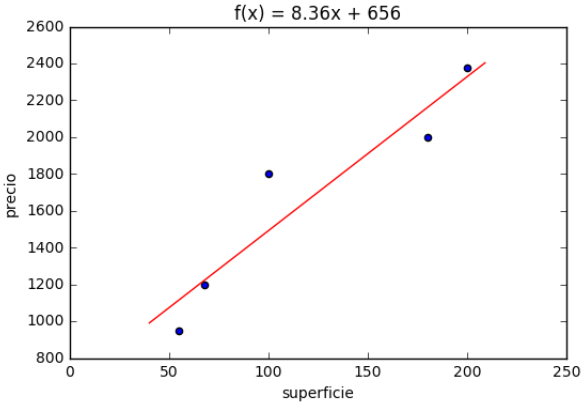
Un ejemplo de problema al que se le puede aplicar una regresión lineal simple sería, a partir de los metros cuadrados de unos inmuebles, inferir el precio al mes de alquiler.

Superficie (m^2)	Precio alquiler (€/mes)
200	2375
68	1200
55	950
100	1800
180	2000

Buscamos una función con la cual podamos predecir el precio de la vivienda dada su superficie. Una posible aproximación podría ser que:

"El precio por mes de cada vivienda viene dado por un poco más de ocho veces su superficie más un precio base de 656 euros", gráficamente:

Figura 2.1: Recta regresión



2.1. Modelo

Definimos nuestra función hipótesis como:

$$\boldsymbol{h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n} \ (\ \boldsymbol{\theta \in \mathbb{R}^{n \times 1}})$$

Para ganar intuición sobre esta función, en el ejemplo anterior podemos interpretar como θ_0 el coste base del apartamento, θ_1 el coste por m^2, etc.

Sea X la matriz de datos de entrenamiento, la expresión de manera vectorizada corresponde a

$$\boldsymbol{h_\theta(X) = X\theta \in \mathbb{R}^{m \times 1}}$$

El término θ_0 siempre estará multiplicado por $x_0^{(i)}$ que es constantemente la unidad.

2.2. Función de coste

El ajuste de los parámetros estará asociado a minimizar una función de coste. Esta función intentará medir lo "buenos" que son los parámetros de $h_\theta(x)$. El coste toma valores reales positivos y será un modelo perfecto si su valor es cero. La función de coste queda definida como:

$$\boldsymbol{J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})}$$

Cualquier persona familiarizada con el modelo de regresión lineal reconocerá que esta función de coste se corresponde a la clásica de mínimos cuadrados. Intuitivamente el mínimo global de nuestra función de coste ocurre cuando nuestra función mapea cada input con su output, siendo su precisión absoluta, o dicho de otra manera, coste cero.

2.3. Interpretación probabilística

En esta sección, vamos a analizar una interpretación probabilística [3]. Bajo ciertas hipótesis, justificará la función de coste anterior. Primero asumimos que nuestras variables siguen una relación via la siguiente ecuación

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

$\epsilon^{(i)}$ es el error provocado por efectos no controlados por el modelo como ruido en los datos o el azar. Además, asumiremos que $\epsilon^{(i)} \sim \mathcal{N}(0,\, \sigma^2)$. Entonces su densidad será

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(\epsilon^{(i)})^2}{2\sigma^2} \right)$$

Lo cual implica que:

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)$$

Siendo "$p(y^{(i)}|x^{(i)};\theta)$" la distribución de $y^{(i)}$ dado $x^{(i)}$ parametrizada por θ. Nótese que θ no es una variable aleatoria, por lo tanto, podemos escribir que $y^{(i)}|x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)},\, \sigma^2)$.

Dados X y θ, ¿cuál es la distribución de $y^{(i)}$?

La probabilidad de los datos viene dada por $p(\vec{y}|X;\theta)$. Cuando queremos expresar esto en función de θ, vamos a utilizar la función de verosimilitud

$$L(\theta) = L(\theta; X,\, \vec{y}) = p(\vec{y}|X;\theta)$$

Nótese que suponiendo hipótesis de independencia sobre los $\epsilon^{(i)}$'s también existirá entre los $y^{(i)}$'s dados los $x^{(i)}$'s, entonces

$$L(\theta) = \prod_{i=1}^{m} p(\vec{y}|X;\theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)$$

El principio de verosimilitud afirma que debemos tomar θ tal que maximice la probabilidad de los datos, es decir, debemos maximizar $L(\theta)$. De manera más simple podemos maximizar el logaritmo de la verosimilitud $l(\theta)$.

$$l(\theta) = \log L(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right) =$$
$$= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \left[\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 \right]$$

En consecuencia, maximizar $l(\theta)$ es similar a minimizar

$$\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 = m \cdot J(\theta)$$

En resumen: Bajo las hipótesis probabilísticas anteriores, la solución mínimos cuadrados encontrada es la que se corresponde a encontrar la máxima verosimilitud en los parámetros. Por lo tanto, la función de coste del apartado anterior es razonable.

2.4. Interpretación de coeficientes

La regresión lineal múltiple es un modelo relativamente sencillo dado que busca relaciones lineales en los datos. Una vez que hemos fijado los parámetros óptimos de nuestro modelo, podemos interpretarlos.
Dada nuestra función:

$$h_\theta(x_1, x_2, ..., x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... + \theta_n x_n$$

Tomamos $x^{k^+} = (x_1, x_2, x_3, ..., x_k + 1, ..., x_n)^T$, es decir, todos los valores de los atributos son similares salvo en x_k que aumenta una unidad. Entonces:

$$h_\theta(x^{k^+}) - h_\theta(x) = \theta_k(x_k + 1) - \theta_k x_k = \theta_k$$

Si fijamos el valor de todas las variables salvo x_k, por cada unidad que crezca o decrezca, el valor de y crecerá o decrecerá respectivamente en θ_k unidades.

En un contexto real puede ser imposible aumentar el valor de una variable manteniendo las demás fijas (modelos cuadráticos, dependencias).

2.5. Optimización con descenso del gradiente

Buscamos encontrar el valor de θ tal que minimice $J(\theta)$. Para encontrarlo, asumimos que iniciamos con un valor inicial para θ y de manera continua, hacemos $J(\theta)$ pequeña hasta que finalmente converja a un valor de θ óptimo. En particular vamos a considerar el algoritmo del descenso del gradiente, el cual comienza con un valor inicial para θ, actualizando:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta). \ (j = 0, 1, ..., n) \quad \text{simultaneamente}$$

Siendo α la tasa de aprendizaje. Se puede probar que para α suficientemente pequeño, siempre existe convergencia sacrificando velocidad. Un radio demasiado grande puede provocar divergencia. Tendríamos:

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

Reescribiéndolo de manera vectorizada para mejorar su implementación

$$\nabla J(\theta) = \frac{1}{m} \vec{x}_j^{\,T} (X\theta - \vec{y}) = \frac{1}{m} X^T (X\theta - \vec{y})$$

Quedando el algoritmo del descenso del gradiente de manera vectorial

$$\boldsymbol{\theta := \theta - \frac{\alpha}{m} X^T (X\theta - \vec{y})} \quad \textbf{hasta convergencia}$$

En este proyecto no se va a entrar en detalle en este algoritmo debido a que existen muy buenas implementaciones de él en lenguajes de programación como R, Python, etc. Se realizará una imprelentación para mostrar la idea con la cuál podemos llegar a obtener los parámetros que minimizan la función de coste y se compararán resultados con la libreria scikit-learn.

2.6. Convexidad

Vamos a demostrar que la función de coste de la regresión lineal múltiple es convexa viendo que su matriz hessiana es semi-definida positiva

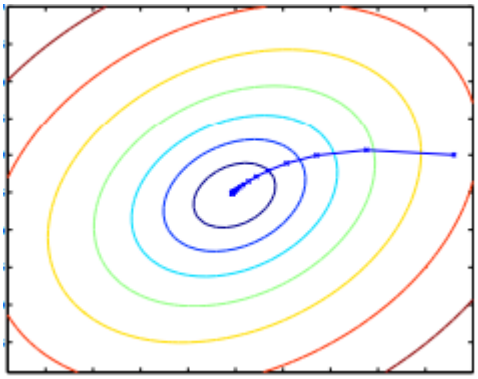
$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y}) = \frac{1}{2m} (\| \vec{y} \|^2 - 2y^T X\theta + \theta^T X^T X\theta)$$

$$\implies \frac{\partial^2 \phi}{\partial \theta^2} = \frac{1}{2m} X^T X. \quad \text{Semi-definida positiva si:} \quad \forall a \ : \ a^T X^T X a \geq 0$$

Tomamos $v = Xa \implies a^T X^T X a = v^T v = \sum_{i=1}^{m} v_i^2 \geq 0$

Como la segunda derivada es semi-definida positiva. Podemos afirmar que la función de coste es convexa. La convexidad de nuestra función implica que tiene todo mínimo local es global, lo cual es una propiedad muy interesante. Nuestro algoritmo del descenso del gradiente, si converge, lo hará al mínimo global. Para hacernos una idea gráficamente, supongamos que queremos minimizar un $J(\theta_0, \theta_1)$, entonces nuestro algoritmo se acercará al mínimo como se muestra en el siguiente gráfico:

Figura 2.2: Convexidad con el descenso del gradiente



En resumen: la función de coste definida para la regresión lineal múltiple es convexa, por lo tanto, tiene un mínimo global y disponemos de herramientas para alcanzarlo. Siendo éste el valor óptimo de los parámetros, que, aplicado a nuestra función, será la que mejor predecirá la variable objetivo.

2.7. Implementación en Python

En el siguiente link, se muestra una implementación vectorizada propia del descenso del gradiente para encontrar los parámetros óptimos de una regresión lineal. Además, posteriormente se compara con un resultado obtenido al utilizar la librería scikit-learn.

Link: `https://github.com/PabloVargasIbarra/TFG_ML_Exercises/blob/master/RegLineal.ipynb`

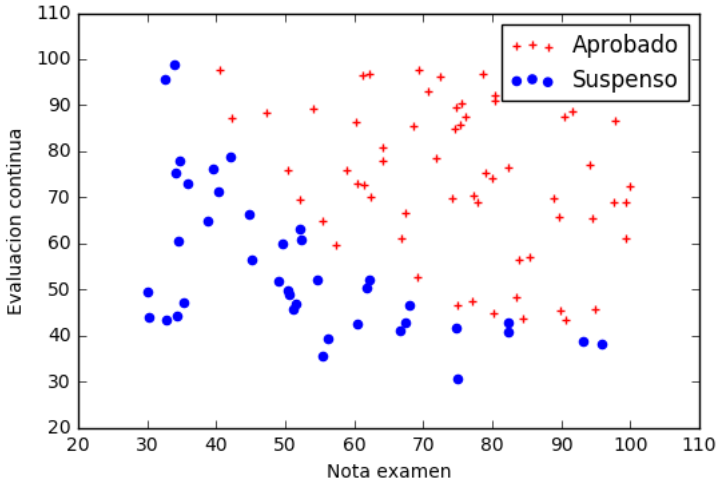
Capítulo 3

Regresión Logística

La regresión logística es un modelo de clasificación binaria. El objetivo es estimar la probabilidad de cada clase a partir de uno más regresores. Fue desarrollada por el estadístico David Cox en 1958.

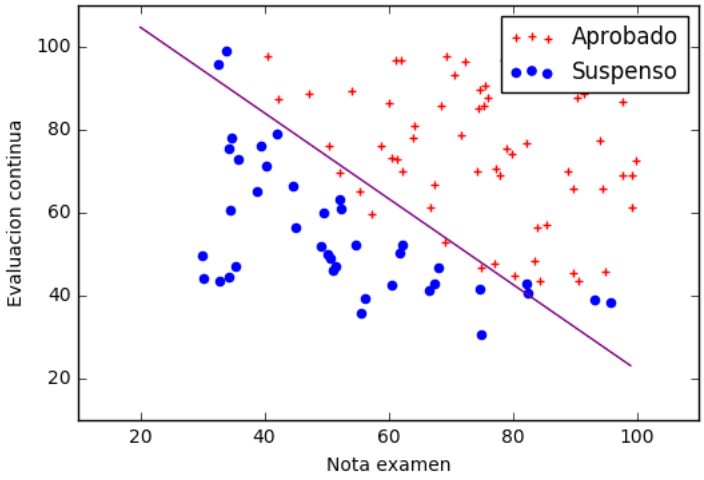
Dado un conjunto de calificaciones de unos alumnos (nota del examen y evaluación continua). Si buscamos inferir la nota final se consideraría una regresión. Sin embargo, si lo que queremos predecir es simplemente si han aprobado o suspendido los alumnos, el problema se transformaría a uno de clasificación. Aquí el gráfico con los datos.

Figura 3.1: Calificaciones finales



Si queremos encontrar una función que separe los suspensos de los aprobados, lo más sencillo sería buscar un clasificador lineal. Este clasificador lineal dependerá exclusivamente de las variables iniciales (nota del examen y de evaluación continua).

Figura 3.2: Clasificador lineal



Podemos observar que no todos los alumnos han sido clasificados correctamente y que el clasificador lineal no es suficiente dado que existes relaciones no lineales (nota mínima para aprobar). Nuestro separador lineal podría ser interpretado como la necesidad de una media ponderada entre las dos calificaciones. Si denotando como:

$$x_1 = \text{Nota del examen}$$

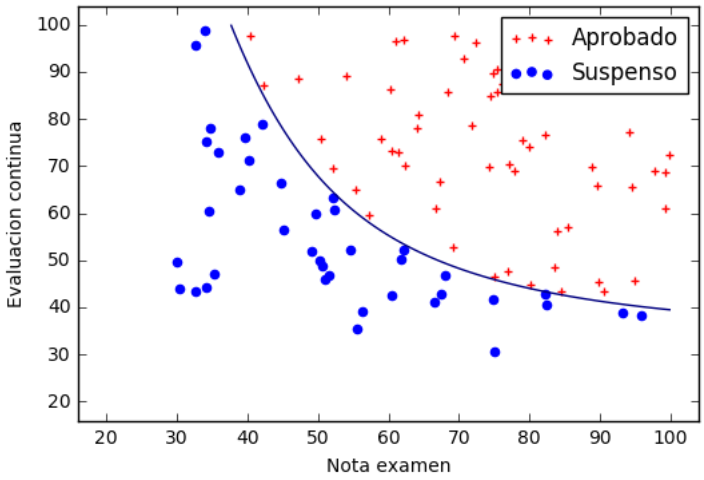
$$x_2 = \text{Evaluación continua}$$

Para estos datos un alumno sería clasificado como aprobado si cumpliera la siguiente inecuación (aproximadamente):

$$0{,}2(x_1 + 0{,}2x_2) > 25$$

Necesitamos detectar las relaciones no lineales, para ello creamos nuevos atributos a partir de los iniciales y buscamos un clasificador no lineal.

Figura 3.3: Clasificador no lineal



Podemos observar que este clasificador es una curva y no una recta como el anterior. Hemos necesitado calcular variables cuadráticas a partir de las iniciales (x_1^2, x_1x_2, x_2^2). Estas variables nos permiten detectar las relaciones no lineales y definir un separador totalmente preciso para este problema.

Una posible interpretación para la no linealidad podría ser la necesidad de una nota mínima para aprobar la asignatura.

Esta introducción a la regresión logística se puede encontrar en detalle en:

Link: `https://github.com/PabloVargasIbarra/TFG_ML_Exercises/blob/master/IntroRegLog.ipynb`

En esta sección se tratará la teoría matemática detrás del modelo de regresión logística

3.1. Modelo

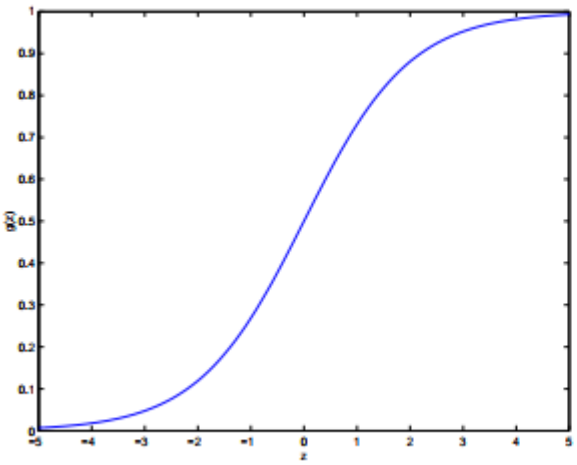
Al igual que para el problema de regresión, vamos a proponer una función que intentará mapear los valores de input x en los de output y. Es decir, dados los datos de entrenamiento, predecir a qué clase pertenece cada individuo. En vez de ser nuestro output un vector de valores continuos como en la regresión, va a ser binario. Es decir, $y \in \{0, 1\}$.

Nuestra función debe satisfacer que $0 \leq h_\theta(x) \leq 1$, para ello se toma:

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

donde $g(z) = \dfrac{1}{1 + e^{-z}}$ es conocida como la función logística. La función logística cumple varias propiedades interesantes por las cuales es una buena elección. Aquí una gráfica:

Figura 3.4: Función logística



$$\begin{aligned} z = 0, e^0 = 1 &\Rightarrow g(z) = \frac{1}{2} \\ z \to \infty, e^{-\infty} \to 0 &\Rightarrow g(z) = 1 \\ z \to -\infty, e^{\infty} \to \infty &\Rightarrow g(z) = 0 \end{aligned}$$

Tomaremos de manera vectorizada

$$\boldsymbol{g(X\theta) = \left(g(x^{(1)}), g(x^{(2)}), ..., g(x^{(m)})\right)^T \in \mathbb{R}^{m \times 1}}$$

3.2. Frontera de decisión

Nuestra función devolverá un valor entre cero y uno, tomando un p límite de decisión clasificaremos a un individuo como:

$$\hat{y} = \begin{cases} 1, & \text{si } h_\theta(x) \geq p \\ 0, & \text{en otro caso ó } h_\theta < p \end{cases} \tag{3.1}$$

La elección del valor de p depende del problema en particular, se tratará en profundidad en el capítulo de validación de modelos. Una opción razonable sería p =0.5, en ese caso:

$$\hat{y} = 1 \Leftrightarrow h_\theta(x) = g(\theta^T x) \geq 0.5 \Leftrightarrow \frac{1}{1 + e^{-\theta^T x}} \geq 0.5 \Leftrightarrow 1 \geq e^{-\theta^T x} \Leftrightarrow \theta^T x \geq 0$$

La frontera de decisión es la linea creada por nuestra función $h_\theta(x)$ que separa las dos clases.

3.3. Función de coste

No podemos utilizar la misma función que la utilizada para la regresión lineal. Esto es debido a que la función logística causaría una "salida ondulada", provocando una gran cantidad de mínimos locales. En otras palabras, no sería una función convexa. La función de coste de la regresión logística será

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\Big(h_\theta(x^{(i)}, y^{(i)}\Big) \quad \text{con}$$

$$\text{Cost}\Big(h_\theta(x), y\Big) = \begin{cases} -\log(h_\theta(x)) & \text{si } \ y = 1 \\ -\log(1 - h_\theta(x)) & \text{si } \ y = 0 \end{cases}$$

Esta elección parece razonable en el sentido de que; Si nuestra función $h_\theta(x)$ mapea nuestro output de manera correcta, la función de coste toma valor cero. Conforme se va alejando de la clasificación correcta, el coste crece hasta infinito. Es decir,

$$\text{Cost}\Big(h_\theta(x), y\Big) = 0 \text{ si } h_\theta(x) = y$$
$$\text{Cost}\Big(h_\theta(x), y\Big) \to \infty \text{ si } y = 0 \text{ con } h_\theta(x) \to 1$$
$$\text{Cost}\Big(h_\theta(x), y\Big) \to \infty \text{ si } y = 1 \text{ con } h_\theta(x) \to 0$$

Podemos comprimir nuestra función de coste en

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

De manera vectorizada sería

$$\boldsymbol{J(\theta) = \frac{1}{m}\Big(- y^T \log(h) - (1-y)^T \log(1-h)\Big)} \text{ siendo } \boldsymbol{h = g(X\theta)}$$

3.4. Interpretación probabilística

Vamos a justificar la elección de la función de coste al igual que hicimos en la anterior sección con la regresión lineal [3]. Se asumen ciertas hipótesis probabilísticas en nuestro modelo clasificatorio, y entonces, se ajustan los parámetros que maximicen la función de verosimilitud.

Primero asumimos que

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Es decir $y \sim \text{Bernoulli}(h_\theta(x))$. Por lo tanto su función de masa es:

$$p(y \mid x; \theta) = \Big(h_\theta(x)\Big)^y \Big(1 - h_\theta(x)\Big)^{1-y}$$

Suponiendo que los m ejemplos de entrenamiento están generados de manera independiente, podemos entonces escribir como función de verosimilitud

$$L(\theta) = p(\vec{y} \mid X; \Theta) = \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} \Big(h_\theta(x^{(i)})\Big)^{y^{(i)}} \Big(1-h_\theta(x^{(i)})\Big)^{1-y^{(i)}}$$

El principio de verosimilitud afirma que debemos tomar θ tal que maximice la probabilidad de los datos, es decir, maximizar $L(\theta)$. De manera más simple podemos maximizar el logaritmo de la verosimilitud $l(\theta)$.

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m}\Big[y^{(i)}\log\Big(h_\theta(x^{(i)})\Big) + (1-y^{(i)})\log\Big(1-h_\theta(x^{(i)})\Big)\Big] = -m \cdot J(\theta)$$

Por lo tanto maximizar la verosimilitud en los parámetros es similar a minimizar la función de coste $J(\theta)$.

3.5. Odds Ratio

Utillizando la anterior interpretación probabilística de la regresión logística vamos a definir los **Odds** como

$$Odds(x) = \frac{P(y=1 \mid x;\theta)}{P(y=0 \mid x;\theta)} = \frac{h_\theta(x)}{1-h_\theta(x)} = \frac{1/(1+e^{-\theta^T x})}{e^{-\theta^T x}/(1+e^{-\theta^T x})} = e^{\theta^T x}$$

Tomamos $x^{k^+} = (x_1, x_2, x_3, ..., x_k{+}1, ..., x_n)^T$, es decir, todos los valores de las características son similares salvo en la k que aumenta una unidad. Entonces

$$\frac{Odds(x^{k^+})}{Odds(x)} = \frac{e^{\theta^T x^{k^+}}}{e^{\theta^T x}} = \frac{e^{\theta_0+\theta_1 x_1+\theta_2 x_2+...+\theta_k(x_k+1)+...+\theta_n x_n}}{e^{\theta_0+\theta_1 x_1+\theta_2 x_2+...+\theta_n x_n}} = e^{\theta_k}\frac{e^{\theta^T x}}{e^{\theta^T x}} = e^{\theta_k}$$

Llamamos OR, odds ratio de la variable x_k como $\text{OR}(x_k) = e^{\theta_k}$.

Fijadas todas las variables que describen a un individuo salvo x_k, por cada unidad que la incrementamos, los odds de que se clasifique como $\hat{y} = 1$ se multiplican por e^{θ_k}. Igual que ocurría con la regresión lineal, en un contexto real, puede ser imposible aumentar el valor de una variable manteniendo las demás fijas (modelos cuadráticos, dependencias).

3.6. Convexidad

Para demostrar la convexidad de la función de coste vamos a utilizar los mismos conceptos que utilizamos en la regresión lineal. Sea la función de coste:

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

Demostrando que $\phi_1(\theta) = -\log\left(h_\theta(x^{(i)})\right)$ y $\phi_2(\theta) = -\log\left(1-h_\theta(x^{(i)})\right)$ son funciones convexas es suficiente. Esto es debido a que $y^{(i)}$ es positiva y la suma de funciones convexas es una función convexa.

Vamos a calcular la segunda derivada de $\phi_1(\theta)$

$$\phi_1(\theta) = -\log\Big(h_\theta(x^{(i)})\Big) = -\log\left(\frac{1}{1+e^{-\theta^T x}}\right) = \log\left(1+e^{-\theta^T x}\right)$$

$$\Rightarrow \frac{\partial \phi_1}{\partial \theta} = \left(\frac{-e^{-\theta^T x}}{1+e^{-\theta^T x}}\right)x = \left(\frac{1}{1+e^{-\theta^T x}} - 1\right)x = \Big(h_\theta(x) - 1\Big)x$$

Calculo la derivada de la función logística:

$$g(z)' = \left(\frac{1}{1+e^{-z}}\right)' = \frac{-1' - (e^{-z})'}{(1+e^{-z})^2} = \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) =$$

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

$$\implies h_\theta(x)' = g(\theta^T x)' = g(\theta^T x)\Big(1-g(\theta^T x)\Big)x = h_\theta(x)\Big(1-h_\theta(x)\Big)x$$

$\dfrac{\partial^2 \phi_1}{\partial \theta^2} = h_\theta(x)\Big(1-h_\theta(x)\Big)xx^T$, que es semi-definida positiva porque

$$\forall a \ : \ a^T \left[h_\theta(x)\Big(1-h_\theta(x)\Big)xx^T\right] a = h_\theta(x)\Big(1-h_\theta(x)\Big)(x^T a)^2 \geq 0$$

Hemos probado que $\phi_1(\theta)$ es convexa. Análogamente vamos con ϕ_2,

$$\phi_2(\theta) = -\log\Big(1-h_\theta(x)\Big) = -\log\left(1-\frac{1}{1+e^{-\theta^T x}}\right) = -\log\left(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}\right) =$$

$$= \theta^T x + \log\left(1+e^{-\theta^T x}\right) \implies \frac{\partial \phi_2}{\partial \theta} = x + \frac{\partial}{\partial \theta}\left[\log\left(1+e^{-\theta^T x}\right)\right]$$

Siendo la segunda derivada

$$\frac{\partial^2 \phi_2}{\partial \theta^2} = \frac{\partial}{\partial \theta}\left(x + \frac{\partial}{\partial \theta}\left[\log\left(1+e^{-\theta^T x}\right)\right]\right) = \frac{\partial^2}{\partial \theta^2}\left[\log\left(1+e^{-\theta^T x}\right)\right] = \frac{\partial^2 \phi_1}{\partial \theta^2}$$

Por el mismo argumento ϕ_2 es convexa.

En resumen, la regresión logística tiene una función de coste convexa. Al igual que con la regresión lineal podemos utilizar algoritmos como el descenso del gradiente sabiendo que en el caso de converger, siempre será al mínimo global.

3.7. Regularización

El concepto de regularización es muy importante en los modelos de Machine Learning [2]. El objetivo de los modelos es poder generalizar con datos nuevos, la función de la regularización está diseñada para ayudar a ello.

Un modelo sufre underfitting cuando nuestra función $h_\theta(x)$ mapea de manera incorrecta nuestro input en el output. Esto será debido a que nuestra función es demasiado simple o dispone de muy pocas variables independientes que expliquen la variable dependiente.

En el otro extremo, nuestro modelo sufrirá overfitting cuando la función $h_\theta(x)$ tiene una alta capacidad predictiva para los datos de entrenamiento, pero no es capaz de generalizar correctamente con nuevos datos [6]. Una posible interpretación geométrica sería que contiene muchas curvas y ángulos innecesarios no relacionados con los datos. La regularización es una técnica que previene el problema de overfitting. Utilizaremos la regularización de Tikhonov(**L_2-norm**).

$$R(\theta) = \frac{1}{2C}\sum_{j=1}^{n}\theta_j^2 \ ; \ C > 0$$

Siendo C el parámetro positivo que nos permitiría ajustar cuanto penalizamos la complejidad paramétrica de nuestra función hipótesis. El parámetro θ_0 no se regulariza por convenio, entonces la función de coste quedaría:

$$J_R(\theta) = J(\theta) + R(\theta) = J(\theta) + \frac{1}{2C}\sum_{j=1}^{n}\theta_j^2 \ ; \ C > 0$$

También conocida como **Ridge** [5]. Sí $C \to \infty$, nuestra función de coste tiende a no estar regularizada. En contraste, si $C \to 0$, nuestra función de coste tendería a tener valor nulo en los parámetros, provocando underfitting.

La regularización penaliza las variables con mayor escala, por lo tanto, es necesario estandarizarlas. El hecho de estandarizar los atributos nos imposibilita la interpretación de parámetros citada anteriormente, sin embargo, nos permite comparar el peso de las variables entre sí.

En el ejemplo en Python que se realiza posteriormente se mostrará de manera

gráfica. La regularización y su correcta aplicación se tratará en profundidad en el capítulo de validación de modelos.

La función de regularización $R(\theta)$ es una función convexa, por lo tanto la función de coste regularizada en la regresión lineal múltiple y logística sigue siendo convexa. Además es diferenciable, permitiéndonos utilizar el descenso del gradiente para minimizarla

3.8. Interpretación bayesiana

Hasta ahora hemos justificado las funciones de coste sin regularizar mediante la verosimilitud, eligiendo los parámetros acordes a

$$\theta_{ML} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)} x^{(i)}; \theta)$$

Hasta ahora hemos visto θ como un parámetro no conocido existente. Esta visión de θ, siendo un valor constante desconocido parte de una visión estadística frecuentista. Bajo la visión frecuentista del mundo [1], θ no es aleatorio, es desconocido y nuestro trabajo es a través de procedimientos estadísticos (como el de máxima verosimilitud) intentar estimarlo.
Una manera alternativa de afrontar este tipo de problemas de estimación de parámetros es a través de la visión bayesiana del mundo, viendo θ como una variable aleatoria cuyo valor es desconocido. Bajo la estadística bayesiana, nosotros especificaríamos una distribución apriori $p(\theta)$ sobre θ que expresase nuestros "conocimientos previos"sobre los parámetros. Dado el conjunto de entrenamiento $S = \{\{x^{(i)}, y^{(i)}\}\}_{i=1}^{m}$, cuando queremos hacer predicciones sobre un nuevo valor x, podemos calcular la distribución a posteriori de los parámetros

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)} = \frac{(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta))p(\theta)}{\int_{\theta} \Big(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)p(\theta) \Big) d\theta}$$

Siendo "$p(y|x, \theta)$" en vez de "$p(y|x; \theta)$" dado que bajo una interpretación bayesiana θ es ahora variable aleatoria.
En esta ecuación, $p(y^{(i)}|x^{(i)}, \theta)$ puede derivar de cualquier modelo de Machine Learning. Por ejemplo, si tomáramos el modelo de regresión logística entonces

$p(y^{(i)}|x^{(i)},\theta) = h_\theta(x^{(i)})^{y^{(i)}}(1-h_\theta(x^{(i)}))^{(1-y^{(i)})}$ donde $h_\theta(x^{(i)}) = (1+\exp(-\theta^T x^{(i)}))^{-1}$.
Cuando queremos realizar una predicción en un nuevo valor x, calculamos la distribución a posteriori de y utilizando la distribución a posteori de θ:

$$p(y|x,S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$$

En consecuencia, el valor esperado de y dado x, vendría dado por

$$E[y|x,S] = \int_y yp(y|x,S)dy$$

Convirtiendose la integral en un sumatorio en caso de ser y discreta. Este procedimiento puede ser considerado una predicción "totalmente" bayesiana, donde nuestra predicción es calculada respecto a la posterior $p(\theta|S)$ sobre θ. Desafortunadamente, es computacionalmente costoso calcular la distribución a posteriori. Esto es debido a que hay que computar integrales, normalmente de alta dimensión.
En la practica se puede aproximar la distribución a posteriori de θ. La estimación maximo a posteriori (**MAP**) de θ viene dada por

$$\theta_{\text{MAP}} = \arg\max_\theta \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)$$

Esta fórmula coincide con la máxima verosimilitud estimada para θ hasta ahora, salvo en el término final, la distribución a priori $p(\theta)$. En el caso en que se tome la distribución a priori $p(\theta) \sim \mathcal{N}_n(0,\, CI)$ (I matriz identidad $n \times n$, $C > 0$) con función de densidad:

$$p(\theta;\sigma_\theta) = \frac{1}{(2\pi)^{n/2}|CI|^{1/2}} \exp\Big(-\frac{1}{2}\theta^T(CI)^{-1}\theta\Big) = \frac{1}{(2\pi)^{n/2}C^{1/2}} \exp(-\frac{1}{2C}\sum_{j=1}^{n}\theta_j^2)$$

La tomamos como función de verosimilitud que queremos maximizar $L(\theta)$, aplicando el logaritmo obtenemos

$$l(\theta) = \log(\frac{1}{(2\pi)^{n/2}C^{1/2}}) - \frac{1}{2C}\sum_{j=1}^{n}\theta_j$$

Por lo tanto maximizar la verosimilitud en la función apriori $p(\theta)$ es similar a minimizar la función de regularización $R(\theta)$.

El término θ_0 no se regulariza. En la regresión logística por ejemplo, si buscásemos regularizarlo, el efecto sería que nuestra frontera de decisión pasase por el origen de coordenadas lo cual no implica ninguna ventaja contra el overfitting.

3.9. Descenso del gradiente con regularización

Recordamos la forma general vectorizada del descenso del gradiente viene dada por la siguiente expresión:

$$\theta := \theta - \alpha \nabla J(\theta) \qquad \text{(hasta convergencia)}$$

Nuestra nueva función de coste regularizada para la regresión logística es

$$J_R(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\log\Big(h_\theta(x^{(i)})\Big) + (1-y^{(i)})\log\Big(1-h_\theta(x^{(i)})\Big)\Big] + \frac{1}{2C}\sum_{j=1}^{n}\theta_j^2$$

Sabiendo que $g(z)' = g(z)\Big(1-g(z)\Big)$, g función logística (ya demostrado).

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{-1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\log\Big(h_\theta(x^{(i)})\Big) + (1-y^{(i)})\log\Big(1-h_\theta(x^{(i)})\Big)\Big] =$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\left[y^{(i)}\frac{\partial}{\partial\theta_j}\log\Big(h_\theta(x^{(i)})\Big) + (1-y^{(i)})\frac{\partial}{\partial\theta_j}\log\Big(1-h_\theta(x^{(i)})\Big)\right] =$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\left[\frac{y^{(i)}\dfrac{\partial}{\partial\theta_j}h_\theta(x^{(i)})}{h_\theta(x^{(i)})} + \frac{(1-y^{(i)})\dfrac{\partial}{\partial\theta_j}\Big(1-h_\theta(x^{(i)})\Big)}{1-h_\theta(x^{(i)})}\right] = (*)$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\Big(1-h_\theta(x^{(i)})\Big)x_j^{(i)} - (1-y^{(i)})h_\theta(x^{(i)})x_j^{(i)}\Big] =$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\Big[y^{(i)}\Big(1-h_\theta(x^{(i)})\Big) - (1-y^{(i)})h_\theta(x^{(i)})\Big]x_j^{(i)} =$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\Big[y^{(i)} - y^{(i)}h_\theta(x^{(i)}) - h_\theta(x^{(i)}) + y^{(i)}h_\theta(x^{(i)})\Big]x_j^{(i)} =$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\Big[y^{(i)} - h_\theta(x^{(i)})\Big]x_j^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\Big[h_\theta(x^{(i)}) - y^{(i)}\Big]x_j^{(i)}$$

$$(*)\quad \frac{\partial}{\partial\theta_j}h_\theta(x^{(i)}) = h_\theta(x^{(i)})\Big(1-h_\theta(x^{(i)})\Big)\frac{\partial}{\partial\theta_j}\theta^T x^{(i)} = h_\theta(x^{(i)})\Big(1-h_\theta(x^{(i)})\Big)x_j^{(i)}$$

Observamos que la expresión para el gradiente de la función de coste en la regresión logística coincide con la regresión lineal. La diferencia es la función $h_\theta(x)$, en este caso es la función logística. De manera vectorizada

$$\nabla J(\theta) = \frac{1}{m} X^T \Big(g(X\theta) - \vec{y} \Big)$$

Derivamos $R(\theta)$, sabiendo que θ_0 no se regulariza por convenio, entonces

$$\frac{\partial}{\partial \theta_0} R(\theta) = 0 \quad \wedge \quad \frac{\partial}{\partial \theta_j} R(\theta) = \frac{1}{2C} \frac{\partial}{\partial \theta_j} \sum_{j=1}^{n} \theta_j^2 = \frac{\theta_j}{C} \implies \nabla R(\theta) = \frac{1}{C} \begin{bmatrix} 0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$\implies \nabla J_R(\theta) = \nabla J(\theta) + \nabla R(\theta) = \frac{1}{m} X^T \Big(g(X\theta) - \vec{y} \Big) + \frac{1}{C} \begin{bmatrix} 0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

La expresión para el algoritmo del descenso del gradiente, utilizando la función de coste regularizada de la regresión logística, sería

$$\boldsymbol{\theta := \theta - \alpha \left(\frac{1}{m} X^T (g(X\theta) - \vec{y}) + \frac{1}{C} \begin{bmatrix} 0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \right)} \quad \text{hasta convergencia}$$

3.10. Implementación en Python

En el siguiente link, se realiza una implementación vectorizada del descenso del gradiente para encontrar los parámetros óptimos de una regresión logística regularizada. Además, posteriormente se compara con un resultado obtenido al utilizar la librería scikit-learn. También se prueban distintos valores para el parámetro de regularización, obteniendo un ejemplo de modelo con alta varianza y otro con alto sesgo.

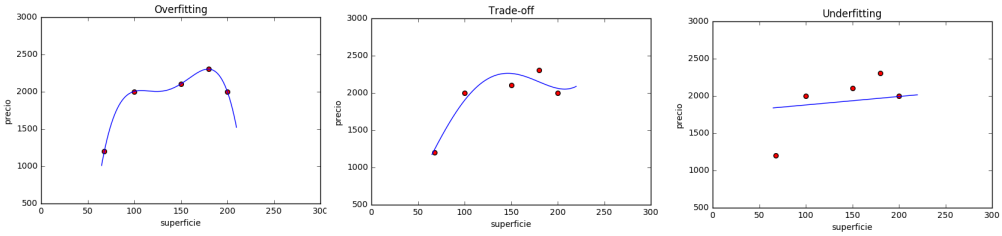
Link: `https://github.com/PabloVargasIbarra/TFG_ML_Exercises/blob/master/RegLogistica.ipynb`

Capítulo 4

Evaluación de modelos

Los modelos de aprendizaje supervisado tienen como objetivo predecir información sobre nuevos individuos, para ello se alimentan de datos del pasado, con los cuales se entrena el modelo. Tanto en la regresión lineal como en la regresión logística, para alcanzar este objetivo, buscamos minimizar la función de coste regularizada. Hemos visto dos causas principales para que un modelo no sea "bueno", overfitting y underfitting.

Figura 4.1: Trade-off



El primer modelo es capaz de predecir correctamente para los datos de entrenamiento el precio de las casas en función de su superficie, sin embargo, lleva patrones espurios implícitos. El tercero tampoco es capaz de predecir correctamente el precio en función de la superficie dado que la hipótesis no representa la relación entre ellos. Normalmente, existe un balanceo (trade-off) entre el sesgo y la varianza. Si nuestro modelo es demasiado "complejo" ($h_\theta(x)$ compleja), entonces puede sufrir alta varianza y bajo sesgo (overfitting). En cambio, si nuestra hipótesis es demasiado simple, entonces puede sufrir de alto sesgo y baja varianza (underfitting).

Evaluaremos los modelos por su error a la hora de **generalizar** con datos nuevos, tanto si sufren overfitting o underfitting, el error será alto. Sin embargo, se afrontarán de manera totalmente diferente.

4.1. Partición de datos

Una forma clásica de medir la capacidad de generalización de nuestro modelo es evaluarlo con datos distintos a los que hemos utilizado en el entrenamiento. Para ello, particionamos los datos en entrenamiento, validación y test.

1. Regresion lineal :

$$J_{\text{train}}(\theta) = \frac{1}{2m_{\text{train}}} \sum_{i=1}^{m_{\text{train}}} \left(h_\theta(x_{\text{train}}^{(i)}) - y_{\text{train}}^{(i)} \right)^2$$

$$J_{\text{val}}(\theta) = \frac{1}{2m_{\text{val}}} \sum_{i=1}^{m_{\text{val}}} \left(h_\theta(x_{\text{val}}^{(i)}) - y_{\text{val}}^{(i)} \right)^2$$

$$J_{\text{test}}(\theta) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left(h_\theta(x_{\text{test}}^{(i)}) - y_{\text{test}}^{(i)} \right)^2$$

2. Regresión logística :

$$J_{\text{train}}(\theta) = -\frac{1}{m_{\text{train}}} \sum_{i=1}^{m_{\text{train}}} \left[y_{\text{train}}^{(i)} \log\left(h_\theta(x_{\text{train}}^{(i)}) \right) + (1 - y_{\text{train}}^{(i)}) \log\left(1 - h_\theta(x_{\text{train}}^{(i)}) \right) \right]$$

De manera análoga a la regresión lineal definiremos $J_{\text{val}}(\theta)$ y $J_{\text{test}}(\theta)$ para la regresión logística. Siendo $[m_{\text{train}}, m_{\text{val}}, m_{\text{test}}]$ el número de ejemplos que utilizamos para entrenar, validar y testear nuestro modelo. Normalmente la proporción suele rondar los 60/20/20 respectívamente aunque no hay una regla óptima, depende de la cantidad de datos y su naturaleza.

La función de coste que minimizaremos para obtener los parámetros siempre estará regularizada. Es importante a la hora de analizar los costes que los

tres conjuntos de datos sigan la misma distribución. Desordenando primero aleatoriamente los datos, podemos ver nuestra partición matricialmente:

$$X^{m\times n} = \begin{bmatrix} x_{\text{train}} \\ x_{\text{val}} \\ x_{\text{test}} \end{bmatrix} \qquad y^{m\times 1} = \begin{bmatrix} y_{\text{train}} \\ y_{\text{val}} \\ y_{\text{test}} \end{bmatrix} \qquad \text{con } m = m_{\text{train}} + m_{\text{val}} + m_{\text{test}}$$

4.2. Elección del modelo

La razón por la que dividimos en tres conjuntos nuestros datos es que vamos a utilizarlos para tareas diferentes. Los datos de entrenamiento se utilizarán para minimizar la función de coste regularizada (entrenar el modelo), los datos de validación nos ayudarán a elegir el modelo y los datos de test para evaluar su capacidad de generalización. Una posible técnica sería:

1. Entrenamos distintos modelos (diferente algoritmo, distinto número de atributos, valor para C o grado en el polinomio) con los datos de entrenamiento.

2.Calculamos $J_{\text{val}}(\theta)$, error de validación, para cada uno de ellos. Quedándonos con el modelo con menor valor.

3. Su capacidad de generalización vendrá dada por $J_{\text{test}}(\theta)$.

En muchos libros y cursos se suele proponer dividir el conjunto de datos sólo en entrenamiento y test. En este caso la elección del modelo se hará en función del que menor error produzca en el conjunto de testeo, considerándolo también el error al generalizar. Esta técnica suele **sobreestimar** la capacidad de generalización debido a que hemos elegido el modelo teniendo en cuenta ese parámetro (valor de $J_{\text{test}}(\theta)$).

4.3. Relación entre parámetros y sesgo-varianza

En la técnica que se propone en la sección anterior, se sugiere elegir el mejor modelo teniendo en cuenta el valor de $J_{\text{val}}(\theta)$. Sabemos que las dos causas principales para que un modelo no generalize correctamente son alto

sesgo y alta varianza, por lo tanto si tenemos:

- Alta varianza : $J_{\text{train}}(\theta)$ será pequeño y $J_{\text{val}}(\theta)$ será mucho mayor.
- Alto sesgo : $J_{\text{train}}(\theta)$ y $J_{\text{val}}(\theta)$ serán altos.$J_{\text{val}}(\theta) \approx J_{\text{train}}(\theta)$.

Los dos parámetros principales que pueden afectar a el trade-off entre sesgo y varianza son:

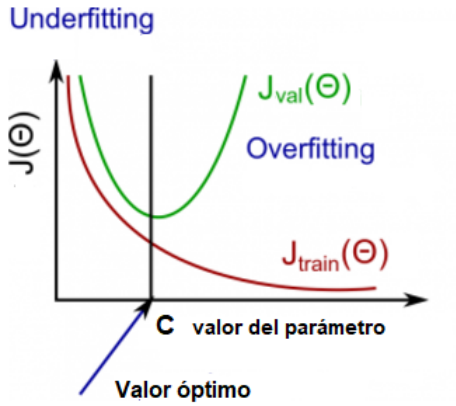
• **Complejidad de la función hipótesis**

El error de entrenamiento es menor conforme aumentamos la complejidad de $h_\theta(x)$ (grado del polinomio, número de variables independientes). Sin embargo, el error en el conjunto de validación sólo disminuye hasta cierto punto, a partir del cual nuestra función estará sobre-ajustada y no generalizará bien.

• **Hiperparámetro de regularización (C)**

Conforme hagamos el hiperparámetro pequeño la penalización de la regularización será muy grande y por lo tanto el valor de los parámetros θ tiende a tener valor cero para que la función de coste sea pequeña, entonces tenemos underfitting. Por el contrario, conforme $(C \to 0)$ tendemos a no regularizar $(R(\theta) \to 0)$ provocando overfitting si la hipótesis es suficientemente compleja. Una curva idealizada sería:

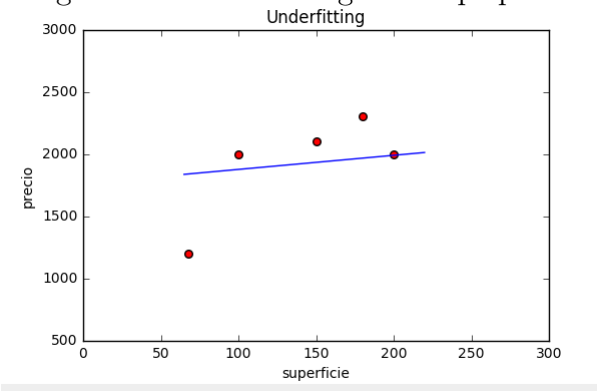
Figura 4.2: Parámetro C y sesgo-varianza



4.4. Curvas de aprendizaje

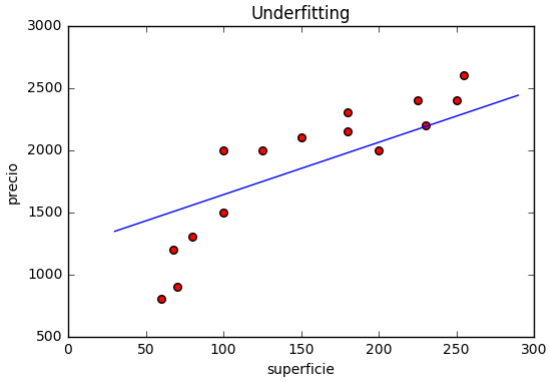
En la ciencia de los datos se suele decir que "cuantos más datos, mejor", en esta sección vamos a ver que esta afirmación no es del todo cierta [7]. Nos planteamos el ejemplo anterior, en el cual tenemos alto sesgo.

Figura 4.3: Underfitting con m pequeño



En este caso nuestra muestra de entrenamiento está formada por sólo cinco ejemplos de precios de hogares en función de la superficie ($m = 5$). Nuestra hipótesis es una recta, y no es capaz de representar correctamente la relación entre las dos variables. Supongamos que como investigador consigo una muestra de datos más grande sobre el estudio que estamos realizando.

Figura 4.4: Underfitting con m grande



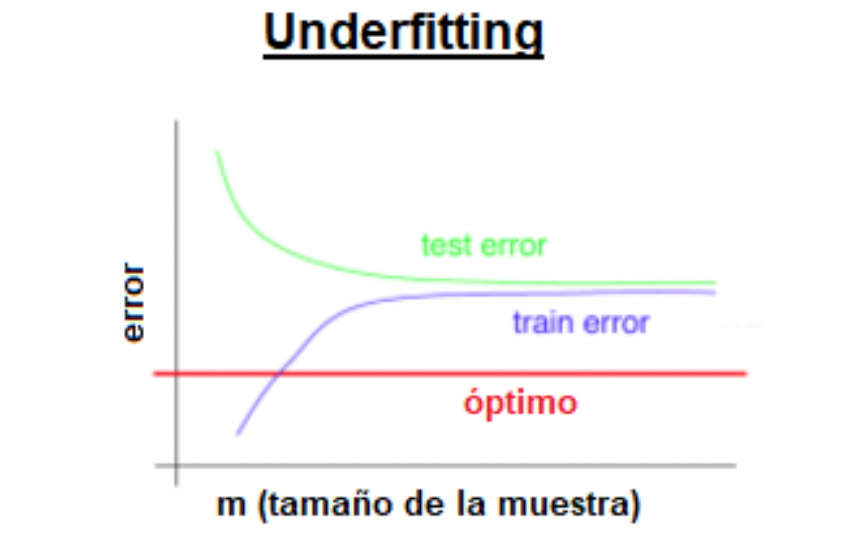
El hecho de aumentar el número de datos, en este caso ($m = 16$) no hará que nuestra hipótesis sea más precisa. De hecho, el coste aumentará al haber cada vez más puntos alejados (en la figura 4.4). Si pensamos en una muestra con ($m = 2$) una recta siempre será capaz de ajustarla perfectamente, siendo su error nulo. En consecuencia, si nuestro modelo sufre **underfitting**, dedicarle tiempo a recolectar nuevos datos **no tiene porque ayudar** por si sólo. En resumen:

• Alto sesgo (**underfitting**)

- Si m pequeño será $J_{\text{train}}(\theta)$ pequeño y $J_{\text{test}}(\theta)$ grande.

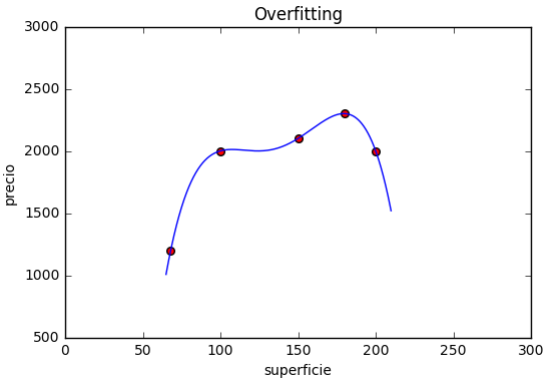
- Si m grande tanto $J_{\text{train}}(\theta)$ como $J_{\text{test}}(\theta)$ serán grandes.

Figura 4.5: Curva de aprendizaje con alto sesgo



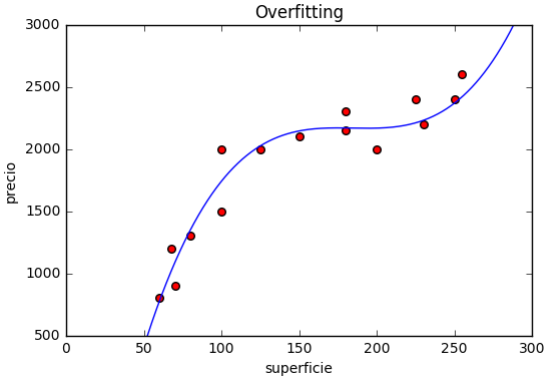
Ahora afrontamos el ejemplo anterior con alta varianza. La muestra del mismo tamaño pero con hipótesis un polinomio de grado alto.

Figura 4.6: Overfitting con m pequeño



En este caso nuestra hipótesis representa de manera perfecta la relación entre la superficie y el precio, sin embargo, no generaliza bien debido a que es demasiado específica para la muestra de entrenamiento. Según la hipótesis, conforme el hogar aumente su superficie a partir de los $180m^2$ su precio tiende a cero, lo cual no es razonable. Lo interesante es que conforme aumentemos el número de datos, nuestra hipótesis cada vez será mejor y por lo tanto representará la relación entre ambas variables. Si nuestro modelo sufre de alta varianza el hecho de aumentar nuestra muestra **probablemente ayude**.

Figura 4.7: Overfitting con m grande



- Alta varianza (**overfitting**)

- Si m pequeño será $J_{\text{train}}(\theta)$ pequeño y $J_{\text{test}}(\theta)$ grande.

- Si aumentamos m, $J_{\text{train}}(\theta)$ aumenta mientras que $J_{\text{test}}(\theta)$ disminuye progresivamente. Además, $J_{\text{train}}(\theta) < J_{\text{test}}(\theta)$ aunque su diferencia se va reduciendo.

Figura 4.8: Curva de aprendizaje con alta varianza

Esta sección nos ha servido para realizar una representación idealizada de las curvas de aprendizaje. Estas curvas nos muestran cómo se comportará nuestro modelo conforme aumentamos el número de datos. Aumentar el conjunto de datos no suele ser una tarea sencilla y es conveniente plantearse la realización de estas curvas de aprendizaje antes que dedicarle mucho esfuerzo, dado que este esfuerzo puede no tener una clara recompensa si el modelo sufre underfitting. Es interesante el hecho de que si nuestro modelo es demasiado complejo para nuestros datos de entrenamiento, aumentar su número nos permitirá "moldear" nuestra hipótesis, siendo cada vez más precisa.

En resumen, **es posible mejorar la calidad de nuestro modelo conforme aumentamos la cantidad de datos en el caso de que sufra overfitting**. Si el modelo sufre underfitting, desregularizar el modelo aumentando el parámetro C puede ayudar. También aumentando el grado del polinomio en nuestra hipótesis o utilizando más variables que expliquen la variable dependiente.

En el siguiente link se puede encontrar un ejemplo sencillo en Python dónde hemos calculado las curvas de aprendizaje para dos regresiones lineales, una con alto sesgo y otra con alta varianza.

Link: `https://github.com/PabloVargasIbarra/TFG_ML_Exercises/blob/master/LearningC.ipynb`

Esta evaluación de modelos se extrapola a muchos algoritmos de machine learning que no se tratan en este proyecto como maquinas vector soporte, redes neuronales, etc.

4.5. Coeficiente de determinación

El coeficiente de determinación (R^2) es una métrica para saber lo bien que ajusta un modelo. Corresponde a la proporción de varianza de la variable dependiente explicada por las variables independientes. Si

$$\overline{y} = \frac{1}{m}\sum_{i=1}^{m} y^{(i)} \qquad \mathrm{SS_T} = \sum_{i=1}^{m}(y^{(i)} - \overline{y})^2 \qquad \mathrm{SS_r} = \sum_{i=1}^{m}\Big(y^{(i)} - h_\theta(x^{(i)})\Big)^2 = \sum_{i=1}^{m}\epsilon^{(i)}$$

$\mathrm{SS_T}$ es proporcional a la varianza de la variable independiente, $\mathrm{SS_r}$ es la suma de los residuos del modelo. Entonces el coeficiente de determinación se define

$$\mathrm{R}^2 = 1 - \mathrm{SS_r}/\mathrm{SS_T}$$

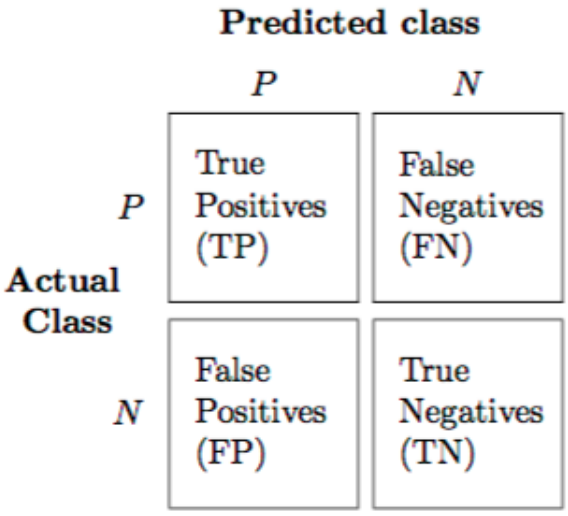
Si el modelo ajusta perfectamente y, entonces el coeficiente de determinación será uno. Si asignamos a cada $x^{(i)}$ la media de y como predicción, obtendríamos un coeficiente con valor 0.

4.6. Matriz de confusión

En los modelos de clasificación como la regresión logística, se suele utilizar la matriz de confusión para evaluar el modelo. En vez de elegir el modelo

con menor valor en la función de coste, se elige el que maximice distintas métricas que se obtienen de esta matriz. La matriz de confusión se define:

Figura 4.9: Matriz de confusión



TP = Casos en los que predigo $y = 1$ y es cierto.
TN = Casos en los que predigo $y = 0$ y es cierto.
FP = Casos en los que predigo $y = 1$ y es falso. Error de tipo 1.
FN = Casos en los que predigo $y = 0$ y es falso. Error de tipo 2.

Las métricas más utilizadas para un modelo clasificatorio y las preguntas a las que responden son :

¿Qué proporción de individuos ha clasificado correctamente mi modelo?

$$\textbf{Accuracy (Exactitud)} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

¿Qué proporción de individuos con $\hat{y} = 1$ he clasificado correctamente?

$$\textbf{Precisión} = \frac{\text{TP}}{TP + FP}$$

¿Qué proporción de individuos con $y = 1$ he clasificado correctamente?

$$\textbf{Sensibilidad} = \frac{\text{TP}}{TP + FN}$$

¿Qué proporción de individuos con $y = 0$ he clasificado correctamente?

$$\textbf{Especificidad} = \frac{\text{TN}}{TN + FP}$$

4.7. Frontera de decisión

En el caso particular de la regresión logística, tenemos un parámetro más que podemos variar a la hora de elegir el modelo. Este parámetro lo hemos denotado como p, límite de decisión. Siendo la clasificación binaria:

$$\hat{y} = \begin{cases} 1, & \text{si } h_\theta(x) \geq p \\ 0, & \text{en otro caso ó } h_\theta < p \end{cases} \tag{4.1}$$

Para entender la importancia de este parámetro, supongamos que $y = 1$ es la clase de un paciente que sufre cáncer. Predeciremos que lo tiene si $h_\theta(x) \geq p$, es decir, si su probabilidad es mayor o igual que el límite de decisión. Si nuestra prioridad como modelizador fuera asegurar que a todo paciente que le digamos que sufre cancer sea con una probabilidad alta, dado que es un golpe psicológico muy fuerte. Tomaríamos p alto. En cambio, si fuera asegurar que ningún paciente que sufre cáncer no le diagnostique como tal, tomaríamos límite de decisión bajo.

Conforme aumentamos p, aumenta la precisión mientras que se reduce la sensibilidad del modelo . Por el contrario, conforme reducimos p aumenta la sensibilidad y se reduce la precisión. Depende del modelizador darle mayor peso a una u otra elección. No se debe caer en el error de sólo maximizar una de las métricas. Por ejemplo, si nuestro objetivo fuese sólo maximizar la sensibilidad, podríamos clasificarlos todos los individuos como $\hat{y} = 1$ y seria máxima (valor 1) lo cual no es razonable.

4.8. Metodología Propuesta

En esta sección se propone una metodología que ayude a encontrar un modelo de regresión que sea capaz de generalizar para futuros datos. Según el tipo de variable dependiente, se elegirá la regresión lineal o logística.

Metodología

1. Dividir la muestra en entrenamiento, validación y test aleatoriamente.

2. Estandarizar las variables dependientes utilizando la media y desviación típica obtenida sobre el conjunto de entrenamiento.

3. Crear una lista de parámetros en la regularización $C \in \{C_1, C_2, C_3, ...\}$ y distintas hipótesis $h \in \{h_1, h_2, h_3, ...\}$.

2. Calcular los θ de la hipótesis iterando los distintos modelos para cada una de las combinaciones de h y C.

3. Calcular nuestra métrica de éxito de cada modelo sobre el conjunto de validación. Nos quedamos con la mejor combinación. (h^*, C^*)

4. El modelo final tiene hipótesis h^* y el parámetro de regularización C^*, su métrica de éxito real se evalúa sobre el conjunto de test.

5. Entrenamos el modelo final con todos los datos, nuestras futuras predicciones se harán con un input coherentes a la hipótesis elegida.

La elección de las distintas hipótesis y el parámetro de regularización se apoya en la teoría de esta sección. Ésta nos puede ayuda a tener una intuición previa sobre las combinaciones adecuadas.

4.9. Breast Cancer Prediction

Una de las aplicaciones más interesantes de los algoritmos de machine learning es la capacidad de detectar enfermedades y factores de riesgo. Una de las enfermedades más importantes es el cáncer, vamos a aplicar varios modelos de regresión logística con el objetivo de clasificar de la mejor manera posible pacientes que sufren tumores benignos y malignos.

La base de datos proviene de la plataforma "Kaggle"[4]. Su titulo es Wisconsin Diagnostic Breast Cancer y consta de 569 pacientes a los cuales se les han medido 30 variables reales. El objetivo es conseguir clasificar el tipo de tumor que sufren nuevos pacientes, el data set tiene 357 con un tumor benigno y 212 con un tumor maligno.

La resolución se encuentra en el siguiente link:

Link: `https://github.com/PabloVargasIbarra/TFG_ML_Exercises/blob/master/CancerPred.ipynb`

Se ha utilizado la metodología propuesta en el apartado anterior.

References

[1] Charlie Frogner. Bayesian interpretations of Regularization MIT. http://www.mit.edu/~9.520/spring09/Classes/class15-bayes.pdf [Accessed: Dic- 2017].

[2] Andrew Ng. CS229 Regularization and Model Selection. http://cs229.stanford.edu/notes/cs229-notes5.pdf [Accessed: Dic- 2017].

[3] Andrew Ng. CS229 Supervised Learning. http://cs229.stanford.edu/notes/cs229-notes1.pdf [Accessed: Dic- 2017].

[4] Breast Cancer Wisconsin Data Set. Kaggle. https://www.kaggle.com/uciml/breast-cancer-wisconsin-data [Accessed: Dic- 2017].

[5] F. Lindsten, T. B. Schön, A. Svensson, N. Wahlström. Probabilistic modeling -linear regression and Gaussian processes. http://www.it.uu.se/edu/course/homepage/sml/literature/probabilistic_modeling_compendium.pdf [Accessed: Dic- 2017].

[6] Hastie T, Tibshirani R, Friedman J (2013) The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer Series in Statistics.

[7] Machine Learning. Coursera. https://www.coursera.org/learn/machine-learning [Accessed: Dic- 2017].

[8] TFG ML Exercises Python Repository. https://github.com/PabloVargasIbarra/TFG_ML_Exercises/ [Feb- 2018].