REGRESIÓN LINEAL

INTRODUCCIÓN

La regresión logística, a pesar de su nombre, no es un algoritmo de regresión, sino un algoritmo de clasificación. Se le denomina “regresión” porque su formulación matemática es similar a la de la regresión lineal. Este método permite realizar clasificación binaria, es decir, problemas con solo dos clases, aunque también puede extenderse a clasificación multiclase.

Algunas aplicaciones comunes de la regresión logística en clasificación binaria incluyen la detección de correos electrónicos spam o la clasificación de tumores como benignos o malignos.

Si intentáramos resolver un problema de clasificación de tumores utilizando regresión lineal, sería necesario definir un valor umbral para tomar decisiones. Por ejemplo, si el modelo de regresión lineal predice un valor de 0.4 para un tumor maligno y el umbral se establece en 0.5, el sistema clasificaría incorrectamente el tumor como no maligno. En un contexto real, este tipo de error podría tener consecuencias graves.

Este ejemplo muestra que la regresión lineal no es adecuada para problemas de clasificación, ya que no está diseñada para producir probabilidades ni decisiones claras entre clases. Por ello, se requiere una técnica específicamente orientada a esta tarea: la regresión logística, que permite modelar probabilidades y tomar decisiones de clasificación de forma más confiable.

MODELO

En este algoritmo, la salida y se modela inicialmente como una función lineal de la variable de entrada x. Sin embargo, cuando y es binaria (por ejemplo, 0 o 1), el problema se vuelve más complejo que en la regresión lineal tradicional, ya que no basta con ajustar una recta para tomar decisiones de clasificación confiables.

Para poder realizar una clasificación adecuada, es necesario contar con información relevante del fenómeno que se desea clasificar. Por ejemplo, en el caso de la clasificación de tumores, la predicción puede basarse en una característica observable como el tamaño del tumor, tal como se ilustra en la Figura 1. A partir de esta información, el modelo puede aprender a diferenciar entre las distintas clases y estimar la probabilidad de que un tumor pertenezca a una u otra categoría.

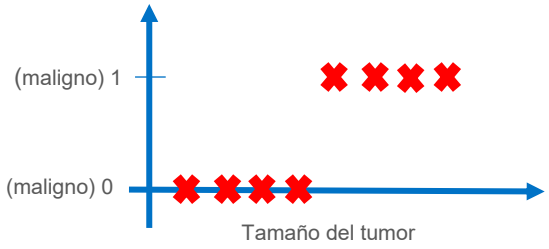

Fig 1. Tamaño de tumor vs clasificación

La ecuación lineal

$$h_\theta(x) = \theta_0 + \theta_1 x$$

es una función cuyo rango se extiende de $-\infty$ a $+\infty$, mientras que la variable de salida y en un problema de clasificación binaria solo puede tomar dos valores posibles. Por esta razón, la regresión lineal no es adecuada directamente para tareas de clasificación.

Para resolver este problema, se utiliza una función que transforme cualquier valor real en un número comprendido entre 0 y 1, permitiendo interpretar la salida como una probabilidad. Esta función se conoce como función sigmoide, y está definida como:

$$a(x) = \frac{1}{1+e^{-x}} \tag{1}$$

La gráfica de esta función se muestra en la Figura 2. Si se encuentran los valores óptimos de θ_0y θ_1, la salida de la función sigmoide puede interpretarse como la probabilidad de que $y$pertenezca a la clase positiva. Por ejemplo, en el caso de la clasificación de tumores, si se establece un umbral de 0.5, el modelo clasificará el tumor como maligno cuando la probabilidad estimada sea mayor o igual a dicho umbral, y como benigno en caso contrario.

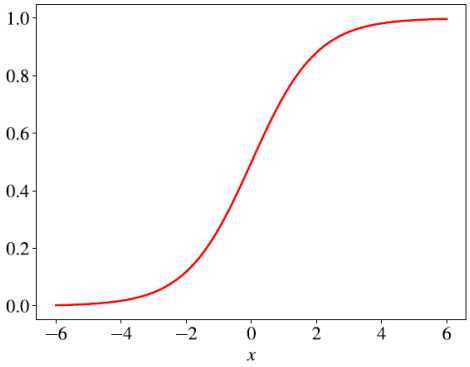

Fig.2. Función sigmoide

El modelo matemático de la regresión logística se expresa como:

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{2}$$

$$a(h_\theta(x)) = \frac{1}{1+e^{-h_\theta(x)}} \tag{3}$$

Supongamos ahora que se desea determinar la probabilidad de que un tumor sea maligno y que, para cierto valor de x, el modelo produce una salida $a(h_\theta(x)) = 0.7$. Este resultado se interpreta como que existe un 70% de probabilidad de que el tumor sea maligno.

FUNCIÓN DE COSTO

Una vez que se ha definido el modelo de regresión logística, el siguiente paso consiste en evaluar qué tan bien está realizando las predicciones. Para ello, se calcula el error mediante una función de costo, la cual cuantifica la diferencia entre las predicciones del modelo y los valores reales.

En el caso de la regresión logística, la función de costo más comúnmente utilizada es la entropía cruzada (cross-entropy loss), ya que es especialmente adecuada para problemas de clasificación binaria. Esta función penaliza fuertemente las predicciones incorrectas y permite que el modelo aprenda de manera más eficiente.

La función de costo de entropía cruzada se define como:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right] \tag{4}$$

donde:

- m: es el número total de muestras
- $y^{(i)}$: es la etiqueta real de la muestra i
- $\hat{y}^{(i)} = a\left(h_\theta\left(x^{(i)}\right)\right)$: es la probabilidad predicha por el modelo
- θ: representa el conjunto de parámetros del modelo

Esta función permite ajustar los parámetros θ de tal manera que la probabilidad predicha por el modelo se aproxime lo más posible a las etiquetas reales, lo que conduce a una mejor capacidad de clasificación.

OPTIMIZACIÓN

Después de calcular el error mediante la función de costo, es necesario actualizar los parámetros θ con el objetivo de minimizar dicha función. Este proceso permite que el modelo mejore progresivamente sus predicciones a partir de los datos disponibles.

Para llevar a cabo esta tarea, se utiliza el algoritmo de optimización más común en el aprendizaje supervisado, conocido como descenso del gradiente (*gradient descent*). Este algoritmo ajusta iterativamente los parámetros del modelo en la dirección opuesta al gradiente de la función de costo, es decir, en la dirección donde el error disminuye más rápidamente.

El algoritmo de descenso del gradiente está definido como:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{5}$$

Donde:

- θ_j representa el parámetro j-ésimo del modelo
- α es la tasa de aprendizaje, que controla el tamaño del paso en cada iteración
- $\frac{\partial J(\theta)}{\partial \theta_j}$ es la derivada parcial de la función de costo con respecto al parámetro θ_j

Este proceso se repite de manera iterativa hasta que la función de costo converge a un valor mínimo, logrando así un modelo de regresión logística correctamente entrenado.

CLASIFICACIÓN MULTICLASE

La clasificación multiclase se presenta cuando en un problema de aprendizaje automático intervienen más de dos clases. En este tipo de escenarios, el objetivo del modelo es asignar cada observación a una sola clase dentro de un conjunto de categorías posibles.

Por ejemplo, se puede realizar una clasificación de figuras geométricas, donde las clases correspondan a cuadrado, círculo, rectángulo y triángulo. En este caso, el modelo debe analizar las características de cada figura y determinar a cuál de estas categorías pertenece, en lugar de limitarse a una decisión binaria.

Este tipo de clasificación es ampliamente utilizado en aplicaciones como el reconocimiento de imágenes, análisis de texto y sistemas de recomendación, donde el número de clases puede ser mayor a dos y el modelo debe discriminar entre múltiples opciones de forma simultánea.

Una de las técnicas más utilizadas para resolver problemas de clasificación multiclase es el enfoque conocido como “uno contra todos” (*one-vs-all* o *one-vs-rest*). En este método, se selecciona una clase de referencia y todas las demás clases se agrupan y se consideran como una sola clase distinta a la clase de interés. De esta manera, el problema original se transforma en un problema de clasificación binaria.

Este procedimiento se repite de forma independiente para cada una de las clases presentes en el conjunto de entrenamiento. Como resultado, se entrenan tantos clasificadores binarios como clases existan. Durante la etapa de predicción, cada clasificador estima la probabilidad de que una muestra pertenezca a su clase correspondiente, y la clase final asignada es aquella que presenta la mayor probabilidad.

En otras palabras, la clasificación multiclase mediante el enfoque *uno contra todos* consiste en resolver múltiples problemas de clasificación binaria, uno por cada clase, para finalmente tomar una decisión global basada en los resultados de todos los clasificadores.

En las figuras 3 y 4 se presentan ejemplos de un problema de clasificación binaria y un ejemplo de clasificación multiclase, respectivamente.

En el caso de la clasificación binaria (Figura 3), el proceso de clasificación resulta relativamente sencillo. El algoritmo busca encontrar una frontera de decisión, representada por una línea recta, que permita separar adecuadamente las dos clases. Todos los puntos que se encuentran a un lado de la línea divisoria son asignados a la clase círculo, mientras que los puntos ubicados al otro lado son clasificados como pertenecientes a la clase cruz. Este tipo de problemas es característico de modelos como la regresión logística binaria, donde el objetivo principal es aprender una frontera que maximice la separación entre ambas clases en el espacio de características.

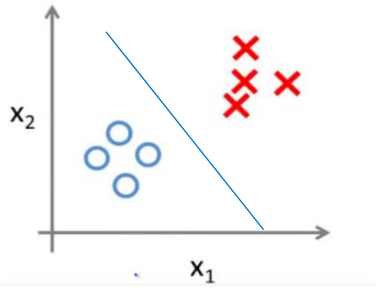

Fig.3. Clasificación binaria

Sin embargo, en un problema de clasificación multiclase, no es posible, en general, utilizar una sola línea recta para separar simultáneamente todas las clases. Por esta razón, se emplea una técnica conocida como “uno contra todos” (*one-vs-all*).

Esta técnica consiste en seleccionar una clase de interés y agrupar todas las demás clases bajo una única categoría denominada *“no pertenece a la clase”*. Por ejemplo, se puede tomar la clase triángulo como clase positiva y considerar las demás figuras como “no triángulo”, tal como se ilustra en la Figura 4. De este modo, el problema multiclase se transforma en un problema de clasificación binaria.

Este procedimiento se repite para cada una de las clases del conjunto de datos, permitiendo entrenar un clasificador independiente por clase y facilitando así la resolución del problema de clasificación multiclase.

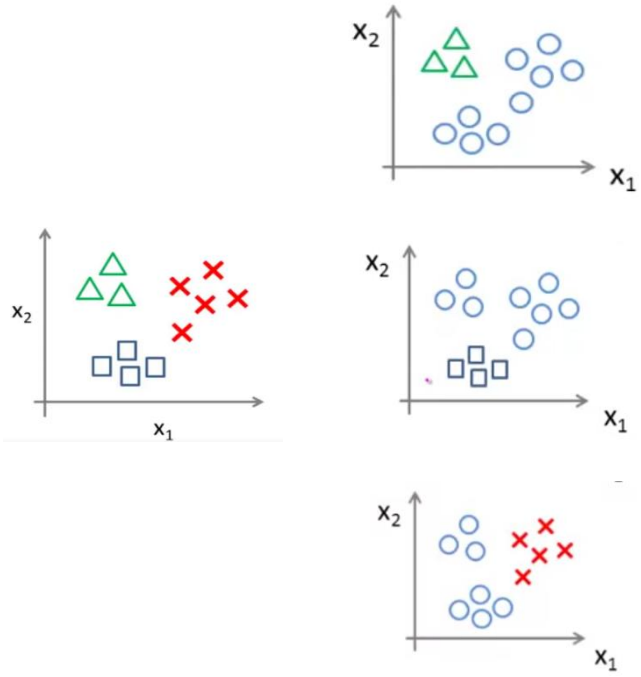

Fig.4. Clasificación multiclase

SOBRE AJUSTE

Una de las principales causas de obtener malos resultados en el aprendizaje automático es un fenómeno conocido como sobreajuste (*overfitting*). Durante el proceso de entrenamiento, se busca que el modelo (ya sea de regresión lineal o regresión logística) aprenda el comportamiento del sistema a partir de los datos disponibles. Sin embargo, cuando ocurre el sobreajuste, el modelo aprende en exceso los datos de entrenamiento y pierde la capacidad de generalizar a datos nuevos.

El objetivo fundamental de un algoritmo de inteligencia artificial es precisamente la generalización, es decir, que el modelo sea capaz de realizar predicciones o clasificaciones correctas sobre datos que no fueron utilizados durante el entrenamiento, siempre y cuando estos datos provengan del mismo fenómeno. Para lograrlo, es necesario contar con una cantidad suficiente y representativa de datos de entrenamiento.

Cuando el modelo se entrena en exceso, se presenta el problema de sobreajuste, lo que implica que el modelo obtiene un buen desempeño únicamente sobre los datos de entrenamiento, pero falla al enfrentarse a datos nuevos. Por el contrario, si el modelo no se entrena lo suficiente o es demasiado simple para representar el fenómeno, se dice que está

sub ajustado (*underfitting*), lo que significa que no logra capturar correctamente las relaciones existentes en los datos y requiere más información o mayor complejidad para generalizar adecuadamente.

El sobreajuste se puede detectar mediante el análisis de las curvas de entrenamiento, las cuales permiten monitorear el desempeño del modelo a lo largo del tiempo y detectar comportamientos no deseados, como el incremento del error en los datos de validación.

El proceso de generalización en los algoritmos de inteligencia artificial es similar a la forma en que los seres humanos aprendemos conceptos. Por ejemplo, si a un niño que nunca ha visto un perro se le muestra una serie fotografías de perros de raza *cocker* y se le dice que es un perro, es probable que, al mostrarle posteriormente la imagen de un perro de raza *san bernardo*, responda que no es un perro, ya que no se parece a lo que aprendió. En este caso, se dice que el "modelo" (el cerebro del niño) está sub ajustado, pues necesita más ejemplos para poder generalizar el concepto.

Ahora supongamos que el conjunto de aprendizaje consiste en una colección de 1,000 fotografías de perros pertenecientes a 10 razas diferentes. Si posteriormente se le muestra una imagen de un perro de una raza que no fue incluida en el conjunto original, el niño será capaz de generalizar y reconocer que se trata de un perro.

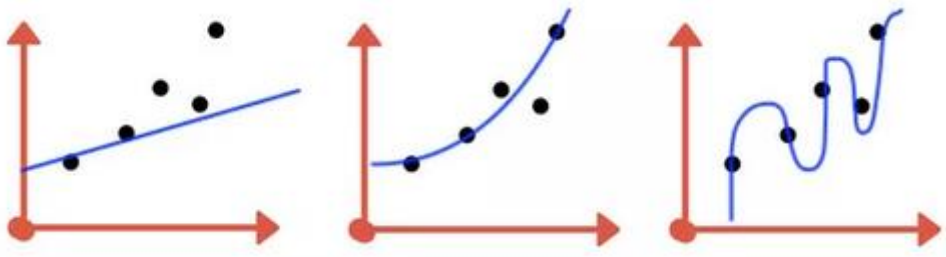

Fig.6. a) sub ajuste, b) buen ajuste, c) sobreajuste

En la Figura 6 se muestra de manera gráfica el comportamiento de un modelo bajo tres escenarios distintos: sub ajuste, sobreajuste y buena generalización de los datos. Estas representaciones permiten visualizar cómo el modelo se comporta frente a los datos de entrenamiento y de validación en cada caso.

Una de las técnicas más utilizadas para identificar si un sistema presenta problemas de sub ajuste o sobreajuste consiste en analizar las curvas de aprendizaje. Estas curvas representan la evolución del error de entrenamiento y del error de validación a lo largo del proceso de entrenamiento.

Cuando la diferencia entre ambas curvas es significativa, es decir, cuando la curva de error de entrenamiento es considerablemente menor que la curva de error de validación, se dice que el modelo presenta un problema de sobreajuste, tal como se observa en la Figura 7. En este caso, el modelo aprende muy bien los datos de entrenamiento, pero no logra generalizar correctamente a datos nuevos.

Por el contrario, cuando ambas curvas presentan valores de error elevados y similares, el modelo se encuentra sub ajustado, mientras que una buena generalización se refleja cuando ambas curvas convergen a valores bajos y cercanos entre sí.

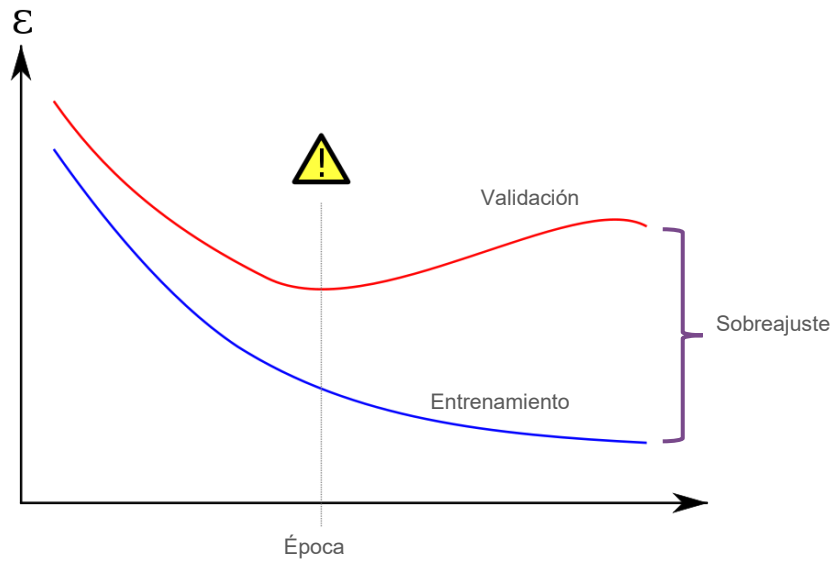

Fig.7. Curvas de entrenamiento. Entre más grande es la diferencia entre la curva de entrenamiento y validación, mayor es la magnitud de sobre ajuste

Para poder generar las dos curvas de aprendizaje (error de entrenamiento y error de validación), es necesario dividir el conjunto de datos disponible en dos subconjuntos: uno para entrenamiento y otro para validación. De manera general, esta división se realiza utilizando aproximadamente un 80 % de los datos para entrenar el modelo y el 20 % restante para validar su desempeño.

El sub ajuste ocurre cuando el modelo es demasiado simple, lo que le impide capturar adecuadamente la relación entre las variables y, por lo tanto, no logra generalizar el comportamiento de los datos. Este tipo de modelos suele presentar un alto sesgo (bias). Por otro lado, el sobreajuste aparece cuando el

modelo es excesivamente complejo y se adapta demasiado a los datos de entrenamiento, lo que provoca una alta varianza en las predicciones y un mal desempeño sobre datos nuevos.

Los conceptos de bias y varianza son fundamentales para diagnosticar el comportamiento de un modelo. Existe un compromiso entre ambos: al reducir el error asociado al bias, es posible incrementar el error debido a la varianza, y viceversa. Por ello, el objetivo es encontrar un equilibrio que permita una buena generalización.

Medidas para evitar el sub ajuste

Una vez identificado un problema de sub ajuste, se pueden aplicar las siguientes estrategias:

- Utilizar una mayor cantidad de ejemplos de entrenamiento en relación con el conjunto de validación.
- Si no se dispone de suficientes datos y la adquisición de nuevos ejemplos es costosa o complicada, se puede aplicar aumento artificial de datos, generando variaciones de los datos de entrenamiento existentes.
- Prolongar el tiempo de entrenamiento, permitiendo que el modelo aprenda mejor las relaciones presentes en los datos.

Medidas para evitar el sobreajuste

Cuando se detecta un problema de sobreajuste, es recomendable aplicar las siguientes acciones:

- Utilizar clases balanceadas, es decir, contar con una cantidad similar de ejemplos para cada clase.
- Seleccionar cuidadosamente los hiper parámetros del modelo, como el número de épocas de entrenamiento o el valor del ritmo de aprendizaje.
- Evitar conjuntos de datos con un número excesivo de dimensiones; en caso necesario, reducir la dimensionalidad mediante técnicas como PCA (Análisis de Componentes Principales).
- Evitar el uso de modelos con una cantidad excesiva de parámetros.
- Implementar técnicas de regularización.

REGULARIZACIÓN

La regularización es una de las técnicas más utilizadas para prevenir el sobreajuste en los algoritmos de inteligencia artificial. Dado que la función de costo de la regresión logística está definida en la ecuación (4), la función de costo regularizada se expresa como:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\ln\left(h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right)\ln\left(1-h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \qquad (6)$$

El término de regularización

$$\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

excluye el parámetro θ_0, ya que este corresponde al sesgo del modelo. De esta manera, las ecuaciones de actualización mediante descenso del gradiente quedan definidas como:

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0^{(i)} \qquad (7)$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\theta_j\right], j \geq 1 \qquad (11)$$

Estas ecuaciones permiten controlar la complejidad del modelo, penalizando valores grandes de los parámetros y favoreciendo así una mejor capacidad de generalización.