Clase 3: Regresión Logística

De la Predicción Numérica a la Clasificación Probabilística

Matías Leoni

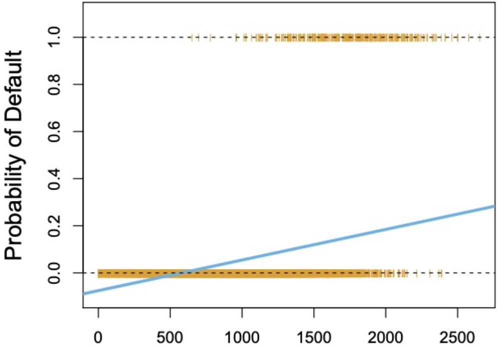
Maestría en IA - Aprendizaje Automático I

1 de julio de 2025

¿Qué aprenderemos hoy?

¿Por qué no usar Regresión Lineal para Clasificar?

- La regresión lineal predice valores continuos, pero la clasificación necesita predecir categorías discretas.
- Al codificar clases como 0 y 1, el modelo lineal puede predecir valores fuera del rango $[0, 1]$, que no son interpretables como probabilidades.
- La relación lineal no captura la naturaleza de "salto" probabilístico inherente a la clasificación.



Introducción a la Clasificación Binaria

- La clasificación consiste en asignar una observación a una categoría o clase predefinida.
- En la clasificación binaria, solo hay dos resultados posibles, a menudo llamados "clase positiva" (1) y "clase negativa" (0).
- **Ejemplos de aplicación:**
  - Detección de fraude: ¿La transacción es *fraudulenta* o *legítima*?
  - Diagnóstico médico: ¿El paciente tiene una *condición* o *no la tiene*?
  - Riesgo crediticio: ¿El cliente entrará en *default* o *no default*?

La Función Sigmoide (o Logística)

- Para modelar una probabilidad, necesitamos una función que transforme cualquier entrada real a un valor en el intervalo $[0, 1]$.
- La función sigmoide tiene una característica forma de "S" que es ideal para este propósito.
- Su fórmula es:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Donde z es la salida de un modelo lineal:

$$z = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

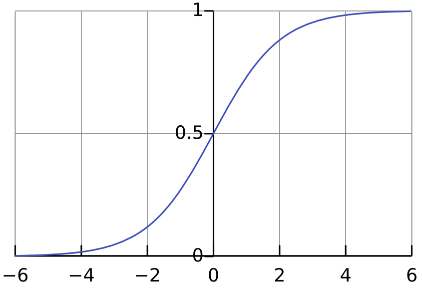


Figura: Gráfico de la función sigmoide.

De la Salida a la Probabilidad Estimada

- La regresión logística modela la probabilidad de que una observación pertenezca a la clase positiva, dado un conjunto de predictores X.
- **Notación del Modelo:**

$$\hat{p}(X) = P(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

- Un resultado $\hat{p}(X) = 0{,}9$ significa que el modelo estima una probabilidad del 90 % de que la observación pertenezca a la clase 1.

El Logit: Despejando la Linealidad

- Manipulando la ecuación del modelo, encontramos una relación lineal subyacente.
- El **Odds Ratio** (razón de momios) se define como:

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}$$

- Al tomar el logaritmo natural, obtenemos el **Logit** o **Log-Odds**:

$$\ln\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

- **Interpretación Clave:** Un aumento de 1 unidad en X_j cambia el log-odds de la clase positiva en β_j unidades.

Encontrando los Mejores Parámetros β

- En lugar de minimizar la suma de errores al cuadrado, buscamos los coeficientes β que maximizan la probabilidad de observar los datos de entrenamiento que tenemos.
- Este principio se llama **Estimación por Máxima Verosimilitud (Maximum Likelihood Estimation - MLE)**.
- La **función de verosimilitud (likelihood)** para un modelo de clasificación binaria es:

$$\mathcal{L}(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

De Maximizar a Minimizar: Log-Loss

- En machine learning, es convencional minimizar una función de costo en lugar de maximizar una de verosimilitud.
- Minimizar el **logaritmo negativo de la verosimilitud** es equivalente y computacionalmente más estable.
- Esto nos da la **Función de Costo Logística** o **Entropía Cruzada Binaria**:

$$J(\beta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

- La función penaliza fuertemente las predicciones seguras pero incorrectas.

Teorema: Convexidad de la Función de Costo

Resultado Fundamental

La función de costo de Entropía Cruzada (Log-Loss) para la Regresión Logística, $J(\beta)$, es una función **convexa** con respecto a los parámetros β.

- **Implicación Directa:** Una función convexa no tiene mínimos locales; cualquier mínimo encontrado es el **mínimo global**.
- **Punto de partida para la derivación:**
  - Demostraremos que la matriz Hessiana de $J(\beta)$, $\nabla^2 J(\beta)$, es semidefinida positiva para todo β.

¿Por qué es tan importante esta propiedad?

- **Convergencia Garantizada:** El Gradiente Descendente, sin importar el punto de inicio, tiene la garantía de converger a la **única mejor solución posible**.
- **Alternativa Contrafáctica:** ¿Qué pasaría si usáramos el Error Cuadrático Medio (MSE) con la salida sigmoide?

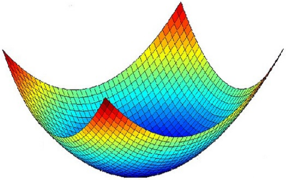
$$J_{MSE}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \sigma(\beta^T x_i))^2$$

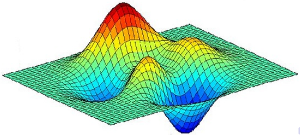
¿Por qué es tan importante esta propiedad?

- **Convergencia Garantizada:** El Gradiente Descendente, sin importar el punto de inicio, tiene la garantía de converger a la **única mejor solución posible**.
- **Alternativa Contrafáctica:** ¿Qué pasaría si usáramos el Error Cuadrático Medio (MSE) con la salida sigmoide?

$$J_{MSE}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \sigma(\beta^T x_i))^2$$

Esta función de costo J_{MSE} **no es convexa**. Presenta múltiples mínimos locales, haciendo la optimización extremadamente difícil y dependiente de la inicialización.



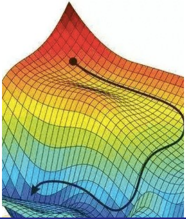


Encontrando el Mínimo de la Función de Costo

- Como ya estudiamos, el Gradiente Descendente es el algoritmo iterativo que usamos para encontrar los valores de β que minimizan $J(\beta)$.
- En cada paso, ajustamos los parámetros en la dirección opuesta al gradiente (derivada) de la función de costo.
- **Regla de Actualización de Parámetros:**

$$\beta_j := \beta_j - \alpha \frac{\partial J(\beta)}{\partial \beta_j}$$

- α es la **tasa de aprendizaje (learning rate)**, un hiperparámetro crucial que controla el tamaño de cada paso.



Analizando Múltiples Predictores

- Al igual que en regresión lineal, podemos usar múltiples predictores $(X_1, X_2, \ldots, X_p)$.
- **¡Cuidado!** El efecto de un predictor puede cambiar drásticamente cuando se incluyen otros. Este fenómeno se conoce como **confusión (confounding)**.
- **Ejemplo:**
    - Individualmente, ser estudiante parece *aumentar* la probabilidad de default.
    - Pero al incluir el `balance` de la tarjeta, ser estudiante *disminuye* la probabilidad de default para un `balance` fijo.
- **Explicación:** Los estudiantes tienden a tener balances más altos, y el balance es el verdadero motor del riesgo.

De Probabilidades a Decisiones de Negocio

- El modelo produce una probabilidad $\hat{p}$. Para clasificar, usamos un **umbral de decisión**. Por defecto es 0.5.
- Ajustar el umbral es clave. Para detectar fraude, podríamos bajarlo a 0.2 para ser más sensibles.
- La **matriz de confusión** organiza los resultados:

	Real: Positivo	**Real: Negativo**
Predicho: Positivo	Verdadero Positivo (TP)	Falso Positivo (FP)
Predicho: Negativo	Falso Negativo (FN)	Verdadero Negativo (TN

- **FP** es un Error de Tipo I.
- **FN** es un Error de Tipo II.

Evaluando el Rendimiento Más Allá del Accuracy

- El **Accuracy** ($\frac{TP+TN}{\text{Total}}$) es engañoso en datasets desbalanceados.
- **Precisión:** De todos los que predijimos como positivos, ¿cuántos acertamos? Mide la calidad de las predicciones positivas.

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

- **Recall (Sensibilidad):** De todos los positivos reales, ¿cuántos fuimos capaces de identificar? Mide la capacidad de encontrar a todos los positivos.

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

- **F1-Score:** La media armónica de Precisión y Recall. Busca un balance entre ambas.

$$F1 = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

- La **Curva ROC** evalúa el clasificador a través de todos los umbrales posibles.
- Grafica **Recall (TPR)** vs. **Tasa de Falsos Positivos (FPR)**.
- El **Área Bajo la Curva (AUC)** resume este rendimiento en un solo valor.
  - AUC = 1: Clasificador perfecto.
  - AUC = 0.5: Azar.
- Es ideal para comparar modelos de forma independiente al umbral.

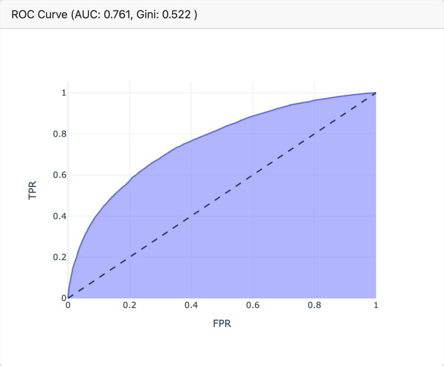


Figura: Ejemplo de una Curva ROC.

Combatiendo el Sobreajuste (Overfitting)

- El sobreajuste ocurre cuando el modelo aprende el ruido de los datos, resultando en coeficientes β muy grandes.
- La **regularización** añade un término de penalización a la función de costo para desincentivar coeficientes grandes.
- **Regularización L2 (Ridge):** La más común. Penaliza la suma de los cuadrados de los coeficientes.

$$J_{reg}(\beta) = J(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- **Regularización L1 (Lasso):** Penaliza la suma de los valores absolutos. Puede llevar coeficientes a cero (selección de variables).

$$J_{reg}(\beta) = J(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

¿Y si hay más de dos clases?

- La regresión logística se generaliza para manejar $K > 2$ clases. Se conoce como **Regresión Logística Multinomial**.
- **Softmax:** Una formulación simétrica que calcula la probabilidad de cada clase directamente. Es el estándar en redes neuronales.
- $$P(Y = k|X) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_k}}, \text{donde} \quad z_k = \beta_k^T \mathbf{x}$$

  .
- **Alternativa: One-vs-Rest (OvR)**. Entrena un clasificador binario por cada clase, comparando contra el resto.

¿Dónde se ubica la Regresión Logística?

Modelos Discriminativos

La Regresión Logística es un modelo discriminativo.

- Modela directamente la probabilidad condicional $P(Y|X)$.
- Aprende una **frontera de decisión** que separa las clases.
- No se preocupa por cómo se generaron los datos de cada clase.

Modelos Generativos

Son una alternativa.

- Modelan la distribución de los predictores para cada clase, $P(X|Y)$.
- Usan el Teorema de Bayes para calcular $P(Y|X)$.
- Ejemplos: Análisis Discriminante Lineal (LDA), QDA, Naive Bayes.

Resumen y Próximos Pasos

- **Resumen de Hoy:** Hemos cubierto la teoría, entrenamiento, evaluación y extensiones de la regresión logística, un pilar fundamental de la clasificación.
- **Próxima Hora:** Taller práctico.
  - Implementaremos un modelo de regresión logística en Python.
  - Usaremos `scikit-learn` para cargar datos, entrenar y evaluar.
  - Interpretaremos las métricas en un problema real.