

Módulo 4: Aprendizaje Automático

Temario de la Clase 2

16 de agosto del 2024

Modelos lineales para clasificación

- Tipos de clasificación
- Consideraciones iniciales
- Estimadores de máxima verosimilitud
- Clasificación binaria
- Clasificación multi-clase
- Funciones de verosimilitud
- Funciones de activación

Diferencia entre regresión y clasificación

Como se dijo anteriormente, la diferencia entre estos conceptos es que la **regresión** busca calcular un valor numérico (cantidades, dimensiones, precios, etc.) mientras que la **clasificación** busca determinar a qué categoría (o categorías) pertenece una observación o ejemplo que sirve de entrada al modelo.

Veremos que ningún modelo puede predecir con absoluta certeza que una entrada pertenece a una determinada categoría, por lo que debemos trabajar con la idea de **probabilidades**.

Al problema de clasificación también se le suele llamar **regresión logística**.

Tipos de clasificación

Existen dos problemas sutilmente diferentes:

- (i) aquellos en los que sólo nos interesan asignaciones estrictas de ejemplos a categorías (clases);
- (ii) aquellos en los que deseamos realizar asignaciones suaves, es decir, evaluar la probabilidad de que se aplique cada categoría.

Podemos distinguir entre dos tipos de clasificación:

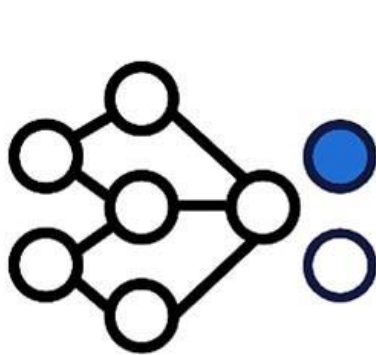
- Clasificación binaria: tenemos dos posibles categorías. Es el caso más sencillo.
- Clasificación multi-clase: tenemos más de dos categorías posibles.

Tipos de clasificación

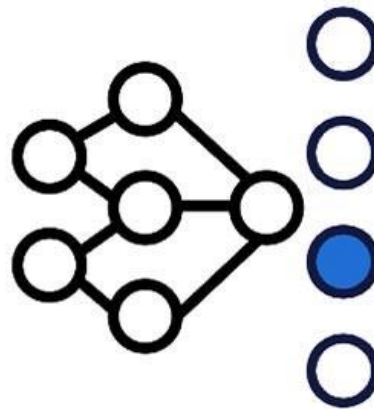
Incluso dentro de la clasificación en varias categorías podemos diferenciar dos casos muy distintos.

- Multi-clase: cuando cada elemento puede pertenecer a solo una categoría. Si buscamos reconocer un dígito numérico, el mismo puede ser {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, pero nunca pertenecer a dos categorías simultáneamente.
- Multi-label o multi-etiqueta: cuando cada elemento puede pertenecer a varias categorías. Por ejemplo, un correo electrónico puede ser categorizado como “importante”, “trabajo” y “personal” al mismo tiempo.

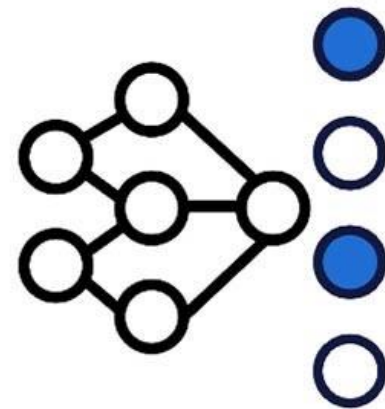
Tipos de clasificación



Binary



Multi-Class



Multi-Label

Consideraciones iniciales

Si bien los modelos lineales vistos para regresión pueden ser utilizados para resolver problemas de clasificación, las limitaciones son muchas.

Por lo tanto vamos a reformular algunas cuestiones para obtener modelos mejor orientados.

Un modelo de regresión puede tener como salida cualquier valor real. Sin embargo, puesto que en clasificación trabajamos con probabilidades, es mejor construir modelos cuyas salidas estén acotadas en el rango $[0, 1]$.

En la clasificación multi-clase, también queremos que la suma de las probabilidades de la entrada respecto de las múltiples clases sea igual a 1, o equivalentemente, al 100%.

Estimadores de máxima verosimilitud

Los parámetros de un modelo se ajustan para que el mismo tenga una buena respuesta sobre un conjunto X compuesto por N datos de entrada independientes e idénticamente distribuidos entre sí.

Si nuestro modelo consta de un conjunto de parámetros θ se establece que existe una familia de funciones paramétricas que dependen de θ y que se evalúan sobre los datos disponibles X .

Se pueden establecer tantos modelos como conjuntos de parámetros tengamos.

$$y_i = f(X|\theta_i) = f(x_1, x_2, \dots, x_n | \theta_i)$$

Estimadores de máxima verosimilitud

Puesto que los datos son independientes se calcula la función de verosimilitud de la siguiente forma:

$$\mathcal{L}(\theta \mid y_1, y_2, \dots, y_n) = \prod_{k=1}^N f(y_i \mid \theta)$$

$$\mathcal{L}(\theta \mid y_1, y_2, \dots, y_n) = f(\theta \mid y_1) * f(\theta \mid y_2) * \dots * f(\theta \mid y_n)$$

Donde ahora, los valores Y son fijos mientras que hacemos variar θ . El objetivo es encontrar el conjunto de parámetros θ_i que maximice la función de verosimilitud \mathcal{L} y haga más probable la salida del modelo para los datos de entrada.

$$\theta_{optimo} = \arg \max \mathcal{L}(\theta \mid Y)$$

Estimadores de máxima verosimilitud

Se puede demostrar que para el caso de los modelos lineales de regresión vistos en la clase pasada, buscar los parámetros que **maximicen** la **función de verosimilitud** es equivalente a buscar los parámetros que **minimicen** la **función de pérdida**.

En ML es más común trabajar con algoritmos que minimicen determinadas cantidades.

Clasificación binaria

Es el proceso de categorizar datos en dos grupos distintos. Por ejemplo, decidir si un correo es "spam" o "no spam", o determinar si una imagen contiene un gato o no.

Se utiliza para modelar la probabilidad de que una muestra pertenezca a una de las dos clases posibles. Se utiliza la función logística (o sigmoide) para transformar una combinación lineal de características en una probabilidad.

En este modelo consideraremos una sola salida

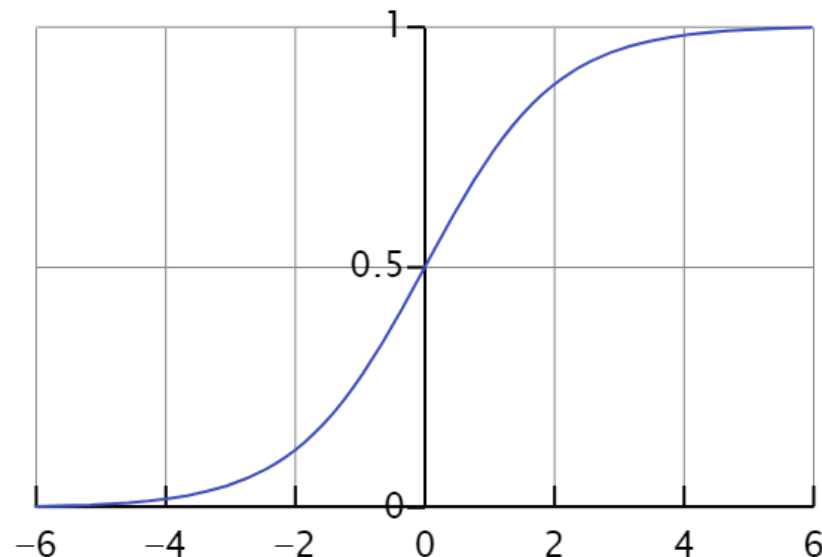
Aplicaciones comunes:

- Diagnóstico médico (e.g., detección de enfermedades a partir de características clínicas)
- Detección de fraude (e.g., identificar transacciones fraudulentas)
- Clasificación de correos electrónicos (e.g., detección de spam)

Función sigmoide

Es una función que nos permite mapear una entrada que puede pertenecer a todo el dominio real $(-\infty, \infty)$ hacia una salida contenida en el rango $[0, 1]$. Esto nos permite trabajar con valores apropiados para el cálculo de probabilidades.

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

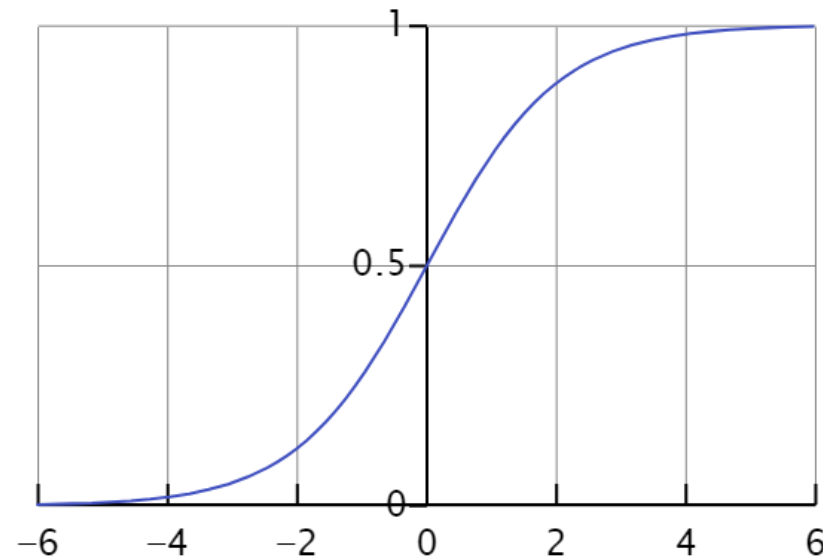


Función sigmoide

En este caso z representa la salida de nuestro modelo lineal. La presencia de la función sigmoide representa una transformación del modelo. Es un caso particular de las denominadas **funciones de activación**, que veremos más adelante en redes neuronales.

$$z = \mathbf{w}^T \mathbf{x} + b = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

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



Modelo de clasificación binaria

Puesto que la salida de la función sigmoide nos dará un valor entre $[0, 1]$ podemos establecer un umbral de probabilidad de valor 0.5 que nos determine que nuestro modelo predice que la entrada pertenece a la clase 0 o a la clase 1.

Entonces tenemos:

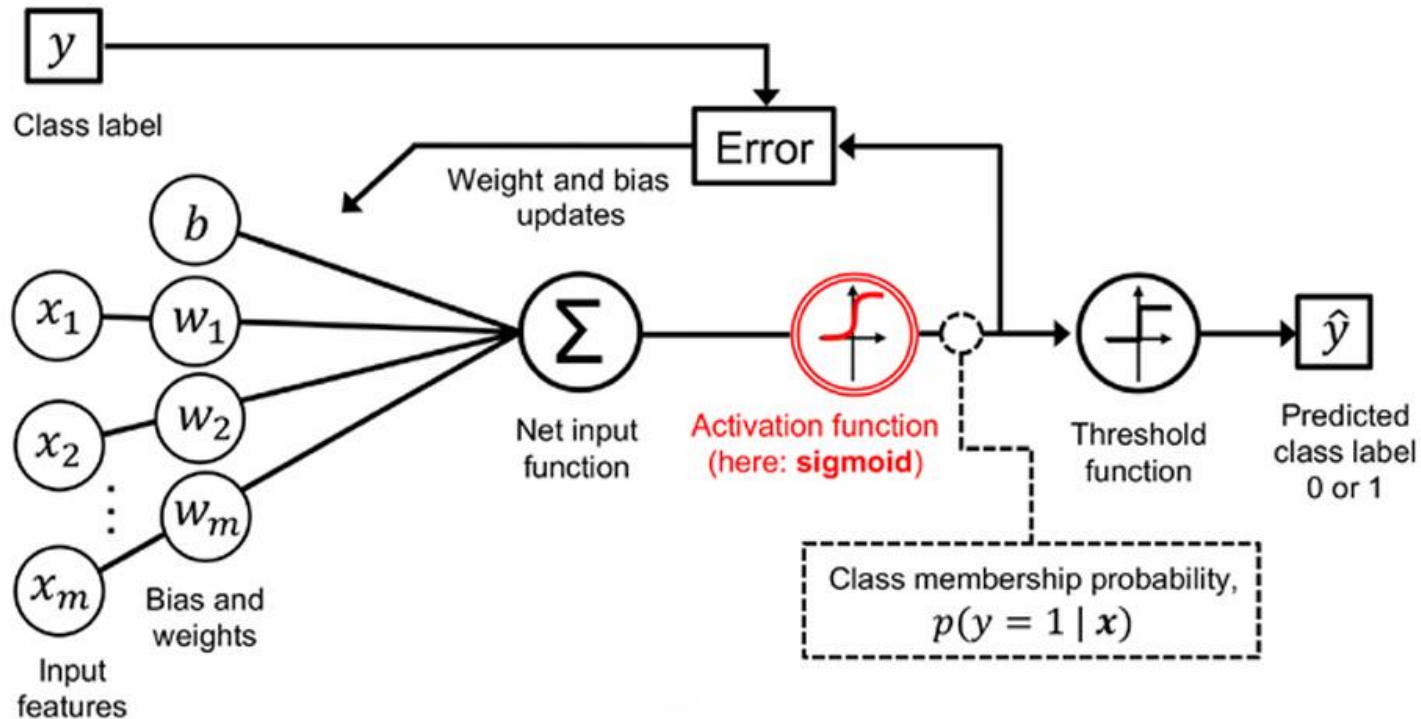
$$\hat{y} = \begin{cases} 1, & \sigma(z) \geq 0.5 \\ 0, & \sigma(z) < 0.5 \end{cases}$$

O lo que es equivalente:

$$\hat{y} = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

Modelo de clasificación binaria

Podemos visualizar nuestro modelo de la siguiente manera:



Función de verosimilitud

Habíamos dicho que la optimización del modelo consistía en encontrar los parámetros que maximicen la función de verosimilitud, la cual puede ser expresada como:

$$L(w, b) = \prod_{i=1}^m P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b)$$

Donde:

- m , es el número de ejemplos de entrenamiento.
- $y^{(i)}$ es la etiqueta real para el i -ésimo ejemplo.
- $P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b)$ es la probabilidad predicha.

Función de verosimilitud

Para clasificación binaria, podemos escribir la probabilidad como:

$$P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) = \begin{cases} \hat{y}^{(i)}, & y^{(i)} = 1 \\ (1 - \hat{y}^{(i)}), & y^{(i)} = 0 \end{cases}$$

$$P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) = (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{1-y^{(i)}}$$

$$P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) = (\sigma(z^{(i)}))^{y^{(i)}} (1 - \sigma(z^{(i)}))^{1-y^{(i)}}$$

Función de log-verosimilitud

Puesto que el cálculo de dicha función involucra el producto de varios valores, esto puede causar un overflow o underflow numérico. Por ello es conveniente tomar el logaritmo de dicha función:

$$L(\mathbf{w}, b) = \prod_{i=1}^m (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{1-y^{(i)}}$$

$$\log L(\mathbf{w}, b) = \sum_{i=1}^m \log P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b)$$

$$\log L(\mathbf{w}, b) = \sum_{i=1}^m [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Función de log-verosimilitud negativa

Puesto que es más común minimizar un parámetro en lugar de maximizarlo, podemos establecer el opuesto de la función de verosimilitud logarítmica como nuestra función a minimizar. Entonces la **función de pérdida** se define como:

$$Loss(\mathbf{w}, b) = -\log L(\mathbf{w}, b)$$

$$Loss(\mathbf{w}, b) = -\sum_{i=1}^m [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Esta función se denomina **entropía cruzada binaria** (binary cross-entropy).

Función de entropía cruzada binaria

El primer término $y^{(i)} \log(\hat{y}^{(i)})$, representa la pérdida cuando el valor verdadero es 1, mientras que el segundo término $(1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$ representa la pérdida cuando el valor verdadero es 0. La función se minimiza cuando la salida predicha $\hat{y}^{(i)}$ se acerca a la salida real $y^{(i)}$.

Cuantifica la diferencia entre las probabilidades predichas y las etiquetas reales. La ecuación de la función de pérdida se puede derivar combinando la función logística con el concepto de probabilidad.

Es una medida utilizada para evaluar el rendimiento de un modelo de clasificación binaria, es decir, un modelo que predice una de dos clases posibles.

Esta función compara las probabilidades predichas por el modelo con las etiquetas reales de los datos, penalizando más fuertemente las predicciones incorrectas.

Calculo del gradiente

Para entrenar nuestro modelo vamos a optimizar los parámetros en base a minimizar la función de entropía cruzada binaria.

Reemplazando $a = \sigma(z) = \frac{1}{1+e^{-z}}$, y sabiendo que $z = \mathbf{w}^T \mathbf{x} + b$

$$L = - \sum_{i=1}^m y \log a + (1 - y) \log(1 - a)$$

Debemos calcular el gradiente respecto a los j-ésimos pesos y el bias.

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w_j}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial b}$$

Calculo del gradiente

Reemplazando $a = \sigma(z) = \frac{1}{1+e^{-z}}$, y sabiendo que $z = \mathbf{w}^T \mathbf{x} + b$

$$L = - \sum_{i=1}^m y \log a + (1 - y) \log(1 - a)$$

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial w_j}$$

$$\frac{\partial L}{\partial a} = \sum_{i=1}^m \frac{a - y}{a(1 - a)}$$

$$\frac{\partial a}{\partial z} = \sum_{i=1}^m a(1 - a)$$

$$\frac{\partial z}{\partial w_j} = \sum_{i=1}^m x_j$$

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^m (a - y) x_j$$

Calculo del gradiente

Reemplazando $a = \sigma(z) = \frac{1}{1+e^{-z}}$, y sabiendo que $z = \mathbf{w}^T \mathbf{x} + b$

$$L = - \sum_{i=1}^m y \log a + (1 - y) \log(1 - a)$$

$$\begin{aligned} \frac{\partial L}{\partial a} &= \sum_{i=1}^m \frac{a - y}{a(1 - a)} & \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial b} \\ \frac{\partial a}{\partial z} &= \sum_{i=1}^m a(1 - a) & \frac{\partial z}{\partial b} &= \sum_{i=1}^m 1 \\ \frac{\partial L}{\partial b} &= \sum_{i=1}^m (a - y) \end{aligned}$$

Actualización de parámetros

Entonces trabajando sobre un batch de m -ejemplos de entrenamiento, los pesos y el bias se actualizan como:

$$w_j^{(i+1)} = w_j^{(i)} - \eta \frac{1}{m} \sum_{i=1}^m (a - y) x_j$$

$$b^{(i+1)} = b^{(i)} - \eta \frac{1}{m} \sum_{i=1}^m (a - y)$$

Donde η es el learning rate.

Clasificación multi-clase

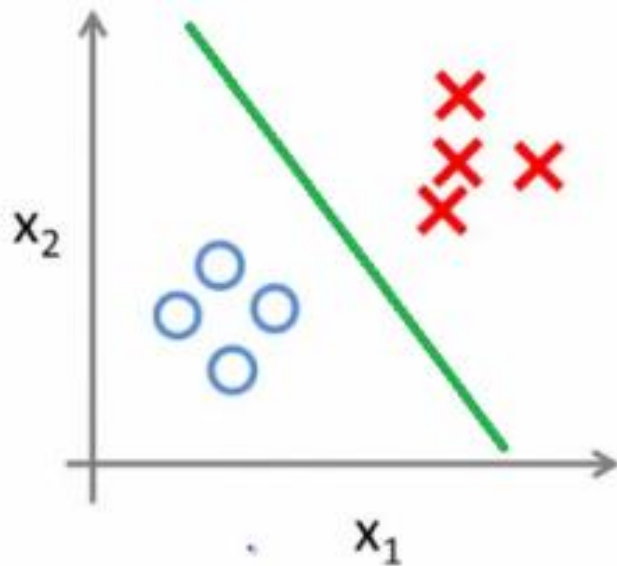
Este tipo de clasificación asigna cada muestra de datos a una de k clases distintas. Por ejemplo, en un problema de clasificación de imágenes, el modelo podría clasificar imágenes de animales en categorías como "gato", "perro", "pájaro", "pez", etc.

Aplicaciones comunes:

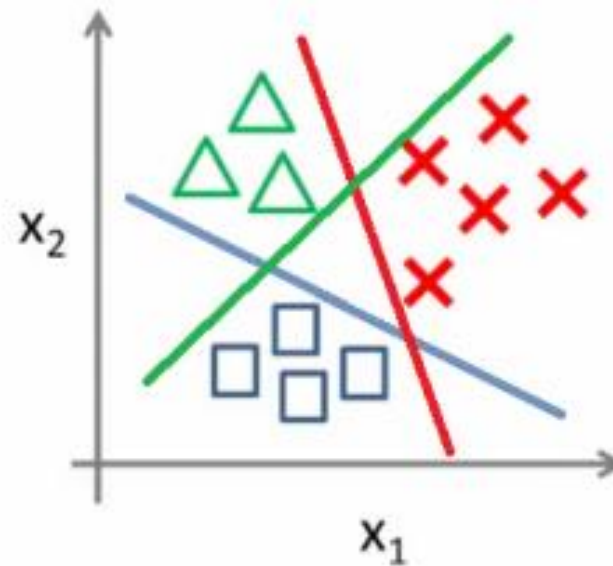
- Reconocimiento de imágenes (clasificación de objetos en imágenes)
- Procesamiento del lenguaje natural (categorización de temas en textos)
- Diagnóstico médico (clasificación de diferentes tipos de enfermedades)

Clasificación multi-clase

Binary classification:

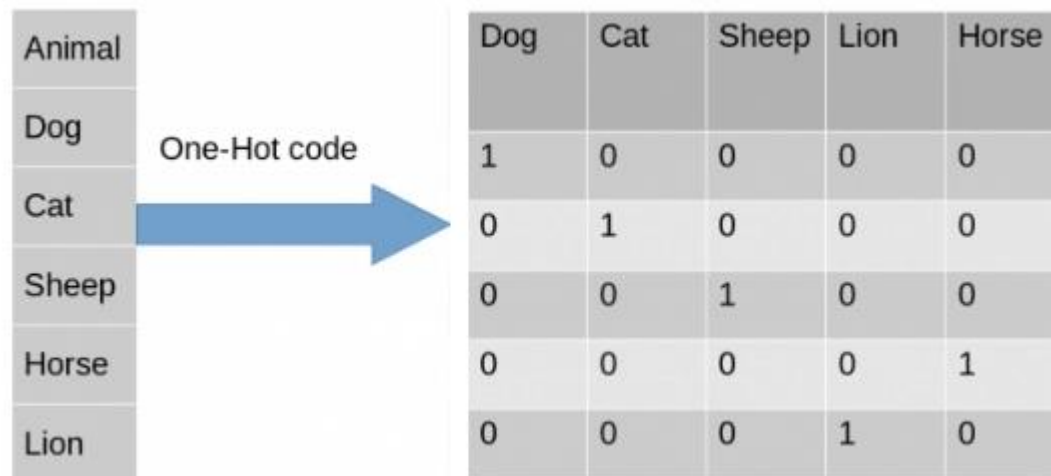


Multi-class classification:



Codificación one-hot

En la clasificación multi-clase se utiliza el one-hot encoding para que el modelo pueda determinar la salida deseada durante el proceso de entrenamiento. Consiste en considerar la salida y como un vector con tantas componentes como clases haya. Cada componente corresponde a una clase específica y la componente cuya clase coincida con el ejemplo en cuestión contendrá un 1, mientras que todas las otras componentes contendrán un 0.



Animal		Dog	Cat	Sheep	Lion	Horse
Dog		1	0	0	0	0
Cat		0	1	0	0	0
Sheep		0	0	1	0	0
Horse		0	0	0	0	1
Lion		0	0	0	1	0

Codificación one-hot

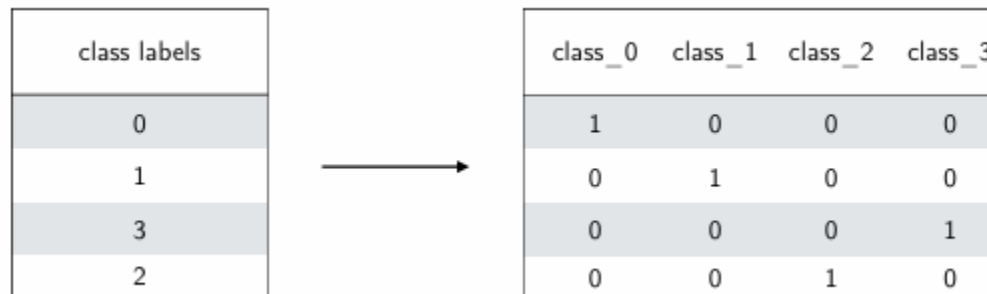
Si por ejemplo queremos clasificar entre 4 categorías, nuestro dataset de entrenamiento debe contener en la salida solamente vectores con las siguientes formas:

[1, 0, 0, 0]

[0, 1, 0, 0]

[0, 0, 1, 0]

[0, 0, 0, 1]



Modelo

Ahora en lugar de tener 1 sumatoria de términos lineales tenemos varios, uno por cada clase o categoría:

$$o_1 = x_1w_{11} + x_2w_{12} + x_3w_{13} + x_4w_{14} + b_1,$$

$$o_2 = x_1w_{21} + x_2w_{22} + x_3w_{23} + x_4w_{24} + b_2,$$

$$o_3 = x_1w_{31} + x_2w_{32} + x_3w_{33} + x_4w_{34} + b_3.$$

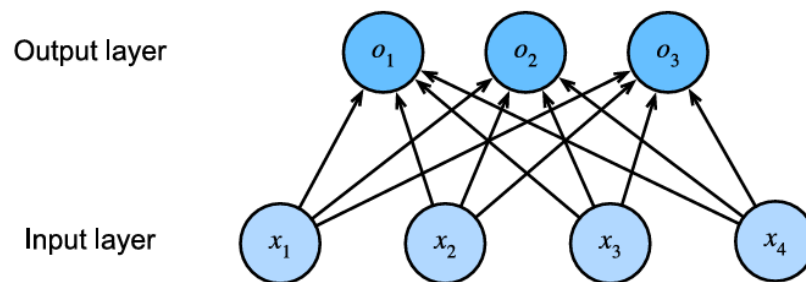
El valor de cada uno de estos términos se denomina logit y sus valores sirven como entrada a una función de activación.

Modelo

$$o_1 = x_1w_{11} + x_2w_{12} + x_3w_{13} + x_4w_{14} + b_1,$$

$$o_2 = x_1w_{21} + x_2w_{22} + x_3w_{23} + x_4w_{24} + b_2,$$

$$o_3 = x_1w_{31} + x_2w_{32} + x_3w_{33} + x_4w_{34} + b_3.$$



Función Softmax

Es la función de activación para este tipo de clasificación. Puesto que tenemos k-salidas distintas y queremos que las sumas de las probabilidades estén definidas en el rango $[0, 1]$ y que además la suma sea igual a la unidad, definimos la función softmax como:

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Donde:

- K , es el número de clases.
- $z_k = w_k^T x + b_k$, es el logit para la salida K .
- \hat{y}_k es la probabilidad calculada de que la entrada pertenezca a la clase K .

Función de verosimilitud

Recordando que:

$$L(\mathbf{w}, b) = \prod_{i=1}^m P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b)$$

Para la clasificación multi-clase tenemos que:

$$P(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}, b) = \prod_{k=1}^K (\hat{y}_k^{(i)})^{y_k^{(i)}}$$

Donde:

- $y_k^{(i)}$ es un indicador binario (1 si la clase k es la verdadera clase para el i -ésimo ejemplo, 0 en caso contrario).
- $\hat{y}_k^{(i)}$ es la probabilidad predicha para la clase k en el i -ésimo ejemplo.

Función de log-verosimilitud y función de pérdida

Entonces tenemos que:

$$\log L(\mathbf{w}, b) = \sum_{i=1}^M \sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$$

La función suma sobre todos los ejemplos de entrenamiento y todas las clases, considerando la probabilidad logarítmica de la clase correcta para cada ejemplo (debido al término $y_k^{(i)}$). Entonces la función de pérdida es:

$$\text{Loss}(\mathbf{w}, b) = - \sum_{i=1}^M \sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$$

Entropía cruzada categórica

Esta función de pérdida se denomina **entropía cruzada categórica** (categorical cross-entropy):

$$\text{Loss}(\mathbf{w}, b) = - \sum_{i=1}^M \sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$$

- m indica el número de ejemplos de entrenamiento.
- El término interno $\sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$ obtiene la probabilidad logarítmica correspondiente a la clase verdadera para cada ejemplo.
- La suma externa $\sum_{i=1}^M$ suma esta cantidad sobre todos los ejemplos de entrenamiento.
- La función indica que tanto las probabilidades predichas por el modelo se ajustan a las etiquetas de los datos.

Ejemplo sencillo:

$$\text{Loss}(\mathbf{w}, b) = - \sum_{i=1}^M \sum_{k=1}^K y_k^{(i)} \log(\hat{y}_k^{(i)})$$

Si tenemos:

$$Y = [0.4, 0.4, 0.2]$$

$$T = [0, 1, 0] \quad \rightarrow \quad L = -0 \cdot \ln(0.4) - 1 \cdot \ln(0.4) - 0 \cdot \ln(0.2) = 0.92$$

$$Y = [0.1, 0.2, 0.7]$$

$$T = [0, 0, 1] \quad \rightarrow \quad L = -0 \cdot \ln(0.1) - 0 \cdot \ln(0.2) - 1 \cdot \ln(0.7) = 0.36$$

La segunda tiene menor error asociado al ofrecer una mejor predicción.

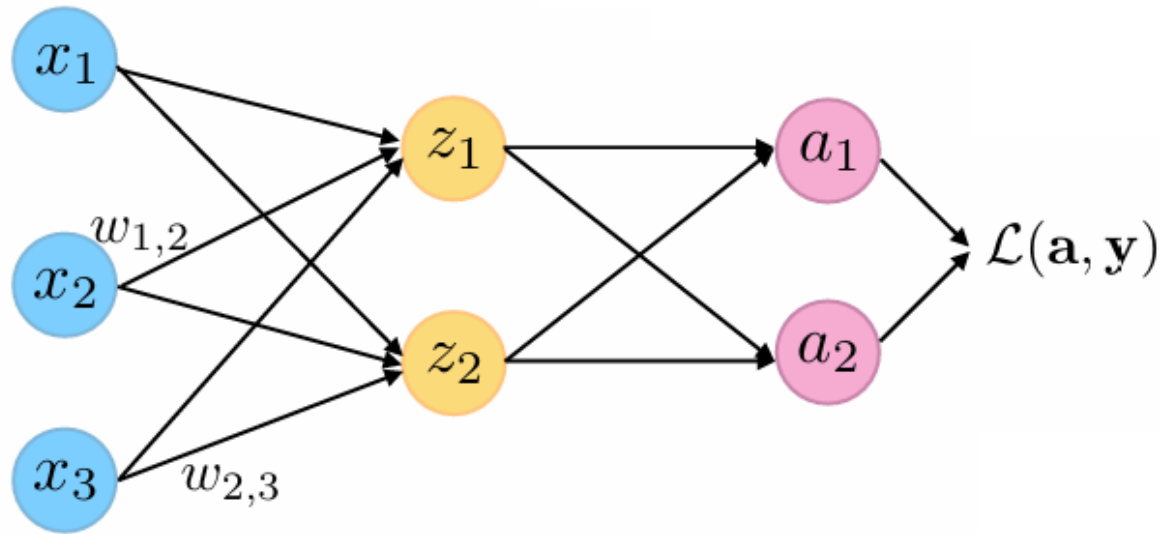
Cálculo del gradiente

Igual que antes, debemos calcular las derivadas utilizando la regla de la cadena.

Puesto que nuestras salidas se ven influenciadas por todos los términos (debido a la función softmax) se debe retrotraer la influencia de cada parámetro en particular.

Mas adelante veremos que esto se llama **retropropagación** o *backpropagation* y es un concepto clave en redes neuronales.

Cálculo del gradiente



$$\begin{aligned}\frac{\partial L}{\partial w_{1,2}} &= \frac{\partial L}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2} \frac{\partial a_2}{\partial z_1} \frac{\partial z_1}{\partial w_{1,2}} \\ &= \frac{-y_1}{a_1} [a_1(1 - a_1)]x_2 + \frac{-y_2}{a_2} (-a_2 a_1)x_2 \\ &= (y_2 a_1 - y_1 + y_1 a_1)x_2 \\ &= (a_1(y_1 + y_2) - y_1)x_2 \\ &= -(y_1 - a_1)x_2\end{aligned}$$

Actualización de parámetros

Una vez calculados los gradientes se actualizan los valores de los parámetros teniendo en cuenta la posición de cada uno en el modelo:

$$w_{j,k}^{(i+1)} = w_{j,k}^{(i)} - \eta \frac{\partial L}{\partial w_{j,k}}$$

$$b_k^{(i+1)} = b_k^{(i)} - \eta \frac{\partial L}{\partial b_k}$$

Donde η es el learning rate.

Cuadro comparativo

	Regresión lineal	Clasificación binaria	Clasificación multi-clase
Función de error	Error cuadrático medio	Entropía cruzada binaria	Entropía cruzada categórica
Función de activación	--	Sigmoide	Softmax
Número de salidas del modelo	1	1	K (número de clases)
Modelo	Lineal	Lineal + función de activación	K factores lineales + función de activación