Regresión Logística Multinomial

La regresión logística multinomial (RLM) (o regresión softmax) es una generalización de la regresión logística, en otras palabras la variable respuesta puede toma valores de entre múltiples clases:

$$y_i \in \{1, \dots, K\} \quad \text{donde } K \text{ es el } \#\text{clases}$$

en otras palabras, la RLM es un modelo para problemas que requieren clasificación de entre múltiples clases ($K > 2$).

Definición del modelo

Al igual que la regresión logística, este modelo consta de tres componentes:

1. Componente aleatorio

Formado por una **variable aleatoria categórica** y, tal que:

$$y_i \in \{1, \dots, K\} \quad i = 1, \dots, n$$

y un **vector de probabilidad** $\pi_i = (\pi_{i1}, \dots, \pi_{iK})$ asociado a y_i, tal que,

$$\pi_{i1} + \ldots + \pi_{iK} = 1$$
$$\sum_{k=1}^{K} \pi_{ik} = 1$$

- π_{ik} es la probabilidad de que la *i-ésima* observación pertenezca a la clase k.
- π_i es un vector de K elementos.

En otras palabras, cada observación x_i tiene asociado un vector de probabilidad π_i, tal que la suma de sus elementos es 1.

> *El valor de y_i es la clase con mayor probabilidad del vector π_i,*

2. Componente Sistémico

Consiste de *transformaciones lineales* (o conocido también como combinaciones lineales)

$$z_{ik} = \theta_{0k}1 + \theta_{1k}x_{i1} + \ldots + \theta_{(p-1)k}x_{i(p-1)} \quad i = 1, \dots, n$$

expresado matricialmente:

$$z_{ik} = \boldsymbol{\theta}_k^T x_i \quad i = 1, \ldots, n; \quad k = 1, \ldots, K$$

- $\boldsymbol{\theta}_k$ es el vector de parámetros $(1 \times p)$ correspondiente a la clase k. Por tanto, habrá K vectores, uno por cada clase.
- x_i es el vector predictor correspondiente a la *i-ésima* observación, consiste de p características (o predictores), donde el primer elemento es 1.
- n el número de observaciones.
- K el número de clases.

3. Función enlace

Conecta el componente aleatorio y el componente sistémico a través de la función *Logit*.

Tomemos arbitrariamente una de las probabilidades como base, en este caso tomaremos la probabilidad de la primera clase π_{i1}, entonces:

$$Logit(\pi_{ik}) = \log\left(\frac{\pi_{ik}}{\pi_{i1}}\right) = z_{ik} \quad i = 1, \ldots, n$$

eliminando el logaritmo y despejando π_{ik}:

$$\pi_{ik} = \pi_{i1} e^{z_{ik}} \quad k = 2, \ldots, K$$

tomando en cuenta la anterior expresión y considerando que $\pi_{i1} + \ldots + \pi_{iK} = 1$, se puede llegar a las siguientes expresiones:

$$\pi_{i1} = \frac{1}{1 + \sum_{j=2}^{K} e^{z_{ij}}} \quad \text{(probabilidad de la 1ra clase)}$$

y para el resto de las clases:

$$\pi_{ik} = \frac{e^{z_{ik}}}{1 + \sum_{j=2}^{K} e^{z_{ij}}}, \quad k = 2, \ldots, K$$

> *Ver anexos, el desarrollo del cálculo de las anteriores expresiones (**Cálculo de** π_{i1} y π_{ik})*

Función Softmax

Realizando más cálculos se puede unificar π_{i1} y π_{ik}, obteniendo así una única expresión que calcule la probabilidad para cualquier clase:

$$\pi_{ik} = \frac{e^{z_{ik}}}{\sum_{j=1}^{K} e^{z_{ij}}}, \quad k = 1, \ldots, K$$

- El denominador básicamente es la suma de $e^{z_{i1}} + \ldots + e^{z_{ik}}$.

esta expresión corresponde a la denominada **Función Softmax**.

Entrenamiento

El objetivo es encontrar los mejores parámetros θ.

Estimador de máxima verosimilitud

Notemos que tomando el componente aleatorio y podemos obtener una función de verosimilitud.

$$\begin{aligned} f(\pi_1, \ldots, \pi_n) &= f(\pi_1) \cdot \ldots \cdot f(\pi_n) \\ &= \prod_{i=1}^{n} f(\pi_i) \end{aligned}$$

como π_i es un vector, entonces:

$$\begin{aligned} &= \prod_{i=1}^{n} f(\pi_{i1}, \pi_{i2}, \ldots, \pi_{iK}) \\ &= \prod_{i=1}^{n} f(\pi_{i1}) \cdot f(\pi_{i2}) \cdot f(\pi_{iK}) \\ &= \prod_{i=1}^{n} \prod_{k=1}^{K} f(\pi_{ik}) \end{aligned}$$

finalmente:

$$f(\pi_{11}, \ldots, \pi_{nK)} = \prod_{i=1}^{n} \prod_{k=1}^{K} \pi_{ik}^{y_{ik}}$$

> ***Nota**: $f(\pi_{ik}) = \pi_{ik}^{y_{ik}}$ es una función indicatriz, así que $y_{ik} \in \{0, 1\}$. Notar que cada y_i esta en codificación one-hot, por esta razón los y_{ik} solo tiene valores 0 o 1.*

aplicando logaritmo a $f(\pi_{11}, \ldots, \pi_{nK)}$:

$$\log(f) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log\left(\pi_{ik}\right)$$

escribiendo $\log(f) = l$ tenemos:

$$l(\pi_{11}, \ldots, \pi_{nK}) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log\left(\pi_{ik}\right)$$

recordar que π_{ik} es una *función softmax* que depende de z_{ik}, además $z_{ik} = \boldsymbol{\theta}_k^T x_i$, entonces la **función de verosimilitud** $l(\pi_{11}, \ldots, \pi_{nK})$ en realidad es una función que depende de los parámetros $\theta = (\theta_1^T, \ldots, \theta_K^T)$, por lo tanto podemos reescribir:

$$l(\pi_1, \ldots, \pi_n) = l(\theta)$$

De esta forma obtenemos un estimador de máxima verosimilitud para la matriz de parámetros θ:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\ l(\theta)$$

para definirlo en **términos de minimización**, reescribimos $J(\theta) = -l(\theta)$, obteniendo así una **función de costo**:

$$J(\theta) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log\left(\pi_{ik}\right)$$

- recordar que π_{ik} es la *función softmax* .

ahora el objetivo es encontrar los parámetros θ que **minimicen la función de costo**:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\ J(\theta)$$

Estimación por el método Gradiente Descendente

Mediante este método podemos minimizar la función de costo y encontrar los mejores parámetros θ.

1. Definir una *tasa de aprendizaje* α y un número de *epochs*.
2. Inicializar θ como una **matriz** $(p \times K)$ con valores aleatorios.
3. Iterar el número de *epochs* y actualizar los valores de θ en cada iteración:

    for epoc to *epochs* do

        $\theta := \theta - \alpha \nabla J(\theta)$

    end

Luego, el reto es calcular el gradiente de la función de costo $\nabla J(\theta)$.

Gradiente de la función de costo

$$\nabla J(\theta) = \frac{\partial J(\theta)}{\partial \theta}$$

realizado el cálculo y expresando en forma matricial, tenemos:

$$\nabla J(\theta) = -X^T(Y - \mathbf{\Pi})$$

- $\mathbf{\Pi}$ es una matriz $n \times K$ el cual contiene las probabilidades π_{ik}, (detalles de cómo calcular esta matriz, ver anexos).
- Y debe ser una matriz de $n \times K$, aplicar codificación *one-hot*.
- X debe contener 1's en la primera columna.

- Realizar una *normalización z-score* de los datos, excepto la columna de 1's

> *se omitió el desarrollo del cálculo del gradiente por cuestiones de simplicidad.*

Predicción

Dado un nuevo conjunto de datos X estimar las respuestas $\hat{y}$.

El proceso de clasificación consiste en calcular las K probabilidades para cada x_i :

$$\pi_{ik} = \frac{e^{\theta_k^T x_i}}{\sum_{j=1}^{K} e^{\theta_j^T x_i}}, \quad k = 1, \ldots, K$$

luego la clase con mayor probabilidad, será la clase elegida como respuesta de la *i-ésima* observación (x_i):

$$\hat{y}_i = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \ \pi_{ik}$$

En algunos casos será necesario definir un umbral para la probabilidad y así mejorar las predicciones.

Anexos

Cálculo de π_{i1} y π_{ik}

Tomemos en cuenta las siguientes expresiones:

$$\pi_{ik} = \pi_{i1} e^{z_{ik}} \quad k = 2, \ldots, K \qquad (1)$$
$$\pi_{i1} + \pi_{i2} + \ldots + \pi_{iK} = 1 \qquad (2)$$

reemplazando (1) en (2):

$$\pi_{i1} + \pi_{i1} e^{z_{i2}} + \ldots + \pi_{i1} e^{z_{iK}} = 1$$
$$\pi_{i1}(1 + e^{z_{i2}} + \ldots + e^{z_{iK}}) = 1$$
$$\pi_{i1}(1 + \sum_{j=2}^{K} e^{z_{ij}}) = 1$$
$$\pi_{i1} = \frac{1}{1 + \sum_{j=2}^{K} e^{z_{ij}}} \qquad (3)$$

reemplazando (3) en (1):

$$\pi_{ik} = \frac{1}{1 + \sum_{j=2}^{K} e^{z_{ij}}} e^{z_{ik}}$$

$$\pi_{ik} = \frac{e^{z_{ik}}}{1 + \sum_{j=2}^{K} e^{z_{ij}}} \quad k = 2, \ldots, K$$

obteniendo así las probabilidades π_{i1} y π_{ik} con $i = 1, \ldots, n$.

Cálculo de la matriz de probabilidades $\mathbf{\Pi}$

Si bien se puede calcular cada probabilidad con la función softmax definida arriba, también podríamos **calcularlo en forma matricial**:

$$\text{softmax}(x_i, \theta)_i = \frac{e^{x_i \theta}}{\sum_{j=1}^{K} e^{x_i \theta_j}}$$

- x_i es un vector $1 \times p$ correspondiente la observación i.
- θ_j es un vector $1 \times p$ correspondiente a los parámetros de una clases en específico.
- θ es una matriz $p \times K$ correspondiente a todos los parámetros de todas las clases.

El resultado de esta función es un vector $1 \times K$.

El paquete *numpy* de Python permite hacer operaciones aritméticas entre matrices a través de un mecanismo llamado *Broadcasting*, por lo tanto, es posible **calcular directamente la matriz de probabilidades $\mathbf{\Pi}$** definiendo una función softmax como:

$$softmax(X, \theta) = \frac{e^{X\theta}}{(\sum_{j=1}^{K} e^{x_1 \theta_j}, \ldots, \sum_{j=1}^{K} e^{x_n \theta_j})}$$

- X es una matriz $n \times p$, es decir, correspondiente a las n observaciones y p variables predictoras. Recordar que la primera columna debe contener 1's.
- θ es una matriz de parámetros $p \times K$.
- x_i es un vector $1 \times p$, que corresponde a la observación i.
- El denominador es un vector $n \times 1$.

El resultado es una matriz $n \times K$.

Referencias

- Softmax Regression http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/