SoftMax Regression

Introduction

In the previous chapter we introduce logistic regression that only worked for binary class dataset. But in the real world exist multiclass dataset there are doesn't applicable binary logistic regression. For this problem we introduce a linear regression that are solve this problem and work with multiclass dataset called SoftMax regression. SoftMax regression nothing but specialization of Logistic regression.

1.1 SoftMax Function

In logistic regression (chapter 5) we use a function that called sigmoid function. Sigmoid function calculates probability of each data point to satisfying one class. The SoftMax function calculates all probability for all classes of each data point.

1.1.1 Derivation of SoftMax Equation

Consider a classification problem which involved k number of classes, x feature vector and y is the corresponding class, where $y \in \{1,2,3,4,5, \dots . , k\}$

Now we would like a probability of y given x , $P(y|x)$ which is a vector of probabilities of y given features x.

$$P(y|x) = \begin{bmatrix} P(y=1|x) \\ P(y=2|x) \\ P(y=3|x) \\ \vdots \\ P(y=k|x) \end{bmatrix}$$

Recall that in logistic regression, log-odd for y=1 with respect to y=0 is assumed to have a linear relationship with the independent variable x.

Using the same analogy, we can assume that the log-odd for $y = i$ with respect to $y = k$ is assumed to have linear relationship with the independent variable x.

$$\ln \frac{P(y=i|x;w)}{P(y=k|x;w)} = xw_i$$

$$\Rightarrow \frac{P(y=i|x)}{P(y=k|x)} = e^{xw_i}$$

$$\Rightarrow P(y=i|x) = P(y=k|x)e^{xw_i}$$

Since, sum of $P(y=j|x)$ for $j = 1,2,3, \dots , k$ is 1

$$\sum_{j=1}^{k} P(y=j|k) = \sum_{j=1}^{k} P(y=k|x)e^{xw_j}$$

$$\Rightarrow 1 = P(y=k|x) \sum_{j=1}^{k} e^{xw_j}$$

$$\Rightarrow \ P(y=k|x) = \frac{1}{\sum_{j=1}^{k} e^{xw_j}}$$

By substitution, we get

$$P(y=i|x) = \frac{e^{xw_i}}{\sum_{j=1}^{k} e^{xw_j}}$$

$$\text{Where } w_i = \begin{bmatrix} \cdots & w_{0i} & \cdots \\ \cdots & w_{1i} & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & w_{mi} & \cdots \end{bmatrix}, m = number\ of\ features$$

Now we write it into following mathematical notation

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Where e^{z_i} = standard exponential function for output vector, e^{z_j} = standard exponential function of output vector, k = number of classes in the multiclass classifier, σ = softmax, m =number of features

1.2 Mathematical intuition of SoftMax Regression

Before proceed, let's gets introduced the indication function which output 1 if argument is true otherwise 0

$$1\{.\} = \begin{cases} 1\ if\ y = i\ is\ true \\ 0\ other\ wise \end{cases}$$

To derive the SoftMax regression model, we can start from the principle of maximum likelihood. Suppose we have a training set of N examples, where each example $x^{(i)} \in \mathbb{R}^M$ is labeled with one of K classes, $y^{(i)} \in \{1, 2, \ldots, K\}$.

To get the likelihood on the training data, we need to compute all of the probabilities of $y = y^{(i)}$ given $x^{(i)}$ for i=1, 2, 3, …, N. (N is the total number of training data)

$$P\left(y^{(i)}|x^{(i)};\beta\right) = P\left(y = y^{(i)}|x^{(i)};\beta\right) = \prod_{k=1}^{K} P\left(y^{(i)} = k|x^{(i)};\beta\right)^{1\{y^{(i)}=k\}}$$

To simplify the notation, we can write β as an $K \times M$ matrix, where each column β_k corresponds to parameters for class k. Then, we can define the linear model for the log-odds of class k as:

$$z_k^{(i)} = x^{(i)}\beta_k$$

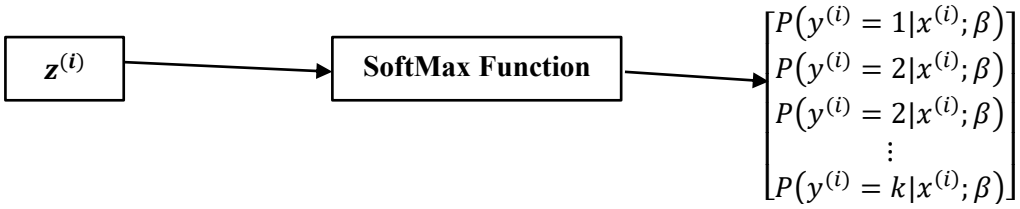
$$\Rightarrow z_k^{(i)} = [x_{i0} \quad x_{i1} \quad x_{i2} \quad \ldots \quad x_{im}] \begin{bmatrix} \cdots & \beta_{0k} & \cdots \\ \cdots & \beta_{1k} & \cdots \\ \cdots & \beta_{2k} & \cdots \\ & \vdots & \\ \cdots & \beta_{mk} & \cdots \end{bmatrix}$$

The SoftMax function then converts the log-odds to probabilities, by exponentiating and normalizing them:

$$O_k^{(i)} = h_\beta\left(x^{(i)}\right)_k = \frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}} = P\left(y^{(i)} = k|x^{(i)};\beta\right)$$

Find prediction's SoftMax value ith row

$$\Rightarrow z^{(i)} = \begin{bmatrix} z_1^{(i)} & z_2^{(i)} & z_2^{(i)} & \ldots & z_k^{(i)} \end{bmatrix} = [x_{i0} \quad x_{i1} \quad x_{i2} \quad \ldots \quad x_{im}] \begin{bmatrix} \beta_{01} & \beta_{02} & \beta_{03} & \cdots & \beta_{0k} \\ \beta_{11} & \beta_{12} & \beta_{13} & \cdots & \beta_{1k} \\ \beta_{21} & \beta_{22} & \beta_{23} & \cdots & \beta_{2k} \\ & & \vdots & & \\ \beta_{m1} & \beta_{m2} & \beta_{m3} & \cdots & \beta_{mk} \end{bmatrix}$$



We want to find the parameters β that maximize the likelihood of the data, given by:

$$L(\beta) = P(Y|X;\beta) = \prod_{i=1}^{N} P\left(y^{(i)}|x^{(i)};\beta\right)$$

$$L(\beta) = \prod_{i=1}^{N}\prod_{k=1}^{k}\left(O_k^{(i)}\right)^{1\{y^{(i)}=k\}}$$

To make optimization easier, we can take average cross entropy Loss function:

$$\Rightarrow J(\beta) = -\frac{1}{N}\ln[L(\beta)] = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\}\ln\left(O_k^{(i)}\right)$$

$$\Rightarrow J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\}\ln\left(\frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}\right)$$

$$\Rightarrow J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} 1\{y^{(i)} = k\}\ln\left(e^{z_k^{(i)}}\right) - \left[\sum_{k=1}^{K} 1\{y^{(i)} = k\}\right]\ln\left[\sum_{j=1}^{K} e^{z_j^{(i)}}\right]\right]$$

$$\Rightarrow J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} 1\{y^{(i)} = k\}z_k^{(i)}\ln(e) - \ln\left[\sum_{j=1}^{K} e^{z_j^{(i)}}\right]\right]$$

Here $\sum_{k=1}^{K} 1\{y^{(i)} = k\} = 1$, because only one condition will be true and others is false.

$$\Rightarrow J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=1}^{K} 1\{y^{(i)} = k\}x^{(i)}\beta_k - \ln\left[\sum_{j=1}^{K} e^{x^{(i)}\beta_k}\right]\right]$$

This is the cost function that we want to minimize with respect to β. To do so, we can use gradient descent or any other optimization algorithm. The gradient of the cost function with respect to β_k is given by:

$$\nabla_{\beta_k} J(\beta) = -\frac{\partial J(\beta)}{\partial \beta_k}$$

Here β_k is vector so applying vector differentiate,

$$\Rightarrow \nabla_{\beta_k} J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\left[1\{y^{(i)} = k\}x^{(i)T} - \frac{x^{(i)T}.e^{x^{(i)}\beta_k}}{\sum_{j=1}^{K} e^{x^{(i)}\beta_j}}\right]$$

$$\left[\frac{\partial(AX)}{\partial X} = A^T \text{and } \frac{\partial(e^{AX})}{\partial X} = (AX)'e^{AX} = A^T.e^{AX}, \text{here } A, X \text{ is matrix where } A \text{ is indepent of } X\right]$$

$$\Rightarrow \nabla_{\beta_k} J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}(1\{y^{(i)} = k\} - O_k^{(i)})x^{(i)T}$$

[See details of the derivation go to section 6.2.1]

To prevent overfitting, we can also add a regularization term to the cost function, such as the L2 norm of θ:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \ln\, O_k^{(i)} + \frac{\lambda}{2}\sum_{k=1}^{K}\sum_{j=1}^{M}\beta_{jk}^2$$

where λ is the regularization parameter that controls the trade-of between the data fit and the model complexity. The gradient of the regularization term is simply $\lambda\beta_k$, so the update rule for gradient descent becomes:

$$\beta_k := \beta_k - \alpha\left(-\frac{1}{N}\sum_{i=1}^{N}(y_k^{(i)} - O_k^{(i)})x^{(i)T} + \lambda\beta_k)\right)$$

$$\Rightarrow \beta_k := \beta_k + \frac{\alpha}{N}\left(\sum_{i=1}^{N}(y_k^{(i)} - O_k^{(i)})x^{(i)T} + \lambda\beta_k)\right)$$

1.2.1 Partial derivation calculation:

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N} E_i\,, where\ E_i = \sum_{k=1}^{K} 1\{y^{(i)} = k\}\ln\left(O_k^{(i)}\right)\ and\ O_k^{(i)} = \frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}$$

Calculating $\frac{\partial E_i}{\partial \beta_{mj}}$, $m = \{1,2,3,4,5, \dots \dots \dots, M\}$:

$$\frac{\partial E_i}{\partial \beta_{mj}} = \frac{\partial\left(\sum_{k=1}^{K} 1\{y^{(i)} = k\}\ln\left(O_k^{(i)}\right)\right)}{\partial \beta_{mj}} = \sum_{k=1}^{K}\frac{\partial}{\partial \beta_{mj}}\left(1\{y^{(i)} = k\}\ln\left(O_k^{(i)}\right)\right)$$

Now, $O_k^{(i)}$ are not direct function of $z_j^{(i)}$, we can apply chain rule and $1\{y^{(i)} = k\}$ is equivalent of $y_k^{(i)}$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = \sum_{k=1}^{K}\left[\frac{\partial\left[y_k^{(i)}\ln\left(O_k^{(i)}\right)\right]}{\partial O_k^{(i)}} \times \frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} \times \frac{\partial z_j^{(i)}}{\partial \beta_{jm}}\right]$$

Focus on $\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}}$ term or Derivatives of SoftMax function

$$\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} = \frac{\partial}{\partial z_j^{(i)}}\left[\frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}\right]$$

$$\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} = \frac{\partial}{\partial z_j^{(i)}}\left[\frac{\left(\sum_{j=1}^{K} e^{z_j^{(i)}}\right)\frac{\partial\left(e^{z_k^{(i)}}\right)}{\partial z_j^{(i)}} - e^{z_k^{(i)}}\frac{\partial\left(\sum_{j=1}^{K} e^{z_j^{(i)}}\right)}{\partial z_j^{(i)}}}{\left[\sum_{j=1}^{K} e^{z_j^{(i)}}\right]^2}\right]$$

Case-I: If j==k,

$$\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} = \frac{\partial}{\partial z_j^{(i)}}\left[\frac{e^{z_k^{(i)}}\sum_{j=1}^{K} e^{z_j^{(i)}} - e^{z_k^{(i)}}.e^{z_j^{(i)}}}{\left[\sum_{j=1}^{K} e^{z_j^{(i)}}\right]^2}\right] = O_k^{(i)}\left(1 - O_j^{(i)}\right)$$

Case-II if $j \neq k$,

$$\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} = \frac{\partial}{\partial z_j^{(i)}}\left[\frac{0 - e^{z_k^{(i)}}.e^{z_j^{(i)}}}{\left[\sum_{j=1}^{K} e^{z_j^{(i)}}\right]^2}\right] = -O_k^{(i)} \times O_j^{(i)}$$

Summery above all:

$\frac{\partial O_k^{(i)}}{\partial z_j^{(i)}} = \begin{cases} O_k^{(i)}\left(1 - O_j^{(i)}\right) if\ j = k \\ -O_k^{(i)} \times O_j^{(i)}\ if\ j \neq k \end{cases} = O_k^{(i)}\left(\delta_{kj} - O_j^{(i)}\right)$, where δ_{kj} is the Kronecker delta that equals 1 if $k = j$, and 0 otherwise.

Focus on $\frac{\partial\left[y_k^{(i)} \ln\left(O_k^{(i)}\right)\right]}{\partial O_k^{(i)}}$ term,

$$\frac{\partial\left[y_k^{(i)} \ln\left(O_k^{(i)}\right)\right]}{\partial O_k^{(i)}} = \frac{y_k^{(i)}}{O_k^{(i)}}$$

And

$$\frac{\partial z_j^{(i)}}{\partial \beta_{mj}} = \frac{\partial\left(\beta^{(j)T} x^{(i)}\right)}{\partial \beta_{mj}} = x_m^{(i)}$$

Substituting all, then we get

$$\frac{\partial E_i}{\partial \beta_{mj}} = \sum_{k=1}^{K}\left[O_k^{(i)}\left(\delta_{kj} - O_j^{(i)}\right) \times \frac{y_k^{(i)}}{O_k^{(i)}} \times x_m^{(i)}\right]$$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = \sum_{k=1}^{K}\left[\left(\delta_{kj} - O_j^{(i)}\right) \times y_k^{(i)} \times x_m^{(i)}\right]$$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = x_m^{(i)} \sum_{k=1}^{K}\left[\left(\delta_{kj} - O_j^{(i)}\right) \times y_k^{(i)}\right]$$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = x_m^{(i)}\left[\sum_{k=1}^{K} y_k^{(i)}\delta_{kj} - \sum_{k=1}^{K} O_j^{(i)} y_k^{(i)}\right]$$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = x_m^{(i)}\left[\sum_{k=1}^{K} y_k^{(i)}\delta_{kj} - O_j^{(i)}\sum_{k=1}^{K} y_k^{(i)}\right]$$

$$\Rightarrow \frac{\partial E_i}{\partial \beta_{mj}} = x_m^{(i)} \left[y_k^{(i)} - O_j^{(i)} \right]$$

Since $\sum_{k=1}^{K} y_k^{(i)} = 1$ and $\sum_{k=1}^{K} y_k^{(i)} \delta_{kj} = y_k^{(i)}$

Now, we find matrix for $\frac{\partial E_i}{\partial \beta_k}$:

$$\frac{\partial E_i}{\partial \beta_k} = \begin{bmatrix} \frac{\partial E_i}{\partial \beta_{0k}} \\ \frac{\partial E_i}{\partial \beta_{1k}} \\ \frac{\partial E_i}{\partial \beta_{2k}} \\ \vdots \\ \frac{\partial E_i}{\partial \beta_{mk}} \end{bmatrix} = \begin{bmatrix} x_0^{(i)} \left(y_k^{(i)} - O_j^{(i)} \right) \\ x_1^{(i)} \left(y_k^{(i)} - O_j^{(i)} \right) \\ x_2^{(i)} \left(y_k^{(i)} - O_j^{(i)} \right) \\ \vdots \\ x_m^{(i)} \left(y_k^{(i)} - O_j^{(i)} \right) \end{bmatrix} = \left(y_k^{(i)} - O_j^{(i)} \right) \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_m^{(i)} \end{bmatrix} = \left(y_k^{(i)} - O_j^{(i)} \right) x^{(i)T}$$

Now, we calculate derivation of $J(\beta)$ w.r.t $\beta^{(k)}$:

$$\frac{\partial J(\beta)}{\partial \beta_k} = -\frac{1}{N} \sum_{i=1}^{N} \left(y_k^{(i)} - O_j^{(i)} \right) x^{(i)T}$$

Gradient descent update rule,

$$\beta_k := \beta_k + \frac{\alpha}{N} \sum_{i=1}^{N} \left(y_k^{(i)} - O_j^{(i)} \right) x^{(i)T}$$

$\alpha =$ learning rate

Note: I collecting every information from different books and website. I just try to explain easily with details.

MD.ROKON MIA

Student, CSE, Begum Rokeya University, Rangpur