Softmax Regression

Lei Zhao

I. The intuitive Idea of Softmax Regression

We need to give the output data meanings in order to build up the measurements of the model. If there are K categories, there will be K output numbers in the output layer, each number will be used to represent one category. The questions is how to represent one category? The intuitive idea is try to regard all these output numbers as scores and for a given data sample, the category with highest score is the class the model has predicted for the given data sample. The original goal we want to achieve is to pick up the index of the largest output number:

$$j^* = \underset{1 \le j \le k}{\text{argmax}} \{o_1, o_2, \cdots, o_K\} \tag{1}$$

Softmax is based on the idea that

$$j^* = \underset{1 \le j \le k}{\text{argmax}} \{o_1, o_2, \cdots, o_K\} = \underset{1 \le j \le k}{\text{argmax}} \{e^{o_1}, e^{o_2}, \cdots, e^{o_K}\} \tag{2}$$

We do not directly use the output number o_j, instead, we use e^{o_j}, they are equivalent for our goal to pick out the index of the largest number. What's more, there is also a benefit using this new representation, the logarithm of the sum of all e^{o_j} will approximate to the largest number of all o_j:

$$\log(\sum_{j=1}^{K} e^{o_j}) \approx \max_{1 \le j \le k} \{o_1, o_2, \cdots, o_K\} \tag{3}$$

That is also the reason we call it the **Softmax** and we will see this part is the core part of Softmax Regression.

The Softmax function helps us to formulate all the output numbers to become non-negative and sum to 1, thus, we can interpret the results of Softmax function as the probabilities that the input data sample belongs to the corresponding category. We use $\hat{y}_i$ to represent the probability that the given data sample

belongs to category i, i.e., $P(\boldsymbol{y_n}|o_i)$

$$\hat{y}_i = P(\boldsymbol{y}|o_i) = \text{softmax}(o_i) = \frac{e^{o_i}}{\sum_{j=1}^{K} e^{o_j}} \tag{4}$$

where $\boldsymbol{y}$ is the one-hot vector to represent the true category that the given data sample belongs to. And we use vector $\hat{\boldsymbol{y}}$ to represent the estimated probability results for the given data sample where $\hat{y}_i$ is its i-th component. Then, we can say

$$\hat{\boldsymbol{y}} = P(\boldsymbol{y}|\boldsymbol{o}) = \text{softmax}(\boldsymbol{o}) = \begin{bmatrix} \frac{e^{o_1}}{\sum_{j=1}^{K} e^{o_j}} \\ \vdots \\ \frac{e^{o_K}}{\sum_{j=1}^{K} e^{o_j}} \end{bmatrix} \tag{5}$$

$\boldsymbol{y}$ is the ground truth, we use $\hat{\boldsymbol{y}}$ to represent the predicted probability from the model. Then, how to evaluate this prediction? We need a loss function to measure the quality of our predicted probabilities and one most popular loss function is **Cross-Entropy Loss**:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{j=1}^{K} y_j \log(\hat{y}_j) \tag{6}$$

where $\boldsymbol{y} \in R^{K\times 1}$ is the ground truth one-hot vector to represent the true class that the given data belongs to, and $\hat{\boldsymbol{y}} \in R^{K\times 1}$ is the estimated probability vector. The cross-entropy loss can be interpreted as the expected value of the loss for a distribution over labels $\boldsymbol{y}$.

How to interpret this loss function?

II. Likelihood Maximization

We know the result from softmax function $\hat{y}_i$ can be interpreted as estimated conditional probabilities of each class given the input o_i to the softmax function. We compare the estimated $\hat{\boldsymbol{y}}$ with the reality $\boldsymbol{y}$ by checking how probable $\boldsymbol{y}$ is, i.e., $\hat{\boldsymbol{y}} = P(\boldsymbol{y}|\boldsymbol{o})$. **Maximizing the estimated probability of the true reality by our model is called the likelihood maximization.** And maximizing this conditional probability is equivalent to minimize the negative logarithm of this conditional probability.

$$\text{maximize } P(\boldsymbol{y}|\boldsymbol{o}) \rightarrow \text{minimize } -\log\left(P(\boldsymbol{y}|\boldsymbol{o})\right) \tag{7}$$

This formulation is purely from the idea that try to maximize the estimated probability of the ground truth. Then we connect this idea with the cross-entropy loss function by the ground truth one-hot vector

$\boldsymbol{y}$:

$$-\log(P(\boldsymbol{y}|\boldsymbol{o})) = -\sum_{j=1}^{K} y_j \log(P(\boldsymbol{y}|o_j)) = -\sum_{j=1}^{K} y_j \log(\hat{y}_j) \quad \text{: cross-enropy loss} \tag{8}$$

this expectation concept can be interpreted as picking up the negative logarithm of the ground truth probability $-\log(\hat{y}_{j^*})$ where only j^*-th component of the one-hot vector $\boldsymbol{y}$ equals to 1, the rest part of $\boldsymbol{y}$ are 0. The loss based on softmax function becomes to

$$L(\boldsymbol{y}, \boldsymbol{o}) = -\sum_{j=1}^{K} y_j \log\left(\frac{e^{o_j}}{\sum_{i=1}^{K} e^{o_i}}\right) = \log(\sum_{i=1}^{K} e^{o_i}) - \sum_{j=1}^{K} y_j o_j \tag{9}$$

And we know $\log(\sum_{i=1}^{K} e^{o_i})$ approximate to the largest number among $\{o_1, o_2, \cdots, o_K\}$, the negative logarithm of the softmax function just try to compare the difference between the approximated largest output number of the neural network with the output number n_{j^*} which is corresponding to the ground truth j^* when $\boldsymbol{y}$ is the one-hot vector:

$$L(\boldsymbol{y}, \boldsymbol{o}) = \log(\sum_{i=1}^{K} e^{o_i}) - o_{j^*} \quad \text{only for understanding} \tag{10}$$

When we compute the gradient of the cross-entropy loss function based on the softmax on the output of the neural network $\boldsymbol{o}$, we can understand why the loss function need to be designed like this.

$$\nabla_{\boldsymbol{o}} L = \begin{bmatrix} \partial_{o_1} L \\ \vdots \\ \partial_{o_K} L \end{bmatrix} = \begin{bmatrix} \frac{e^{o_1}}{\sum_{i=1}^{K} e^{o_i}} - y_1 \\ \vdots \\ \frac{e^{o_K}}{\sum_{i=1}^{K} e^{o_i}} - y_K \end{bmatrix} = \begin{bmatrix} \text{softmax}(o_1) - y_1 \\ \vdots \\ \text{softmax}(o_1) - y_K \end{bmatrix} = \begin{bmatrix} P(y_1 = 1|\boldsymbol{o}) - y_1 \\ \vdots \\ P(y_K = 1|\boldsymbol{o}) - y_K \end{bmatrix} \tag{11}$$