Variant loss function to fix sample-imbalanced problems

Linzhi Wu

July 13, 2019

0.1 Introduce

The sample-imbalanced problem mainly exists in supervised machine learning tasks. When encountering unbalanced dataset, the traditional classification algorithm with the overall classification accuracy as the learning goal will pay more attention to the majority class, thus making the classification performance of the minority class samples worse. Most common machine learning algorithms cannot work well for unbalanced datasets. In this report, We will specially discuss how to solve the sample-imbalanced problem effectively by applying the variant of traditional *cross entropy loss function* in deep learning field.

0.2 Solutions

Since traditional N-class cross entropy loss function

$$Loss_{ce} = -\sum_{k}^{N} y_k \log \hat{y}_k \tag{1}$$

where y_k is the label value of category k, $\hat{y}_k$ is the output of neural network and *log* is natural logarithm. Unfortunately, equation (1) is usually limited when the samples of different categories are balanced. Therefore, we can consider improving it to fix the problem below.

we will discuss two and multiple text classification, respectively. As for two classification(N=2), we append coefficient $\lambda_{y,\hat{y}}$ on the basis of equation (1).

$$Loss_{new\_ce} = -\sum_{k}^{N} \lambda_{y,\hat{y}} y_k \log \hat{y}_k \tag{2}$$

where $\lambda_{y,\hat{y}} = 1 - \epsilon(y_k - m)\epsilon(\hat{y}_k - m) - \epsilon(1 - m - y_k)\epsilon(1 - m - \hat{y}_k)$. m is a positive constant, which ranges from (0.5, 1), and

$$\epsilon(x) = \begin{cases} 0 & x > 0 \\ 0.5 & x = 0 \\ 1 & x < 0 \end{cases} \tag{3}$$

is step function.

Apparently, we have $\lambda_{y,\hat{y}} = 1 - \epsilon(\hat{y}_k - m)$ for positive samples ($y_k = 1$). Thus, if the prediction of network $\hat{y}_k$ is greater than m, then $\lambda_{y,\hat{y}} = 0$ while $\lambda_{y,\hat{y}} = 1$ for $\hat{y}_k \leq m$. So, we have

$$Loss_{new\_ce} = \begin{cases} 0 & \hat{y}_k > m \\ -\sum_{k}^{N} 0.5 y_k \log \hat{y}_k & \hat{y}_k = m \\ -\sum_{k}^{N} y_k \log \hat{y}_k & \hat{y}_k < m \end{cases} \tag{4}$$

Equation (4) denotes only when $\hat{y}_k \leq m$, the loss value is non-zero value and the weight of network will be normally updated. Otherwise, the weight will not be updated beacause of zero loss.

In the same way, for negative samples($y_k = 0$), we have $\lambda_{y,\hat{y}=1-\epsilon(1-m-\hat{y}_k)}$. then,

$$Loss_{new\_ce} = \begin{cases} -\sum_k^N y_k \log \hat{y}_k & \hat{y}_k > 1-m \\ -\sum_k^N 0.5 y_k \log \hat{y}_k & \hat{y}_k = 1-m \\ 0 & \hat{y}_k < 1-m \end{cases} \tag{5}$$

Equation (5) denotes only when $\hat{y}_k \geq 1-m$, the loss value is non-zero value and the weight of network will be updated normally. Otherwise, the weight will not be updated.

In summary, only when the prediction ranges in [1-m, m], the weight of network will be updated, and we call the interval "wrong-easily area" (As Figure 1 shows).

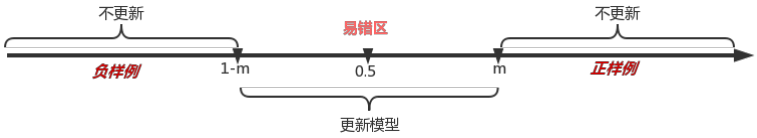


Figure 1: the rules of weight updating

Further, for multi-classification($N \geq 3$), we have learnt from the *focal loss* proposed by Kaiming He in one paper about dense object detection. We adaptively apply his appoach and ideas to text classification of NLP. Just like equation (2), we can also append constraint factor before equation (1). Hence, we have equation (6) as follow:

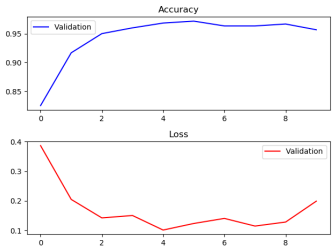
$$Loss_{fl} = -\sum_k^N (1-\hat{y}_k)^\gamma y_k \log \hat{y}_k \tag{6}$$

where γ is non-negative constant, and obviously $(1-\hat{y}_k)^\gamma$ is monotone reduction function. Intuitively, if the real result belongs to positive sample($y_k = 1$), but the network gives a negative one, then coefficient $(1-\hat{y}_k)^\gamma$ and loss value will become greater, which means increasing the punishment beacause of prediction errors. On the contrary, $(1-\hat{y}_k)^\gamma$ is very small and the weight will hardly update.

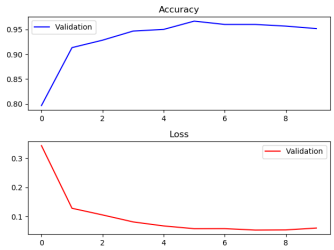
When the training set samples are unbalanced, the category with a large number of samples is easier to classify, while the category with a smaller number of samples is more difficult. The variant loss functions above can reduce the weight of the easily classified samples, so that the loss of the minority of prediction errors is greater than that of the majority, and the model will focus more on the samples that are easier to make mistakes during training, thus improving the performance of the overall prediction.

In order to prove our idea is correct, we conduct a comparative experiment on 3000 e-commerce text remarks, including 2000 positive ones and 1000 negative ones.

As shown in the pictures 2(a) and 2(b), we can clearly draw a conclusion: there is almost no difference in the accuracy of the model prediction between two loss function. However, after using the traditional one, the loss value is extremely unstable and there exists a reverse growth trend (easy to over-fitting)



(a) Traditional cross entropy loss function.



(b) Variant cross entropy loss function.

at about seventh epoch of training. Instead using the improved one, the loss value decreases steadily, which enables the model more robust and applicable.

Welcome to correct !!!