handling learnability imbalance in multiclass-classification

Theodor Peifer
email: thp7219@thi.de
Technische Hochschule Ingolstadt

***Abstract*—Neural Networks have proven themselves to be powerful classification tools by solving problems in a range of domains with high accuracy. Yet this accuracy is never evenly distributed across all classes, which means that the true-positive rates of each class separately are different. This can happen even in balanced datasets since some classes are more difficult to learn by the model than others (this phenomenon is further referred to as *learnability-imbalance*). A common way to address this problem is to give a weight to the error function for each class to penalize losses of certain classes higher or lower. This research will address the determination of such weights to counteract the learnability-imbalance in balanced datasets using previously calculated evaluation scores. Therefore the goal is to find methods to lower the variance of the true positive rates of each class.**

I. Introduction

A frequent problem in classification appears when working with datasets that have an unequal amount of samples per class and are therefore called a *class imbalanced* datasets. Since there will be some classes, that have less elements for the model to learn from, their features will be harder to extract what finally will result in a lower true positive rate, i.e. a per-class accuracy [1]. Thus, the consequence of having different class sizes can be described as having a *learnability imbalance* in the dataset since some classes are more difficult to learn that others.

But such learnability differences can appear also in balanced datasets for a variety of reasons, e.g. when the quality of the data of a class is lower than the rest of the data. A second reason, that this research will be focusing on, is that when some classes are similiar, the model can confuse their samples with each other more easily which will often result in a lower accuracy of those classes.

Even though this issue is an inevitable product of every normal classification, in most cases the learnability difference of the classes is either low or not from great interest. But there can be more extreme cases in which a model needs to produce fair and unbiased results using a dataset that has an obvious learnability imbalance. An example for that is *name-ethnicity classification*, where a model predicts the ethnicity of a name only by its letters [5]. Nationalities that use the same language and therefore have similiar names (e.g. *british* and *american*) result in a lower accuracy (see figure 1). Thus, when a model that is trained on such a dataset is used, e.g. for social science experiments, it can lead to unfair results and wrong interpretations.

Another dataset that is a good showcase for the learnability imbalance is the CIFAR-10 [2] dataset (32×32 RGB images of ten different classes) because it also contains similiar classes such as *dog* and *cat*. When looking at the true positive rates produced by a convolutional neural network [3], trained on that dataset, it can be seen that there are some classes that got misclassified more often (in this case bird, cat, deer and dog) which hints to a more difficult learnability. The CIFAR-10 and the name-nationality dataset will be used for experiments in order to find methods for minimizing such variances in the evaluation scores.

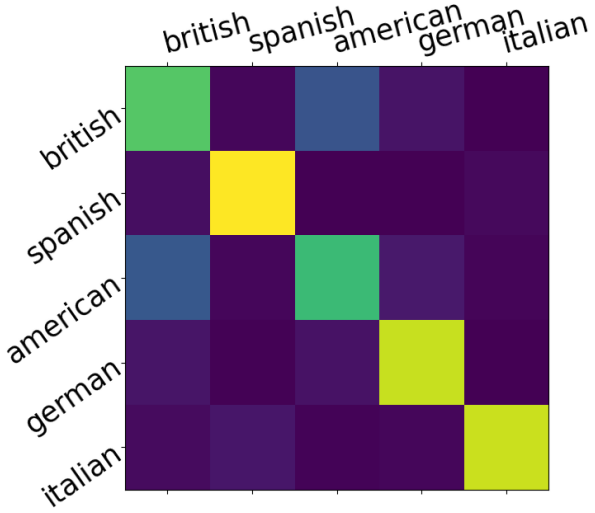

Fig. 1. A confusion matrix representing the true positive distribution of the name-nationality dataset produced by a recurrent neural network and the confusion of british and american names

To figure out such methods one should first go back to the class imbalance problem because there are already solutions existing. One of which is to weight the error function [1] according to the size of each class, i.e. the number of samples it contains. Therefore, for every class there is a weight, which is greater the fewer elements it contains and that gets multiplied with the loss produced by its samples. That will cause, that samples from a smaller class will produce a higher error and, since the aim of a neural network is to minimize the loss [4], will have a higher impact on the learning process.

Other methods will be discussed when reviewing exisiting literature on this topic. But since all of those approaches rely on the different proportions of the classes in the dataset it raises the question how to apply them on datasets that,

even though they don't suffer from class imbalance, still show a big learnability imbalance. When working with such datasets the learnability differences of the individual classes are mostly only identifiable after the model has been trained normally und unweighted. The calculation of the true positive rates and visualization (e.g. confusion matrix) can then reveal what classes were learned the best and which should have received a higher weight. These evaluation scores can then be used to determine which loss weights or how much data augmentation should be used for each class (an example is shown in *algorithm 1*).

In summary, this research will focus on creating and testing methods which use the true positive rate of each class in order to find out how much a class should be weighted or augmentated in order to counteract the learnability imbalance of the dataset.

The aim is to bring all per-class accuracies to the same value by increasing the true positive rate of classes that are harder to learn and decrease the rate of classes that more easily to learn. Further, it will address potential limitations and risks such as the reduction of the overall accuracy that could potentially happen from this process.

Algorithm 1 creating loss weights for a balanced dataset

C $\hat{=}$ *amount of classes*
train(weights) *describes the initialization and training process of a classification model in which* $weights_i$ *will be multiplied to every loss generated by a sample of class* i*.*
evaluate() *creates the set* s *with* $|s| = C$ *and* $s_i \in [0;1]$ *which contains the true positive scores of all classes of the test dataset.*
W(s) *is a function that creates a set of loss-weights* w *with* $|w| = C$ *and* $w_i \in \mathbb{R}^+$ *using a set of true positive scores* s*:* $W : [0;1] \to \mathbb{R}^+$ *;* $\{s_i, ..., s_C\} \to \{w_i, ..., w_C\}$
process:

1: $train(weights{=}\{w_1{=}1, w_2{=}1, ..., w_C{=}1\})$
2: $s = evaluate()$
3: $w = W(s)$
4: $train(weights{=}w)$
5: $s' = evaluate()$
6: $compare(s, s')$

II. LITERATURE REVIEW

As mentioned before, the learnability imbalance has to be handled the same way as the class imbalanced dataset problem, but by using the true positive rates instead of the class proportions of the dataset. Therefore, the proposed solutions to class imbalance are fundamental to this research and will be reviewed in the following (the methods in summary: loss weighting, augmentation and over-/ undersampling).

Since it was used as a main example in the introducion, the first method to examine is loss weighting [N]. This approach has proven itself to be a good way for archieving a lower variance in the per-class accuracies of unbalanced datasets and will be adapted to this research. In general, the weights are chosen to be inversely proportional to the amount of samples in the classes [Na], for example by using the function shown in forumla 1:

$$w_c = \frac{1 - \beta^{N_c}}{1 - \beta}$$

This method gets described as the *effective number of samples* weight [N], where N_c is the amount of samples of class c and β is a tuneable hyper-parameter with $\beta \in [0; 1[$.

Formula 2 shows how to use such weights along with the cross entropy loss function [N] for one sample:

$$\mathrm{CE}(x, c) = w_c \cdot (-\log(\hat{y}_c)))$$

$\hat{y}$ is the output of a model $\mathrm{F}(x, \theta)$ with input sample x and learnable parameter θ. It be described as a probability distribution for which $\hat{y}_i$ is the confidence of the model, that the input x corresponds to the i-th class. Therefore $\hat{y_c}$ represents the probability that the model correctly classified x as the wanted class c. As shown, the weight w_c that corresponds to the correct class gets multiplied with the normal error value (in the case of cross entropy: the negative $\log$ of the probablity).

Another loss weighting method is the so called *focal loss* [N], which is especially interesting for this research because it uses the true positive rates instead of the sizes of the classes to generate the weights. But since it is also mainly tested and practiced on class imbalanced datasets, this research will investigate its effectiveness on class balanced datasets that have high learnability variances. The weight for the focal loss is defined as the following (γ is a positive hyperparameter for scaling):

$$w_c = (1 - \hat{y}_c)^\gamma$$

When multiplying this weight with the loss, it will be bigger, the smaller the confidence of the correct class was. The adavantage of the focal loss over the proposed weight generation of this research is that it does not require a pre-training in order to figure out which classes a harder to learn, since it does this during the training process. But it can be hypothized that this weighting could have less impact on learnability imbalanced datasets because it figures out the imbalance while training. In contrast to this, when training unweighted first, calculating the weights afterwards and then train again with those weights, the loss function can start penalizing the classes differently right from the beginning.

Another method is *augmentation* [N], in which the input samples get randomly modified in order to synthetically generate more training samples and therefore help the model to generalize better. For example, in image classification it is common to flip, mirror or to add white noise to the images. The idea, originally proposed as *SMOTE* (Synthetic Minority Over-sampling Technique) [N], is that such transformations can be used on smaller classes to provide the model with more of their samples. The same technique can be used on learnability imbalanced datasets but by orienting on the per-class accuracies instead of the class sizes.

Another way to try to compensate for class imbalance is by doing *oversampling* [N] where the model gets fed samples of the minority classes more often.

Fig. 2. An example of augmentation applied on a CIFAR-10 32×32 image using the "imgaug" Python libary [N] (ltr: flip, gaussian noise, crop)

Even though *undersampling* (the ignoring of samples of big classes) is also an option in extreme cases, it is more likely to have an negative impact on the learnability imbalance. That's because, when reducing the size of easier to learn classes, it will probably result in the decrease of their accuracy without an increase of the accuracy of the other classes. This hypothesis will also be checked in this research.

From the approaches mentioned above, the focal loss is the only one that can be tested on balanced datasets as it is. The others require a pre-training in order to determine the learnability differences and therefore the true positive rates, which then can be used to estimate or calculate how much these methods should affect each class.

III. APPROACH

TODO second section here

IV. THIRD SECTION

TODO third section here

REFERENCES

[1] Name Name, Name Name, Name Name (2006). Title, 24(1), 29-33.