Validation for Learning to Reweight Examples for Robust Deep Learning

Summer Research Fellowship (SRF) 2020 for Science Students

LIANG Zhuo; Supervisor: Dr. LIU Zhonghua
Poster No.: B4; Student's Major: Decision Analytics; University Number: 3035608849

Abstract

Regularizers and example reweighting algorithms are used to solve problems like training set biases and label noises. However, continuous careful hyperparameter adjustment is needed to have a satisfactory outcome. So, we propose a meta-learning algorithm that learns to assign weights base on gradient descent. This method avoids additional hyperparameter tuning, performs well on class imbalance and noisy labeled data sets, and only needs a small and clean validation set.

Introduction

Deep neutral networks (DNNs) tend to overfit to training set biases. Training set biases sometimes can be solved by resampling, which is to assign different weights to different samples and minimize the weighted training loss. The weights are typically calculated using training loss in many algorithms, for example, AdaBoost (Freund & Schapire, 1997), hard negative mining (Malisiewicz et al., 2011), self-paced learning (Kumar et al., 2010), and other more recent work (Chang et al., 2017; Jiang et al., 2017).

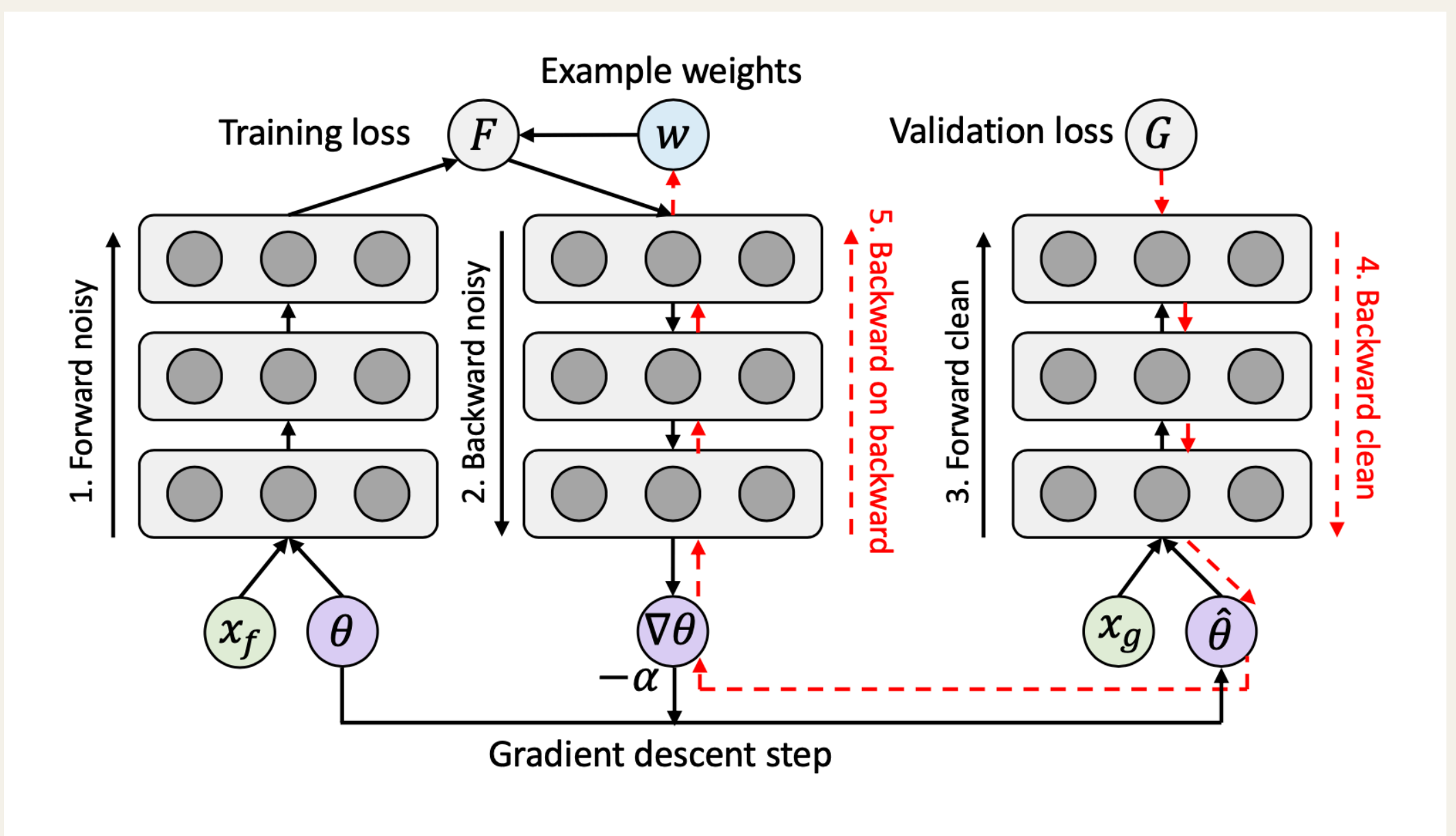
However, there are two conflicting views on the problem of training set biases. For noisy labeled datasets, we prefer to train samples with small loss because they are more likely to be clean. Yet for imbalance datasets, we often choose to train samples with big loss since they usually are the minority. If the dataset is imbalance and, at the same time, noisy labeled, how should we assign the weights becomes our major focus. In the paper, we proved that we need a small, clean, and unbiased validation set to learn to train.

Different from the existing methods, we use an online meta-learning algorithm. In general, validation comes after the training is done, and if we regard sample weights as hyperparameters to adjust, the cost is very high. Instead, we do validation in every iteration and regulate the weight of the current batch dynamically. This new approach works well on both class imbalance and noisy label problems.

Methodology

Let (x, y) be an input-target pair, and $\{(x_i,\ y_i),\ 1 \le i\ \le N\}$ be the training set. $\{(x_i^v,\ y_i^v),\ 1\ \le i\ \le M\}$ is a small unbiased and clean validation set where $M \ll N$ and the validation set is in the training set. In the past general methods, we use $\theta^*(w) = argmin_\theta \sum_{i=1}^N w_i\, f_i(\theta)$ to represent the weighted loss function; here, w is the hypermeter that needs adjustment, and $w^* = argmin_{w,w\ge0} \frac{1}{M} \sum_{i=1}^M f_i^v(\theta^*(w))$ shows that the performance on the validation set decides the value of the hyperparameter.

Our online approximation method inspects the descent direction of some samples on the training loss surface and do adjustments by the similarity to the descent direction on the validation loss surface. We introduce the vanilla SGD: $\theta_{t+1} = \theta_t - \alpha\nabla(\frac{1}{n}\sum_{i=1}^n f_i(\theta_t)$, where we want to detect the effect of sample i towards validation set at step t; and we consider perturbing the weight for every sample in the mini batch, SGD then becomes: $f_{i,\varepsilon}(\theta) = \epsilon_i f_i(\theta)$; $\hat{\theta}_{t+1}(\epsilon) = \theta_t - \alpha\nabla\sum_{i=1}^n f_{i,\epsilon}(\theta)\,|_{\theta=\theta_t}$. If we use $\epsilon_t^* = argmin_\epsilon \frac{1}{M}\sum_{i=1}^M f_i^v(\theta_{t+1}(\epsilon))$ to find the optimal ϵ that minimizes the validation loss, the calculation cost is very high. So, we simplify the above step, instead of using the whole valid set for calculation, we only use a valid sample of a mini batch: $u_{i,t} = -\eta \frac{\partial}{\partial\epsilon_{i,t}} \frac{1}{m}\sum_{j=1}^m f_j^v(\theta_{t+1}(\epsilon))|_{\epsilon_{i,t}=0}$, and to keep w as positive, we rectify the output: $\widetilde{w}_{i,t} = \max(u_{i,t}, 0)$.

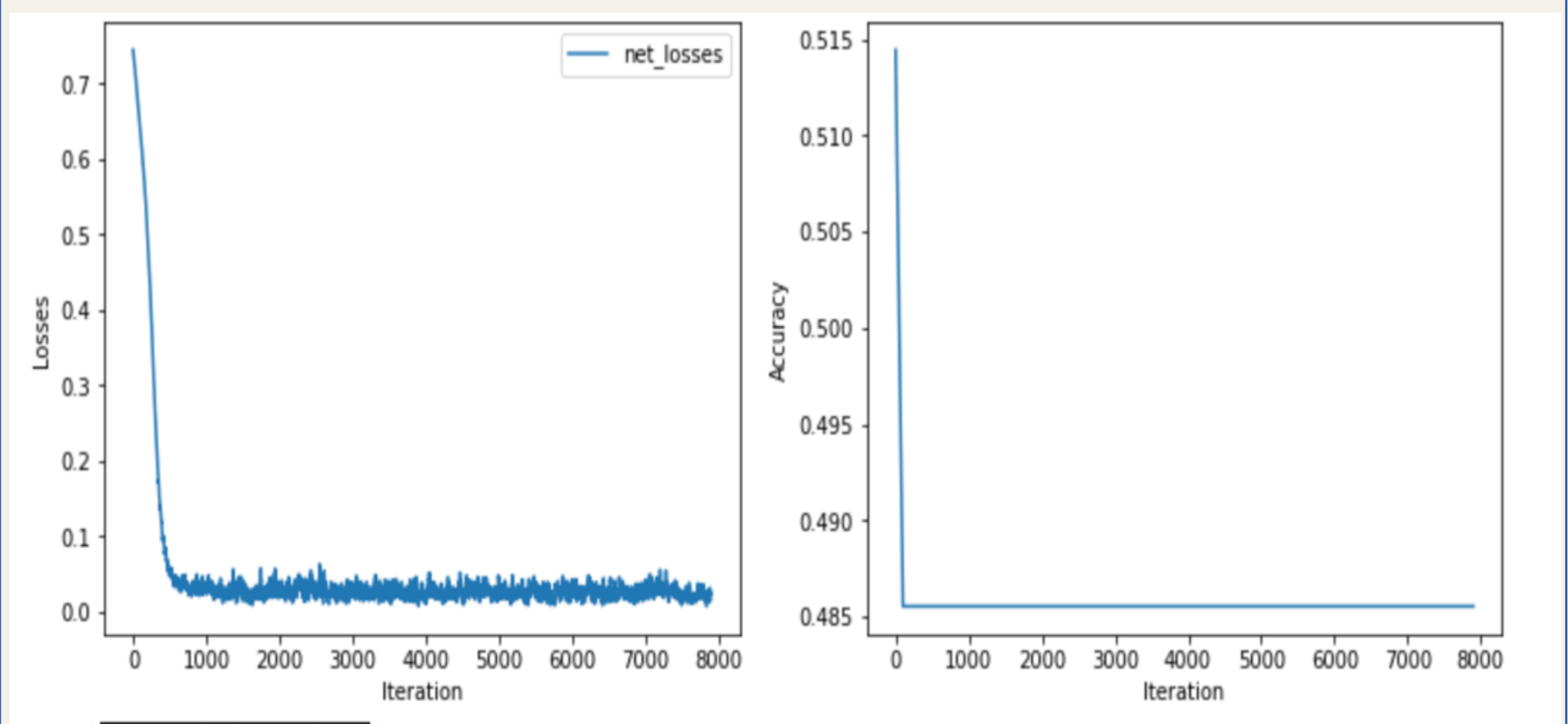


Methodology

In short, we used to choose a mini batch when we calculate the training loss. Now, to reassign weighting to samples in every batch, we first use a mini batch in the valid set to calculate the validation loss, and recalculate the weighting based on the validation loss. Training loss is used to update the model parameters, while validation loss for weighting, which is hyperparameter. This is how meta-learning works. If the distribution of the training sample is similar with the validation one, they share similar descent direction as well; this kind of sample is a "good" one which we need to increase its weight, and vice versa. We could make the model unbiased by this way.

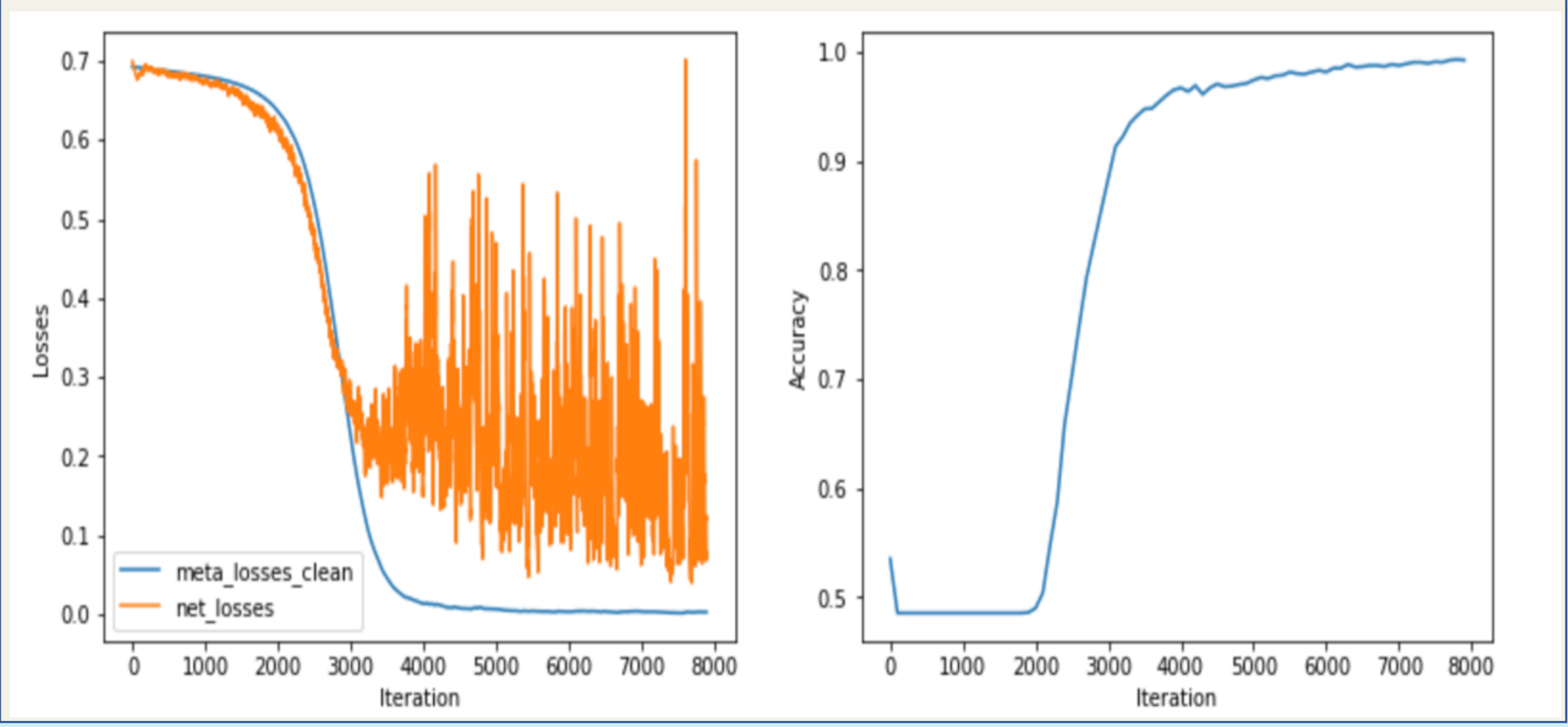
Validation

Refer to danieltan07's code on GitHub (https://github.com/danieltan07/learning-to-reweight-examples), I ran the experiments for the class imbalance problem on Google Colaboratory. I tried to test on two classes, '8' and '2' of the MNIST dataset, where '8' is the dominant class. We can see that at first, the model could not accurately differentiate the two numbers due to heavily imbalanced data.



```
100%|████████████| 8000/8000 [08:05<00:00, 16.49it/s]
```

Using the new method proposed by the paper, though I make the dataset even more imbalanced, the model is robust and gives very high accuracy.


meta_losses_clean
net_losses
Losses
Accuracy
Iteration