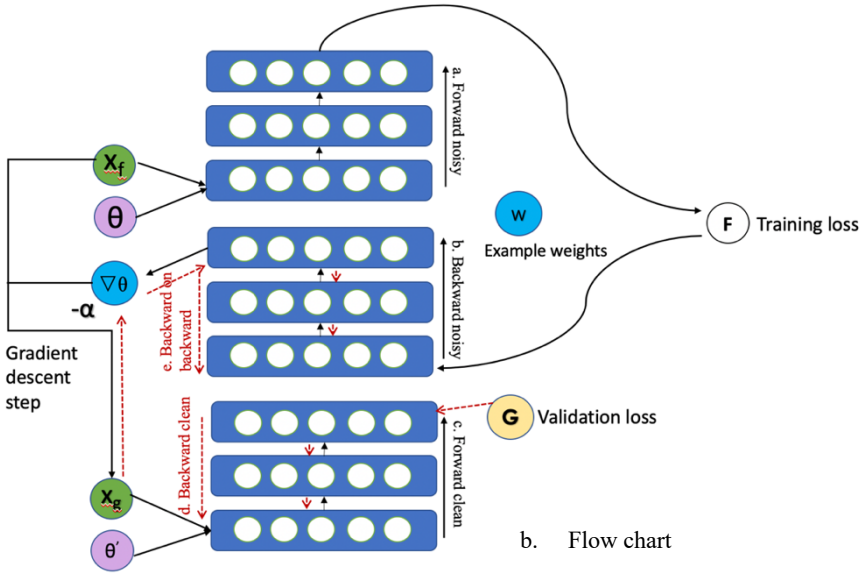
A Study on an Algorithm to Reweight Examples for Robust Deep Learning

Poster No.:B5
Student: Wu Yuxi UID:3035534957
Student Major: Decision Analytics
Supervisor: Dr. Liu Zhonghua

Abstract

Deep neural networks play an important role in modeling for supervised learning tasks. Researchers devise new algorithms to forestall overfit and mitigate the effect of set biases and label noises. The meta-learning algorithm we investigate offers a new way to efficiently deal with a range of problems, such as set biases and label noises, using a clean validation set. This study mainly examines the new algorithm's efficiency towards set bias, which served its purpose.



b. Flow chart

Theory and Hypothesis

The main idea and unique spirit of this approach is to adapt the weight of each data through a single optimization loop. During each iteration of training, we compare the descent direction of training sets, locally on the training loss surface, and that of validation sets locally on the validation loss surface. On the basis of this comparation, we further adjust the weight in the training set and improve the accuracy.

An example algorithm is on the right, and a flow chart is as above.

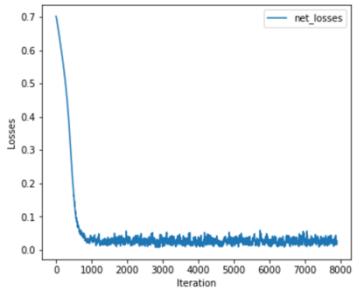
I used the standard MNIST handwritten digit classification dataset of class 3

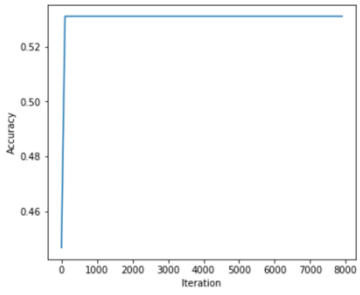
Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

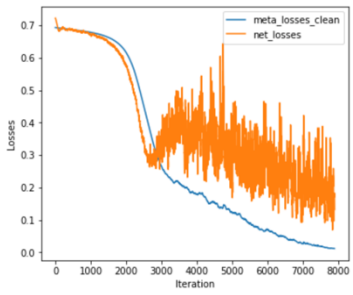
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: θ_T

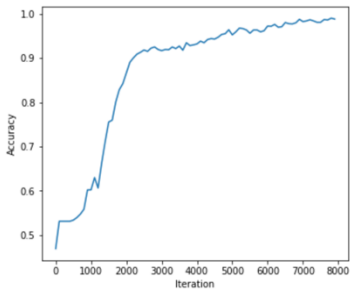
for $t = 0 \ldots T-1$ **do**
  $\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$
  $\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$
  $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
  $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
  $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
  $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
  $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
  $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
  $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
  $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
  $\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$
  $\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$
end for

a. Example algorithm









Experiment

Referring to danieltan07's code on GitHub (https://github.com/danieltan07/learning-to-reweight-examples), I ran the experiments using the original general algorithms on Google Colaboratory, and the results are the same as the original study. The accuracy was not satisfying and very close to random guesses.

Following the new algorithms, the accuracy raised significantly and was very close to 1.

Conclusion

The algorithm that we studied is for reweighting training examples and training more robust deep learning models is significantly effective with class imbalance. This method can be applied directly to any deep learning architecture, and is expected to train end-to-end without any additional hyperparameter research, which implies great contribution and bright future.