Advanced Scheduling Algorithm for Managing Pseudo Labels in Semi-Supervised Learning

Donghyun Kim
20170086
dong32@kaist.ac.kr

Yongmin Lee
20170475
http://github.com/chocolatefudge/AdThMix

Abstract

With a great amount of labeled data, the role of unlabeled samples in semi-supervised learning often becomes insignificant. Because proper usage of unlabeled data is crucial for semi-supervised learning to be applicable in real-world problems, we have to refrain from training steps to be dominated by labeled data. To utilize unlabeled data more elegantly, we devised an advanced scheduling algorithm for managing pseudo labels in intermediate steps. Our algorithm filters the pseudo label by its confidence depending on the model's current learning phase. As a result, it increased the model's performance as well as adding more robustness where there are few labeled data. We gained 1.25% additional top 1 accuracy compared to MixMatch. Our research also can be combined with other SSL methods, since it's compact and well applicable to other domains.

1. Introduction

Recent achievements in Semi-Supervised learning allowed us to gain significant performance when not all data have their labels. It reduced a lot of workload since manually labeling data is quite a tough job. However, handling unlabeled data also can be troublesome in many situations because we don't have exact answer labels for them.

MixMatch [1] reached great performance by mixing labeled and unlabeled data with MixUp augmentation. It produces pseudo labels from unlabeled data and mix it with labeled data to enhance the model's strength. However, we found some problems with unlabeled data handling in MixMatch. Because unlabeled data error is calculated through MSE and loss term has a relatively small scale, the training phase was often dominated by labeled data. Unlabeled data, therefore, provided some safety margin and added robustness to model, but failed to actively participate in training.

In a relatively prolific condition where many labeled data is given, MixMatch's approach might perform well. However, it cannot be ideal for a semi-supervised setting, because the usage of unlabeled data is quite important. To make unlabeled data have great influence in the training phase, appropriate scheduling and thresholding are necessary. The policy that manages pseudo labels generated from unlabeled data is indeed suggested in many recent SSL papers. However, handling a pseudo label is not usually considered as the main part of the algorithm but as an auxiliary method.

Therefore, in this research, we focus in-depth on the method to manage unlabeled data by providing a threshold for pseudo labels. Our main algorithm is adaptive thresholding, which computes the threshold value by considering the model's current learning pace. Starting from crystal-clear data can help the model to build a robust base. As training proceeds, providing more hard, boundary-lying data can also help the model to distinguish boundary case, while avoiding to overfit. Our method increased MixMatch's performance at a reasonable margin without appending complicated computation. Also, it can be further combined with other techniques or utilized in other domains.

2. Methods

2.1. Pseudo Labels

Because we don't have any labels for unlabeled data, a model requires some criterion to learn from them. A pseudo label is a straight-forwarded technique that generates 'pseudo' labels from unlabeled data using models. By feeding unlabeled data into our model, we could just pick up the class which has the maximum predicted probability [6] to use as if they're real labels. In general, however, created targets for unlabeled data doesn't result in perfect one-hot shape. It has higher confidence in some classes, or it might lie in boundaries with not having clear confidence in any of the classes. Furthermore, the generated pseudo label's quality depends on the model's current performance which might be very poor in the early phase of training. As a result, we need additional techniques to utilize pseudo label in SSL.

Algorithm 1 AdThMix: Pseudocode for Adaptive threshold policy.

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = ((x_b, p_b); b \in (1, ..., B))$, batch of unlabeled examples $\mathcal{U} = (u_b; b \in (1, ..., B))$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.

for b=1 **to** B **do**
  $\hat{x} = \text{Augment}(x_b)$
  for k=1 **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k p_{model}(y|\hat{u}_{b,k}; \theta)$
  $q_b = \text{Sharpen}(\bar{q}_b, T)$
end for
$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, ..., B))$
$b = \text{Argsort}(max(q_b), b \in (1, ..., B))$ ▷ Sort index of $\hat{\mathcal{U}}$ by $max(q_b)$
$\hat{\mathcal{U}} = ((\hat{u}_{b,k}; b \in (1, ..., B), b \leq B \cdot \frac{train\_accuracy}{100}, max(q_b) \geq 0.5)$ ▷ Applying adaptive percentile and threshold
$\mathcal{W} = \text{Shuffle}(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}}))$
$\mathcal{X}' = (\text{MixUp}(\hat{\mathcal{X}}_i, \hat{\mathcal{W}}_i); i \in (1, ..., |\hat{\mathcal{X}}|))$
$\mathcal{U}' = (\text{MixUp}(\hat{\mathcal{U}}_i, \hat{\mathcal{W}}_{i+|\hat{\mathcal{X}}|}); i \in (1, ..., |\hat{\mathcal{U}}|))$
return $\mathcal{X}', \mathcal{U}'$

2.2. Using Unlabeled Data

Based on generated pseudo labels, the usage of unlabeled data could be categorized into two contrastive approaches. In one viewpoint, we might suppose that generated pseudo labels, even though it's correct or not, will eventually help the model to learn from them. This enables the model to utilize all pseudo labels with some preprocessing such as sharpening and augmentations [1]. On the other hand, one might be unsure about generated pseudo label's quality, since misclassified targets might yield confirmation bias. In this approach, we might provide some threshold to use unlabeled data only if it satisfies some criterion [3]. However, only selecting classified pseudo labels might fail to generalize well and fall in overfitting.

2.3. Adaptive Threshold

To overcome the downsides of methods suggested in 2.2, we developed a thresholding algorithm called Adaptive Thresholding. Basic idea is to provide appropriate threshold values that consider the model's current ability. Modifying the amount of unlabeled data in the training phase was already suggested in the curriculum labeling [4]. However, more detailed coordination of threshold value is desirable than manually changing the percentile in fixed scale.

As Described in the pseudocode in Algorithm 1, most of our algorithm inherits MixMatch's. The difference comes in thresholding pseudo labels, where the threshold is calculated via maximum value between 0.5 and the model's current train accuracy. Because the threshold depends on the model's accuracy, in the earlier step of training, only a few, very confident unlabeled data is selected. This helps the model to avoid bias and increase its accuracy. After the model has achieved sufficient ability, more unlabeled data with less confidence value is fed into the model. This helps the model to learn through boundary cases and doesn't overfit to only some of the datasets. Also, a minimum value of 0.5 acts as a safety margin so that model is not exposed to unlabeled data which has low confidence and could confuse the model.

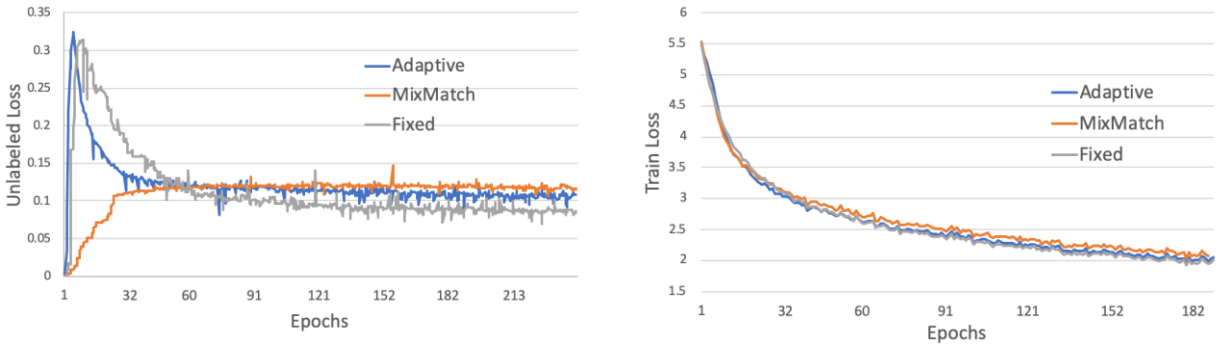
$$\mathcal{X}', \mathcal{U}' = \text{AdThMix}(\mathcal{X}, U, T, K, \alpha) \tag{1}$$

$$\mathcal{L}_\chi = \frac{1}{\mathcal{L}|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} H(p, p_{model}(y|x; \theta)) \tag{2}$$

$$\mathcal{L}_u = \frac{1}{\mathcal{L}|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} H(q, p_{model}(y|u; \theta)) \tag{3}$$

$$\mathcal{L} = \mathcal{L}_\chi + \lambda_u \mathcal{L}_u \tag{4}$$

For the loss function eq. (4), we used cross-entropy loss for unlabeled data in eq. (3) instead of a mean-squared error. In MixMatch, MSE is preferred in the sense that it is bounded and less sensitive to wrong predictions [1]. However, in our algorithm, we carefully choose unlabeled data by an adaptive threshold. Therefore, it's less vulnerable to predicting wrong pseudo labels. As a result, switching to cross-entropy gains more effect in unlabeled loss term while not increasing risks.



(a) (b)

Figure 1: Loss values as a function of training epoch with different threshold policies. (a) Unlabeled losses which are computed by only unlabeled data show notable difference, (b) while total losses decrease in almost the same manner.

Top1 Accuracy (%)	
MixMatch	73.89
Fixed Threshold	72.84
Adaptive Threshold	73.96

Table 1: Top1 accuracies when using different threshold policy. All methods in this experiment used MSE loss for unlabeled data loss.

Top1 Accuracy (%)	
MixMatch	73.89
Adaptive Threshold ($\lambda_u = 1$)	75.06
Adaptive Threshold ($\lambda_u = 1.5$)	75.14
Adaptive Threshold ($\lambda_u = 2$)	74.79

Table 2: Top 1 accuracies when using different weights for unlabeled loss (λ_u). Cross entropy loss was used for Adaptive Threshold method.

Top1 Accuracy (%)	
MixMatch	59.58
Adaptive Threshold	61.15

Table 3: Top1 accuracies when using only 4000 labeled data. Cross entropy loss was used for Adaptive Threshold method with λ_u= 1.

3. Experiments

3.1. Implementation Details

In all experiments, we used the same network architecture (EfficientNet-b3 [5] with 11.1M parameters). The model is pretrained from https://github.com/lukemelas/EfficientNet-PyTorch. Every dataset used for experiments is a fashion dataset provided by NAVER, which has 265 labels. There are approximately 16,000 labeled data and 40,000 unlabeled data for the training set, and 4,000 unlabeled data for validation. The total train was preceded in 300 epochs, while the batch size of the labeled and unlabeled data are 20 and 50 each. We used 5e-4 as our learning rate and SGD optimizer with a weight decay of 0.0004. T=0.5 is applied for sharpening. Otherwise noted, most of our implementation details follow one of MixMatch [1]. Specific hyperparameters for fixed/adaptive thres
holding policy is described in each experiment section.

3.2. Threshold Policy

We evaluate the performance of the model under the MixMatch baseline with only differing threshold schedule. The fixed schedule used a threshold value of 0.8, while the adaptive threshold used current training accuracy as a percentile of unlabeled data to use, and MSE loss was used for all unlabeled losses. Despite that there was a remarkable difference in unlabeled loss in fig. 1a, the entire loss was not affected by unlabeled loss in fig. 1b. Therefore, entire loss and accuracy in Table 1 produced a similar result regardless of the threshold policy. This implied that the training phase is often dominated by labeled data and we need further modification on loss and thresholding to increase the impact of unlabeled data.

3.3. Adaptive Threshold with CE Loss

By selecting the final version of the adaptive threshold policy mentioned in 2.3, a significant amount of performance was gained compared to the original MixMatch. By using λ_u =1, 75.06% of accuracy was

achieved in Table 2. Things to note in this experiment is that this margin is only generated by filtering unlabeled data in an adaptive manner and changing loss function. Thus, the result in this experiment expresses adaptive scheduling has a notable benefit on MixMatch, as well as it is possible to further be improved with some other technologies such as augmentation or distribution alignment.

Also, by trying on different λ_u values on an adaptive threshold, we found that λ_u of 1.5 slightly outperforms λ_u =1, while accuracy decreased when we set λ_u to 2 in Table 2. This implies that our effort to enhance unlabeled data's impact on training might be too much. Appropriate weight value between labeled and unlabeled loss term could be determined by further hyperparameter fitting.

3.4. Ablation Study

To show that our policy benefits from unlabeled data more than the original MixMatch, we conducted an ablation study with less amount of unlabeled data. With only 4k labeled data, the gap between the two algorithm's accuracy slightly increased in Table 3. In MixMatch, due to the small amount of labeled data, the model produced some uncertain predictions on pseudo labels, which biased the training phase. In contrast, the adaptive algorithm succeeds to gain a certain amount of performance in this scarce condition. The main reason is that it could still begin training from easy and clear unlabeled data and widen its range to boundary values as a model's performance increases.

4. Conclusion

Only changing a threshold policy in MixMatch setting didn't improve its performance by a significant margin, since most of the training in MixMatch is dominated by labeled data. However, using an adaptive threshold with cross-entropy loss resulted in non-negligible improvements compared to original MixMatch. By carefully choosing pseudo labeled data regarding the model's performance gained more accuracy while not overfitting too much on some of the unlabeled data. On the ablation study, it has also shown that adoptive policy also survives when labels are scarce. This acts as a solid base on our research that adaptive thresholding utilizes unlabeled data better, and hence more useful on conditions where many unlabeled data are given, which is most SSL researcher's main focus.

Another strength of our algorithm comes from the fact that it's not computationally expensive and open for other methods to adopt it. Since it doesn't increase original MixMatch's complexity much, the adaptive threshold can also act as a great helper method for other existing algorithms, too. Since most of the recent SSL methods are hybrid with many components, our thresholding policy might be a great tool to gain additional power for utilizing unlabeled data.

References

[1] Berthelot, David, et al. "Mixmatch: A holistic approach to semi-supervised learning." *Advances in Neural Information Processing Systems*. 2019.

[2] Berthelot, David, et al. "ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring." *arXiv preprint arXiv:1911.09785* (2019).

[3] Sohn, Kihyuk, et al. "Fixmatch: Simplifying semi-supervised learning with consistency and confidence." *arXiv preprint arXiv:2001.07685* (2020).

[4] Cascante-Bonilla, Paola, et al. "Curriculum Labeling: Self-paced Pseudo-Labeling for Semi-Supervised Learning." *arXiv preprint arXiv:2001.06001* (2020).

[5] Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *arXiv preprint arXiv:1905.11946* (2019).

[6] Lee, Dong-Hyun. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks." *Workshop on challenges in representation learning, ICML*. Vol. 3. 2013.

[7] Arazo, Eric, et al. "Pseudo-labeling and confirmation bias in deep semi-supervised learning." *arXiv preprint arXiv:1908.02983* (2019).