Report on Do We Need Zero Training Loss After Achieving Zero Training Error (ICML 2020)

Anonymous Author(s)
Affiliation
Address
`email`

1 Paper Summary

In machine learning, training a model too much means that both the training error and training loss will eventually approach zero. Training error refers to the percentage of training samples that the model got incorrect, and training loss refers to the value of the objective function of which the model is minimizing. Since the training data can be memorized by the model, the training loss can become zero. At this point, the model over-fits to the training data set, and the trained models do not generalize well to new data, causing reduction in test performance. This work proposed a new approach dubbed as "flooding" to address this problem. This idea is to induce a limit to the reduction of the training error to a small, reasonable value called a "flooding level", and let the model float around that value. More specifically, the model would do mini-batch gradient ascent if the loss is smaller than the flooding level.

Other previous regularization methods deal with over fitting in various ways including early stopping and smoothing the training labels. This work focuses on the assumption that zero training loss is harmful. While other regularization methods want to minimize the training loss, flooding allows users to control the level of training loss to a level other than zero.

2 Related Work

The authors referred to many papers in the background section to discuss about this work's novelty. Drawing similarities and inspirations from previous work, most of these related works tackle the same generalized issue of overfitting models.

Overfitting to the data has always been a problem since the beginning of machine learning. Regularization methods are ways to prevent training models from overfitting and allow them to generalize better. Some well known methods include weight decay[1], dropout[2], and early stopping[3]. Early stopping and weight decay are methods that were used in comparison in the experiments section of this work. The authors claims that the main difference between this work and other regularizers is the ability to choose the training loss directly rather than training for zero training loss.

The authors cited double descent studies[4, 5]. This phenomenon speculates about the relationship between model complexity and test error. As simple models increase in complexity, the test error will decrease until the model becomes too complex to the point where test error will increase. This is the issue with many modern machine learning techniques, which is that complex models lead to poor generalization. Early stopping is a method employed specifically to handle this issue. However, the double descent studies found that when the model is complex, eventually, increasing the complexity will only decrease test error. This fits in with the current modern view on machine learning which is larger models generalize better. The experiment results from this work supports the double descent theory as the result graphs from the paper showed double descent curves.

This flooding idea can be generalized into putting a lower bound on risk to avoid overfitting. In this flooding technique, that risk is training loss. Previous work[6] has also set a lower bound on empirical risk. The methodology deployed by Kiryo et al. is different in that they are addressing negative empirical risk, while flooding is creating a positive lower bound.

In doing some research, I have found an existing work[7] that I feel is related to this work but not mentioned. In this related work, the authors tested the effect of using gradient ascent on a Boosting algorithm(Ada boosting) in order to build a stronger classifier. Similar to the problem statement of the paper in interest, Boosting algorithms often suffers from overfitting. This is yet another application of using the gradient ascent approach aimed to avoid overfitting.

3 Theorems and Algorithms

3.1 Main Algorithm

The implementation for flooding can be summed up in the following formula.

$$\tilde{J}(\theta) = |J(\theta) - b| + b$$

Where J is the original learning objective function, θis the model parameter, and b is the flood level specified by a user.

The effect of this new objective function is summarized as follows. When the objective is higher than the flood level, we will apply gradient descent. When the objective is lower than the flood level, we will apply gradient ascent. As a result, the training loss will oscillate around the flood level, which the paper claims and shows that this behavior will lead to better generalization and better performance.

3.2 Implementation

The authors used mini-batched stochastic optimization by applying gradient descent updates in the direction of the gradient of $|\widehat{R}(g) - b| + b$. By Jensen's inequality, the authors derive that

$$\tilde{R}(g) \leq \frac{1}{M} \sum_{m=1}^{M} |\widehat{R}_m(g) - b| + b$$

Where $\widehat{R}_m(g)$ is the empirical risk for m-th mini-batch.

Jensen's inequality states that the secant line of a convex function lies above the function. Since $\widehat{R}(g)$ is wrapped around a absolute value function, the gradient is indeed convex. This result indicates that doing mini-batched optimization will yield an upper bound of the full-batch case with $\tilde{R}(g)$

3.3 Theorem

For any measurable vector-valued function g, if b satisfies $b < R(g)$ and $Pr[\widehat{R}(g) < b] > 0$, then

$$MSE(\widehat{R}(g)) > MSE(\tilde{R}(g)).$$

Where $\tilde{R}$ is R applied with the flooding algorithm, and $\widehat{R}$ is the training empirical risk.

This theorem is proved in the appendix A section of the paper. The mean squared error of the flooded risk estimator is smaller than that of the original risk estimator without flooding. Smaller MSE would imply the model is closer to the actual data.

4 Review

The experimental data shown by authors have supported the claim that using the flooding technique improves with performance on the test data. In the synthetic data setting, the flooding technique consistently yielded higher test accuracy than the those without using flooding. The flood level of

0.00 to 0.50 in increments of 0.01 is trained, and the flood level with the best validation accuracy is chosen. The authors did not reason about the decision or how they decided to choose this level range. This implies a flaw in this technique that that picking a flood level is non-trivial, and could be quite computationally intensive. Furthermore, the large standard deviation of the flood level from the result shows the inconsistency of this method. For some cases(ex. Gaussians on low label noise without early stopping), the flood level chosen has a standard deviation larger than 0.10. The difficulty and overhead of picking a flood level may cause this technique to be impractical in the real world setting, especially if the trained flood level has such a high variance on the dataset.

In addition to synthetic data, the authors used some benchmark data sets to evaluate this flooding technique. It is worth to note that in the case of CIFAR-10 and CIFAR-100 with the ResNet44 model without any other regularization, using flooding actually degraded the test performance. In these tests, they ran a dataset with a specific model (ex. MLP) with a combination of regularization setups. The data suggests that flooding combining multiple regularization techniques can yield a higher performance.

5 Experiment

5.1 Synthetic Data

The authors used 3 types of synthetic data: Gaussians, Sinusoid, and Spiral. The model used is a 5-hidden-layer feedforward neural network with ReLU activation function. The network is trained for 500 epochs with 100 mini-batch size and learning rate of 0.001. The flood level is chosen from 0 to 0.5 in increments of 0.01. In addition, noise was added to the dataset by flipping the labels randomly. Noise was introduced in 3 levels: low(1%), middle(5%), and high(10%).

They reported the test accuracy of the flood level with the best validation accuracy. For the baseline without the effect of flooding, they used a flood level of 0. They have also checked performance of flooding with and without early stopping. Early stopping means that the model will stop training once the performance measured is over a threshold. Each method is ran over 10 trials. The first observation is that the flood level is larger for more label noises. In summary, the accuracy of the method applying flooding yielded with higher performance. With combination of flooding and early stopping, the performance is similar to flooding without early stopping. In some settings, flooding with early stopping performed slightly better than flooding without early stopping. This suggests that the combination of these techniques can be beneficial.

5.2 Benchmark Datasets

On top of synthetic datasets, the authors used MNIST, Kuzushiji-MNIST, SVHN, CIFAR-10, and CIFAR-100. The model for training is a two-hidden-layer feedforward neural network with ReLU activation function. They performed a exhaustive hyper-parameter search for the flood level from 0.00 to 0.10. Unlike the synthetic data, there was no additional label noise.

The authors tries to visualize the effect of the flooding technique by testing it in combination with other regularization methods. These include regularization techniques like weight decay and early stopping. In many cases, flooding improved the test accuracy. This implies that the models can be more powerful by combining flooding with other normalization techniques.

5.3 Experiment results

In this section, I aim to test the validity of the paper's results for the synthetic data set. The setup I ran was using the sinusoid data and the spiral data. The model used for training is multi-layered perceptron(mlp) model using stochastic gradient descent optimization and a learning rate of 0.01 over 200 epochs.

These two datasets are trained on the three noise levels: low(1%), middle(5%), and high(10%). One small adjustment I have made is that I have trained the flooding level range from 0.00 to 0.50 in increments of 0.05 instead of 0.01. Each trial lasted over an hour. This suggests that the training

Data	Noise	Without Flooding	With Flooding	Flood Level
Sinusoid	Low	95.70%	94.67%	0.05
Sinusoid	Middle	94.86%	96.16%	0.10
Sinusoid	High	94.88%	96.16%	0.10
Spiral	Low	100%	100%	0.10
Spiral	Middle	99.59%	100%	0.15
Spiral	High	95.89%	100%	0.15

Table 1: Experimental results derived from running the source code.

process is quite intensive. In the demo code, the authors only included 3 flood levels to train, and running running time of that setup took half an hour. The code and data can be found on Github[1].

Figure 1 to 3 shows the result of the performance in the accuracy as a function of epochs on the Sinusoid data. From figure 1, we can see that the test accuracy without flooding actually performed slightly better. This is most likely a result of limiting the flood level to be in an increment of 0.05. Because of the low noise, there may exist a flood level between 0.00 and 0.05 that performs better. Because of the flood level, training accuracy with flooding performances worse, but the test accuracy improves as shown by figures 2 and 3.

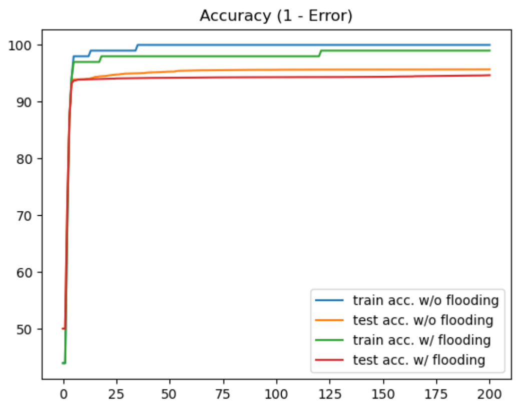

Figure 1: Low noise(1%) on Sinusoid

[1]https github.comDavidLiu0304cse257_flooding

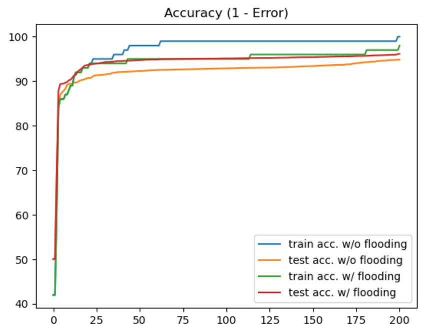

Figure 2: Middle noise(5%) on Sinusoid

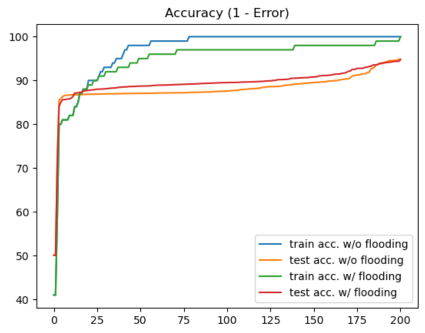

Figure 3: High noise(10%) on Sinusoid

Data	Noise	Without Flooding	With Flooding	Flood Level
Sinusoid	20%	92.89%	100%	0.15
Sinusoid	30%	82.82%	100%	0.15

Table 2: Running on the same setup as table 1 with different noise level.

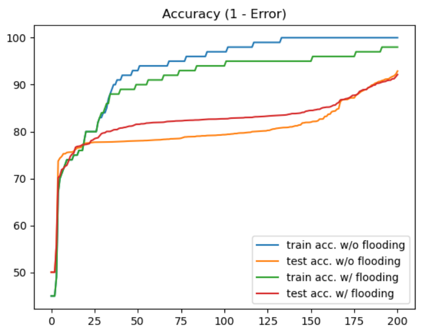

Figure 4: 20% noise on Sinusoid

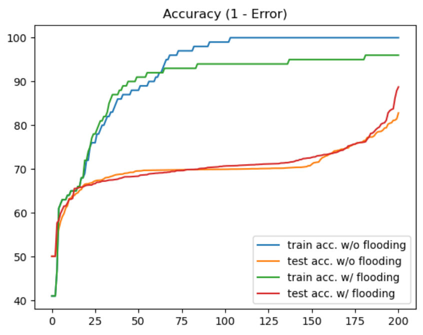

Figure 5: 30% noise on Sinusoid

5.4 Noise and Flood Level

An observation of the results from the paper and the results from running the source code is that higher noise level yields higher flood level. In this section, I run the same experimental setup as the previous section with the difference of increasing the noise level. The results are shown in table 2 and Figures 4 and 5 show the accuracy of the trained model as a function of epochs.

With higher level of noise, the base model becomes worse in performance. The results were surprising to me as flood level 0.15 resulted in test accuracy of 100%. It seems like this particular level works well for this dataset. As noise increase, the gap between the test accuracy of the models with flooding and without flooding grows at around 200 epochs. This suggests that using flooding may be a good way to train models that deal with with datasets containing lots of noisy data.

From running the experiments, the data supports the authors claims that flooding improves the performance. The correlation between the noise level and the flood level seems to be positive.

References

[1] Hanson, S.J. & Pratt, L. Y. Comparing Biases for Minimal Network Construction with Back-Propagation. 1988.

[2] Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research 15*, pages 1929-1958 2014.

[3] Morgan N. & Bourlard H. Generalization and Parameter Estimation in Feedforward Nets. 1990.

[4] Belkin et al. Reconciling Modern Machine-Learning Practice and the Classical Bias-Variance Trade-Off. 2019.

[5] Preetum et al. Deep Double Descent: Where Bigger Models and More Data Hurt. *ICLR*, 2020.

[6] Kiryo et al. Positive-Unlabeled Learning with Non-Negative Risk Estimator. 2017.

[7] Basha et al. Impact of Gradient Ascent and Boosting Algorithm in Classification. *INASS*, 2017.