COMP 551 Project 4

Byron Chen, Kevin Li, Isabella Hao

December, 2021

Abstract

To find an algorithm that can train sparse neural networks (SNN) without being bounded to specific architectures, this project aims to reproduce part of the paper: *"A Bregman Learning Framework for Sparse Neural Networks"* [2]. In this study, we use the codes provided by the authors in this repository to reproduce the results in a two-hidden-layer MLP, a ConvNet and ResNet-18. To further investigate if the algorithms can perform well in other models, VGG-16 and VGG-13 are implemented in this study. In the end, the results is found to be unstable, since we cannot reproduce the results even the codes are provided by the researchers. Apart from this, the Bregman algorithms fail to work in VGG-16 and VGG-13 which shows that they are model-dependent, thus not general.

1 Introduction

It is known that large and deep neural networks can produce great performances. However, a huge drawback is shown while working with them that is they require a huge amount of computational resources and this can be a big limitation for real-time architectures and normal users. Consequently, many researchers aim for training **sparse neural networks (SNN)** which is based on neglecting small parameters or those with little influence on the network output [1].

However, many approaches for training SNN bound to specific architectures. We decided to reproduce the paper: *"A Bregman Learning Framework for Sparse Neural Networks"* [2] which aims to train SNN with Bregman algorithms that can be available for any models.

In this study, we first experimented with three algorithms in a model with two hidden layers MLP, namely SGD, LinBreg, and ProxGD, and compared them with their accuracies and sparsities. The sparsity means the percentage of non-zero entries used which decides the computational resources a network needs. Then we picked the best algorithm LinBreg and compared it with its two variants, LinBreg with momentum and AdaBreg [2], which utilizes the Adam algorithm [4] provided in the reproduced paper. After training the SNN with the algorithms and compared the results with that in the paper, we found AdaBreg did not produce the same results as provided in the paper. Furthermore, we trained SNN with SGD, LinBreg and its variants in different networks, namely ConvNet, ResNet, VGG-16, and VGG-13 to see if they perform well in different models. However, we found the performance in ConvNet and ResNet is not as well as it is claimed in the paper. Furthermore, they failed to train the SNN in VGG-16 and VGG-13 which shows that the effects of the Bregman algorithms are model-dependent and thus not general.

2 Summary of the Paper

In this paper [2], the researchers propose a Bregman learning framework that utilizes "linearized Bregman iterations [3]." One of the advantages of the Bregman learning framework is that it can train "sparse neural networks without being bounded to specific architectures." The paper measures the sparsity of one network by its percentage of the non-zero entries. The authors have conducted their experiments progressively. First, they compare their baseline optimizer, LinBreg, with SGD and ProxGD; this experiment was conducted with a 2-layer MLP on the MNIST dataset. Concretely, the study has applied l_1-regularization to both ProxGD and LinBreg but not SGD; both ProxGD and LinBreg can arrive at a similar validation accuracy with much fewer non-zero entries. As a result, the authors have chosen the LinBreg with regularization parameter $\lambda = 0.1$ for future investigations. Then, they have derived two variants, namely LinBreg with Momentum and AdaBreg. With the same experimental setup, the study has found that AdaBreg has achieved the highest validation accuracy with the fewest non-zero entries. Next, the study tests their proposed optimizers in other tasks: the ConvNet model (the layers are shown in Figure 1) on the Fashion-

MNIST dataset and ResNet-18 on the CIFAR-10 dataset. In both settings, all three Bregman optimizers (LinBreg, LinBreg with Momentum, and AdaBreg) can achieve similar validation accuracy yet at a much lower model sparsity than Adam.

3 Datasets

Both the MNIST and the Fashion-MNIST dataset [5] consists of 60,000 images which we split into 55,000 images used for the training and 5,000 images used as the validation set. The CIFAR-10 dataset [6] we used consists of 50,000 images which we split into 47,500 images used for the training and 2,500 images used as the validation set. All the datasets are preprocessed with masks (M^l) where each entry in M^l is distributed with a parameter $r \in [0, 1]$, i.e.,

$$M^l_{i,j} \sim Bernoulli(r)$$

and the input parameters are initialized with 1% non-zero entries, i.e. $r = 0.01$ in this study for the Bregman and ProxGD optimizers.

4 Scope of Reproducibility

First, we wish to reproduce the results exhibited in the paper. Due to the limited resources, we cannot conduct all experiments laid out in the article. Specifically, we wish to prove three points. First of all, LinBreg surpasses ProxGD and SGD in model sparsity. Secondly, the two variants, LinBreg with Momentum and AdaBreg, perform better than LinBreg. Finally, all three Bregman optimizers can work well for ConvNet and ResNet-18. After reproducing these results, we wish to further explore the Bregman optimizers by confirming that they are general solutions and not pertaining to specific models. Hence, we shall analyze the Bregman optimizers' performance in other settings (VGG16 and VGG13 on CIFAR-10).

5 Experiments and Results

All the following experiments are done based on the codes given by the researchers which can be found in this repository. In this study, "the learning rate is chosen as $\tau = 0.1$ and whenever the validation accuracy stagnates, it is multiplied by 0.5."

5.1 Reproduce Results of Optimizers

In Figure 2 and Figure 3 we compared the training accuracies and validation accuracies of various optimizers with different λ. We compared the sparsity level of various optimizers in Figure 4. The figures show that the LinBreg can achieve higher validation accuracy than with SGD while only consuming 80% of the weights. For $\lambda = 0.1$, the LinBreg uses only 10% of the weights, yet achieves a validation accuracy that is 0.2% higher than that of SGD; the ProxGD, with a similar number of non-zero entries, has validation accuracy that is 2% lower than that of SGD. As a result, we pick LinBreg with $\lambda = 0.1$ for future investigations.

5.2 Reproduce Results of Bregman Optimizers

Given $\lambda = 0.1$, the performance of LinBreg, LinBreg with Momentum, and AdaBreg are compared. We recorded the trends of validation accuracy of the three Bregman optimizers in Figure 5 and Figure 6 and the trends of sparsity levels in Figure 7 and Figure 8. AdaBreg fails to train the model as the highest validation accuracy is approximately 10%. Moreover, AdaBreg does not reduce the sparsity of the model. Another observation is that having momentum in LinBreg does not improve the performance as suggested by the study. LinBreg with Momentum reaches a lower validation accuracy and higher sparsity than LinBreg does, though it converges faster.

5.3 Reproduce Results of Bregman Optimizers in ConvNet and ResNet

In this experiment, the ConvNet (Figure 1) was applied to the Fashion-MNIST dataset, and ResNet-18 was applied to CIFAR-10. For both settings, we tried all three Bregman optimizers and the Adam optimizer, which serves as a reference.

ConvNet

Table 1 shows the test accuracy, training accuracy, and sparsity of various optimizers with ConvNet. The Adam optimizer reaches the highest test accuracy of 91.3%. The LinBreg reaches 87.1% test accuracay with a sparsity level of 51.8%. The AdaBreg reaches 79.1% test accuracy with the lowest sparsity of 39.7%.

ResNet-18

Table 2 shows the test accuracy, training accuracy, and sparsity of various optimizers with ResNet-18. The Adam optimizer reaches the highest test accuracy of 92.5%. The LinBreg reaches 81.9% test accuracay with a sparsity of 11.2%. The AdaBreg reaches 86.2% test accuracy with the sparsity of 71.2%.

5.4 Further Investigations of Bregman Optimizers in VGGs

In this experiment, the VGG-16 and VGG-13 was applied to CIFAR-10. For both settings, we tried all three Bregman optimizers and the Adam optimizer, which serves as a reference.

VGG-16

Table 3 shows the test accuracy, training accuracy, and sparsity of various optimizers with VGG-16. The results obviously show that Bregman algorithms fails to train the SNN in VGG-16 since LinBreg and AdaBreg have only 9.9% and 10.4% test accuracies separately.

VGG-13

Table 4 shows the test accuracy, training accuracy, and sparsity of various optimizers with VGG-13. The results obviously show that Bregman algorithms fails to train the SNN in VGG-16 since LinBreg and AdaBreg have only 10.7% and 9.3% test accuracies separately.

6 Discussion and Conclusion

6.1 Results Comparison

By comparing Figure 2, Figure 3, and Figure 4 with Figure 9, Figure 10, and Figure 11 provided in the paper, it is obvious that the results are well reproduced, and LinBreg achieves a validation accuracy that is higher than SGD which does better than that shown in the paper. However, as in Figure 5 and Figure 7, we did not reproduce the effects of the AdaBreg, shown in the paper (Figure 12). It is easy to see that AdaBreg fails to train the networks.

In the next experiment, we partially reproduce the results in ConvNet. In Table 1, it shows that the accuracies achieved by the optimizers are acceptable with lower sparsity compared with that in Adam, However, the results are not reproduced completely since the sparesities of the algorithms are around $35\% - 45\%$ higher than that provided in the paper (Table 5).

In the experiment with Reset-18, Table 2 shows that the results are partially reproduced. We have shown that LinBreg and LinBreg with Momentum achieves the similar performance provided in the paper (Table 6). In contrast, AdaBreg shows a much higher sparsity than expected.

To prove the Bregman algorithms proposed by the researchers are not bounded to specific architectures, we train the VGG-16 and VGG-13 with Bregman algorithms. The results are shown in Table 3 and Table 4. It is obvious that all the Bregman algorithms fail to train the networks since the validation accuracy does not improve and stays low during training. Combined the above observations, we can see that the effects of the Bregman algorithms are model-dependent and thus not general.

Since we cannot reproduce the results even the codes are provided by the researchers, the Bregman algorithms per se might be unstable.

6.2 Future Investigation

As we have discussed previously, the effects of the Bregman optimizers are model-dependent. In the future, we may investigate the circumstances under which the Bregman optimizers are beneficial.

6.3 Important Points when Reproducing the Paper

In the paper[2], the sparsity is defined to be $1 - N_{Total}$ (percentage of non-zero entries). However, we found that the sparsity is implemented to be the same as N_{Total} in the codes provided. In this project, we adopt the definition of sparsity according to the codes provided, i.e., the sparsity of a model is its percentage of non-zero entries.

Statement of Contributions

Byron Chen - Report Write-up
Kevin Li - Experiment Design
Isabella Hao - Code Implementation

References

[1] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. “Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks”. In: arXiv preprint arXiv:2102.00554 (2021).

[2] Leon Bungert, Tim Roith, Daniel Tenbrinck, and Martin Burger. "A Bregman Learning Framework for Sparse Neural Networks" arXiv preprint arXiv:2105.04319 (2021).

[3] Wotao Yin, Stanley Osher, Donald Goldfarb, and Jerome Darbon. “Bregman iterative algorithms for ‘1-minimization with applications to compressed sensing”. In: SIAM Journal on Imaging sciences 1.1 (2008), pp. 143–168.

[4] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: arXiv preprint arXiv:1412.6980 (2014).

[5] Yann LeCun and Corinna Cortes. “MNIST handwritten digit database”. In: (2010). url: http://yann.lecun.com/exdb/mnist/.

[6] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. url: https://www.cs.toronto.edu/~kriz/cifar.html.

Appendix A: Graphs from Experiments

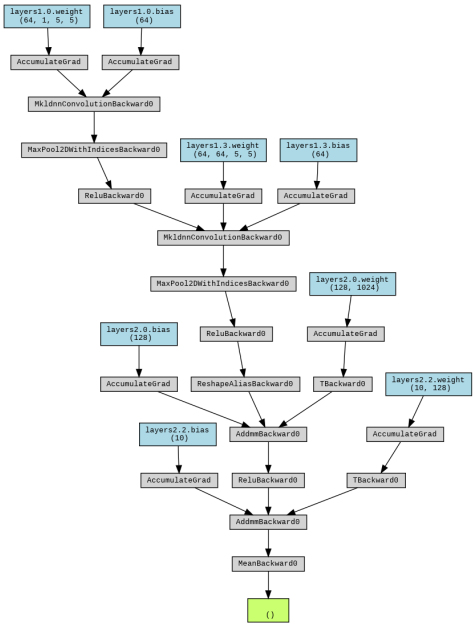

Figure 1: Visualization of the layers of ConvNet

Appendix B: Graphs from Experiments

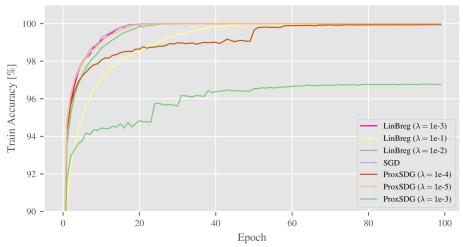

Figure 2: Training Accuracy of SGD, LinBreg, and ProxGD (wrongly labeled in the graph)

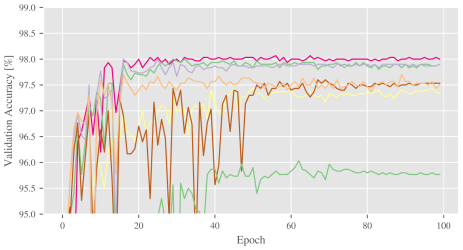

Figure 3: Validation Accuracy of SGD, LinBreg, and ProxGD

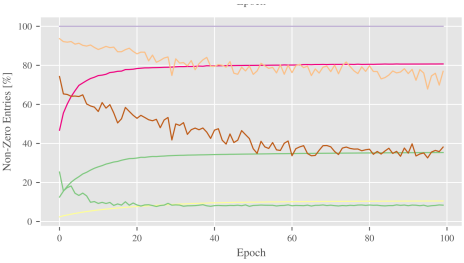

Figure 4: Sparsity levels of SGD, LinBreg, and ProxGD

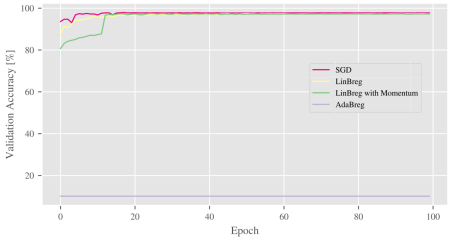

Figure 5: Validation Accuracy of SGD, LinBreg, LinBreg with momentum, and AdaBreg

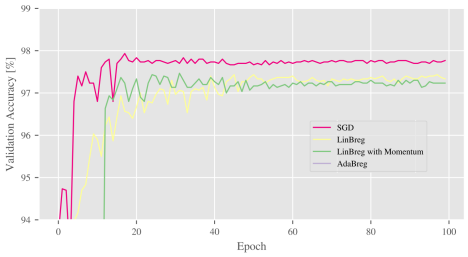

Figure 6: Validation Accuracy of SGD, LinBreg, and LinBreg with momentum

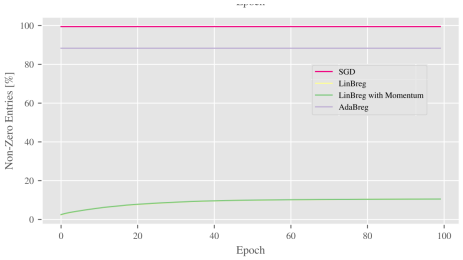

Figure 7: Sparsity levels of SGD, LinBreg, LinBreg with momentum, and AdaBreg

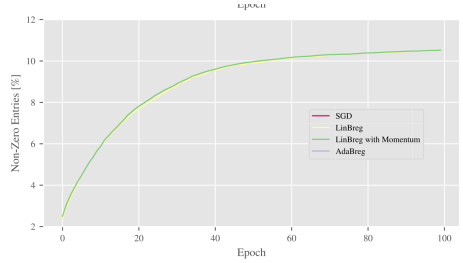

Figure 8: Sparsity levels of LinBreg and LinBreg with momentum

Appendix C: Graphs from the Paper

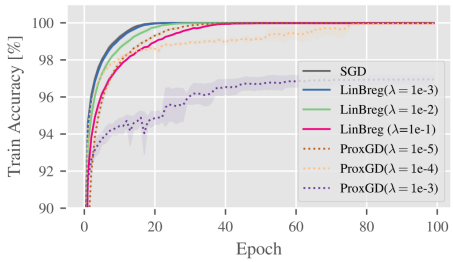

Figure 9: Training Accuracy of SGD, LinBreg, and ProxGD provided in the paper

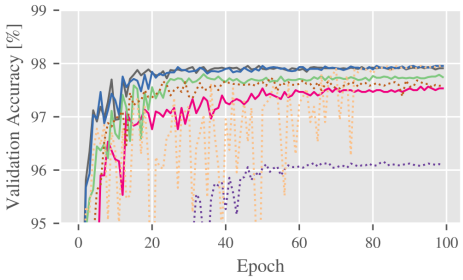

Figure 10: Validation Accuracy of SGD, LinBreg, and ProxGD provided in the paper

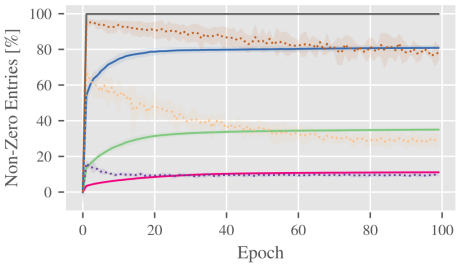

Figure 11: Sparsity levels of SGD, LinBreg, and ProxGD provided in the paper

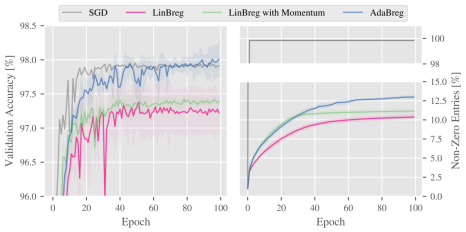

Figure 12: Validation Accuracy and sparsity levels of SGD, LinBreg, LinBreg with momentum, and AdaBreg provided in the paper

Appendix D: Tables

Table 1: Sparsity levels and accuracies on the Fashion-MNIST dataset with ConvNet after 100 epochs

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	91.3	100.0
LinBreg	51.8	87.1	94.1
LinBreg ($\beta = 0.9$)	50.2	87.7	93.9
AdaBreg	39.7	79.1	81.3

Table 2: Sparsity levels and accuracies on the CIFAR-10 dataset with ResNet-18 after 100 epochs

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	92.5	100.0
LinBreg	11.2	81.9	91.1
LinBreg ($\beta = 0.9$)	9.1	81.2	90.0
AdaBreg	71.2	86.2	98.1

Table 3: Sparsity levels and accuracies on the CIFAR-10 dataset with VGG-16 after 100 epochs

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	84.7	100.0
LinBreg	88.9	9.9	10.1
LinBreg ($\beta = 0.9$)	89.5	9.3	10.2
AdaBreg	58.2	10.4	10.8

Table 4: Sparsity levels and accuracies on the CIFAR-10 dataset with VGG-13 after 50 epochs

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	84.4	100.0
LinBreg	92.5	10.7	9.9
LinBreg ($\beta = 0.9$)	92.5	10.0	10.0
AdaBreg	60	9.3	10

Table 5: Sparsity levels and accuracies on the Fashion-MNIST dataset with ConvNet after 100 epochs provided in the paper

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	92.1	100.0
LinBreg	1.9	89.2	91.1
LinBreg ($\beta = 0.9$)	2.7	89.9	93.8
AdaBreg	2.3	90.5	93.6

Table 6: Sparsity levels and accuracies on the CIFAR-10 dataset with ResNet-18 after 100 epochs provided in the paper

Optimizer	N_{total} in [%]	Test Acc	Train Acc
Adam	100	93.6	100.0
LinBreg	5.5	90.9	99.5
LinBreg ($\beta = 0.9$)	4.8	90.4	100
AdaBreg	14.7	92.3	100