Linear Backprop in non-linear networks

Mehrdad Yazdani
Qualcomm Institute
University of California San Diego
La Jolla, CA 92093
myazdani@gmail.com

Abstract

Backprop is the primary learning algorithm used in many machine learning algorithms. In practice, however, Backprop in deep neural networks is a highly sensitive learning algorithm and successful learning depends on numerous conditions and constraints. One set of constraints is to avoid weights that lead to saturated units. The motivation for avoiding unit saturation is that gradients vanish and as a result learning comes to a halt. Careful weight initialization and re-scaling schemes such as batch normalization ensure that input activity to the neuron is within the linear regime where gradients are not vanished and can flow. Here we investigate backpropagating error terms only linearly. That is, we ignore the saturation that arise by ensuring gradients always flow. We refer to this learning rule as Linear Backprop since in the backward pass the network appears to be linear. In addition to ensuring persistent gradient flow, Linear Backprop is also favorable when computation is expensive since gradients are never computed. Our early results suggest that learning with Linear Backprop is competitive with Backprop and saves expensive gradient computations.

1 Overview

It is has been long known that deep neural networks with non-polynomial and non-linear units are universal function approximators [Leshno et al., 1993]. In the early days of neural network research, however, it was not clear what learning algorithms would find the optimal set of synaptic weights for effective learning. Rosenblatt’s pioneering work essentially only learned the weights at the output layer of the Multi-Layer Perceptron and keeping the input-layer weights fixed at random while Fukushima used Hebbian learning. Backprop as introduced into neural network research [Werbos 1974, Rumelhart et al. 1986] has been enormously successful at learning diverse sets of tasks by various deep neural architectures and as a result is by far the most used learning algorithm.

Although enormously successful, Backprop is a highly sensitive learning algorithm and numerous tricks have been collected to make it work in practice [Baydin et al. 2016, Bottou 2012]. Some of these issues are: dead or saturated units, appropriate learning rates, batch sizes, number of epochs in addition to many other issues. In particular, considerable effort has been placed into avoiding saturated units. The primary problem with saturated neurons is that gradients vanish in these regions and hence learning comes to a halt. As a result, considerable effort has been placed into ensuring that the input activity to neurons are in the linear region. Some of these efforts are the introduction of regularization such as $l_2$ penalty (also referred to as weight decay) [Krogh and Hertz 1992, Srivastava et al. 2014], batch normalization [Ioffe and Szegedy 2015], and careful weight initialization schemes [Glorot and Bengio 2010, Mishkin and Matas 2015]. Other solutions is to consider activation functions that have limited saturating regions to ensure gradient flows [Nair and Hinton 2010, Klambauer et al. 2017, Gulcehre et al. 2016] extensively study activation functions with non-saturating regions to provide gradients.
We begin by considering a synthetic data example to investigate the overfitting and generalization capabilities of Linear Backprop. There has been considerable recent interest in the generalization capabilities of deep neural networks [Zhang et al., 2016, Poggio et al., 2018]. The gist of the research is how networks that have far more parameters than training data (often by many orders of magnitude) generalize well to unseen data. Many experiments have shown that even when training loss is 0 (severe overfitting), there is still generalization [Zhang et al., 2016, Wu et al., 2017].

Here we reproduce the motivating example from [Wu et al., 2017] to understand if Linear Backprop shares similar generalization properties. We consider learning the third-order polynomial \( y = x^3 - 3x^2 + x + 1 + N(0,0.1) \). In this experiment, the training set consists of only 5 points and the neural network is trained until the training error is small (for example, \( \leq 1 \times 10^{-6} \)). The neural network is a feed forward Multi-Layer Perceptron (MLP) with 4 hidden layers each with width of 50 ReLU units (over 7,000 parameters). For a problem with 5 points, this is a highly overparameterized network.
We reproduce the result from [Wu et al., 2017] that such an overparameterized network overfits the training data yet also generalizes gracefully to unseen data (see Figure 1). We also observe similar generalization capability with the same architecture when trained with Linear Backprop. The Linear Backprop also overfits on the training data and has similar generalization when trained with Backprop.

1.2 VGG and ResNet Binarized Neural Networks on CIFAR-10

We compare Backprop and Linear Backprop on the CIFAR-10 data set using two different architecture: the 19 layer VGG architecture [Simonyan and Zisserman, 2014] and a Binarized Neural Network [Courbariaux et al., 2016] using the ResNet architecture [Hubara, 2018]. For VGG architecture, we use tanh activation functions with vanilla SGD and sweep the learning rate in $\{10^{-3}, 10^{-4}, 10^{-5}\}$ and $l_2$ penalty in $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. In all experiments we use a batch size of 128 points, train with 100 epochs, and evaluate on the same test set and select the best learning rate and $l_2$ for Backprop and Linear Backprop respectively. Figure 2 shows the learning curves on the test set for the best learning rate and $l_2$ penalty for each learning rule. The tanh activation function is in particular prone to vanishing gradients since it has many saturating regions. By using Linear Backprop, however, we are able to ensure that gradients flow to continue learning.

We also consider how learning changes when the architecture is changed. For the ResNet Binarized Neural Network, we use the hard tanh activation function and the same ResNet architecture in [Courbariaux et al., 2016]. We similarly use only vanilla SGD with sweeping the learning learning rate in $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. Again, in all experiments we use a batch size of 128 points, train with 100 epochs, and evaluate on the same test set. We train this architecture with both Backprop and Linear Backprop. At 100 epochs, the best Precision@5 validation error we find using Backprop is 28.11% whereas using Linear Backprop the best Precision@5 validation error is 18.43%. These networks are significantly slower to train and in our investigation we are interested in knowing which learning algorithm is more favorable at the start of training. Although these Precision levels are far from state-of-the-art, at only 100 epochs we find that Linear Backprop is highly competitive with traditional Backprop and that further fine-tuning (longer epochs and better batch size) will yield much better results.
Acknowledgments

This work was supported in part by NSF awards CNS-1730158, ACI-1540112, ACI-1541349, and the University of California San Diego’s California Institute for Telecommunications and Information Technology. We also thank Larry Smarr, Tom DeFanti, John Graham, and Dmitry Mishin for the support and guidance provided.

References


Figure 3: **(Left)** Comparing learning CIFAR-10 using the ReLU activation function with Backprop against using the tanh activation function using Linear Backprop. Learning ReLU with Backprop can yield similar results to tanh with Linear Backprop, although it appears the variance with Backprop learning is much higher. **(Right)** Similarly, we compare learning CIFAR-10 using the ReLU activation function with Backprop against using the tanh activation function using Linear Backprop. Learning ReLU with Backprop can yield similar results to tanh with Linear Backprop, although it appears the variance with Backprop learning is much higher.


## A Comparing Linear Backprop on ReLU activation functions on CIFAR-10 with VGG19

The ReLU activation function is designed to ensure that gradients flow maximally. For this reason we compare how learning with ReLU activation functions compare with Linear Backprop. We again sweep the learning rate in \{10^{-3}, 10^{-4}, 10^{-5}\} and \ell_2\ penality in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\} for 100 epochs and select the best learning rate and \ell_2\ for all activation function and learning algorithm combinations. Figure 3 (left) shows 1000 epochs of learning comparing Linear Backprop with tanh and Backprop with ReLU.

Unlike Backprop with tanh shown in Figure 2, Backprop with ReLU is able to match learning with tanh and Linear Backprop. However, the variance of Backprop learning with ReLU is significantly higher. We see similar results when we compare Linear Backprop learning with ReLU and Backprop learning with ReLU as shown in Figure 3 in the right.
Figure 4: (Left) Comparing Backprop and Linear Backprop learning curves on CIFAR-10 test accuracy for fixed VGG architecture with varying learning rates and weight decay terms (higher is better). Light and dark color markings indicate Backprop and Linear Backprop respectively, different color indicate different weight decay rates. For large learning rate, to avoid unit saturation and stay in linear regime of the neuron, larger weight decay is required for Linear Backprop as we expect.

**B Sweeping different learning rates and weight decays for VGG19 on CIFAR-10**

In all our VGG19 experiments, we sweep the learning rate in \( \{10^{-3}, 10^{-4}, 10^{-5}\} \) and \( l_2 \) penalty in \( \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\} \) for 100 epochs. It is instructive to observe how learning progresses for these different parameter settings. In Figure 4 we show the learning curves using the tanh activation function compared with Backprop (lighter color) and Linear Backprop (darker color). In the early stages of learning, Linear Backprop in particular performs favorably compared with Backprop learning.