Deep Orthogonal Neural Networks

Ben Nosarzewski | bln@stanford.edu

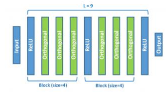


Fig 3. Diagram for alternating orthogonal/ReLU architecture. Definition of the block size B is shown.

Introduction

Very deep neural networks achieve the best performance on some of the most challenging machine learning tasks. Theoretical arguments suggest that deep networks can be more efficient than shallow networks, and intuition suggests than deep networks can learn complex high-level abstractions [1,2]. However, the exponentially vanishing gradient problem is a fundamental obstacle for training deep networks. ResNets and DenseNets successfully avoid this problem by introducing skip connections which help the gradient back-propagate through the network [3,4]. Nevertheless, recent results suggest that because of the nature of skip connections, these networks may simply be an ensemble of shallower networks, widening of these networks is more effective than deepening, and their actual depth is unclear [5,6]. In this work we propose a novel type of neural network which avoids the exponentially vanishing gradient problem by using layer weights corresponding to orthogonal matrices and a norm-preserving nonlinear operation. We find that our deep orthogonal neural networks are able to train at larger depths than standard feedforward networks.

Definitions

We design a nonlinear operation, *g*, which preserves norm for both forward and back propagation. The operation changes the signs of elements of the input according to the following:

Suppose $a, z \in \mathbb{R}^n$ and $z = g(a)$. Then define

$$z_i = \begin{cases} +a_i & \text{if } i \le n/2, \\ -a_i & \text{if } i > n/2 \text{ and } a_{i-n/2} < 0, \\ +a_i & \text{otherwise.} \end{cases}$$

We then define forward propagation through an orthogonal layer as:

$$z^{[l]} = Q^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g(z^{[l]})$$

where $Q^{[l]}$ is an orthogonal matrix.

Gradient Analysis

Let J be the softmax cross entropy cost function.

$$\begin{aligned} \mathrm{Var}(\partial J/\partial a^{[l-1]}) &= \mathrm{Var}(Q^{[l-1]} \partial J/\partial z^{[l]}) \\ &= \mathrm{Var}(\partial J/\partial z^{[l]}) \\ &= \mathrm{Var}(\partial J/\partial a^{[l]}) \end{aligned}$$

So the size of the gradients of the layer activations are perfectly constant during backpropagation!

The bias $b^{[l]}$ does not preserve the norm of the $a^{[l]}$ during forward propagation. However the resulting change in $\mathrm{Var}(a^{[l]})$ is an *additive* change and therefore does not result in exponential decay of the gradients $dJ/dO^{[l]}$ and $dJ/db^{[l]}$ (see Fig 2c).

$$\frac{\partial J}{\partial Q^{[l]}} = \frac{1}{m} \frac{\partial J}{\partial z^{[l]}} (a^{[l-1]})^T$$

$$\frac{\partial J}{\partial b^{[l]}} = \frac{1}{m} \sum_{j=1}^{m} \frac{\partial J}{\partial z^{[l](j)}}$$

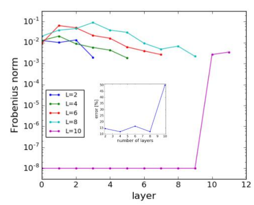


Fig 1. Plot of the gradients of the layer activations for standard feedforward neural networks of variable depths with ReLU activations and 2 hidden units trained on the dataset shown in Fig 2a. Inset shows the classification error for each of the network sizes.

Deep Orthogonal Network

Standard feedforward neural networks suffer from exponentially vanishing gradients because matrix multiplication and the ReLU activation function do not preserve norm of the activations and gradients during forward and back propagation. Fig 1 demonstrates a scenario in which the gradients exponentially vanish for a standard network with 10 or more layers. In contrast, as shown in Fig 2, a deep orthogonal network with 20 layers and the same number of hidden units is able to achieve an error of 1% when trained on the same dataset. The gradients of the activations are constant and the gradients of the neural network parameters do not exhibit exponential decay. The 1% error is better performance than that of the standard feedforward networks at any depth for the same number of hidden units which have errors greater than 10%. We note that we purposefully chose a very small number of hidden units for all networks tested in this project in order to force the networks to use their depth and not allow them to rely on width.

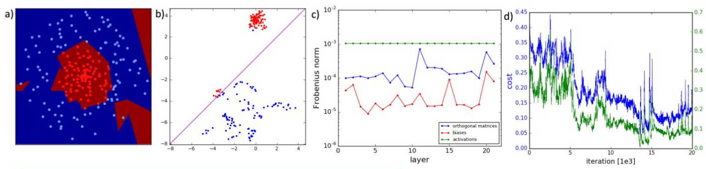
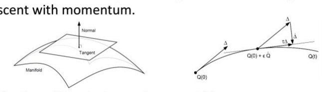


Fig 2. a) Classification boundaries after training deep orthogonal network with 20 layers and 2 hidden units in each layer. b) Activations of the last layer before softmax cross entropy. The line x=y should ideally separate the red and blue classes. c) Magnitude of the gradients for the activations, orthogonal parameters, and bias parameters. d) Cost function and error during training.

Training Algorithm

Training a deep orthogonal neural network involves performing gradient descent along the manifold of orthogonal matrices, a special case of the Stiefel manifold. We use geodesics along this manifold to update the parameters of the orthogonal matrices and also parallel transport the tangent space projection of the gradient to the new updated location on the manifold when performing each iteration of gradient descent with momentum.



Projection of Z onto tangent space at Q:

$$\pi_T(Z) = QA + (I - QQ^T)Z$$
$$A = \tfrac{1}{2}(Q^T Z - Z^T Q)$$

Geodesic equation in direction $\pi_T(Z)$ from Q:

$$Q(t) = Q(0)e^{At}$$

Parallel transport of Δ along the geodesic:

$$\Delta \equiv Q(t)B(t)$$

$$\Delta = Q(0)e^{At/2}B(0)e^{At/2}$$

Matrix exponentiation when calculating the geodesic path is in principle costly because it will take $O(n_h^3 L)$ time where L is the number of layers and n_h is the number of hidden units. However, the advantage of deep networks is that they use fewer hidden units where n_h is relatively small and so calculating the geodesic may take less time than forward and back propagation.

Alternating Orthogonal/ReLU Architecture

In Fig 4 we show the results of using an architecture which combines standard layers with ReLU activations and orthogonal layers. The types of layers alternate as shown in Fig 3. This demonstrates that orthogonal layers can be inserted into a network to help control the gradient flow during backpropagation.

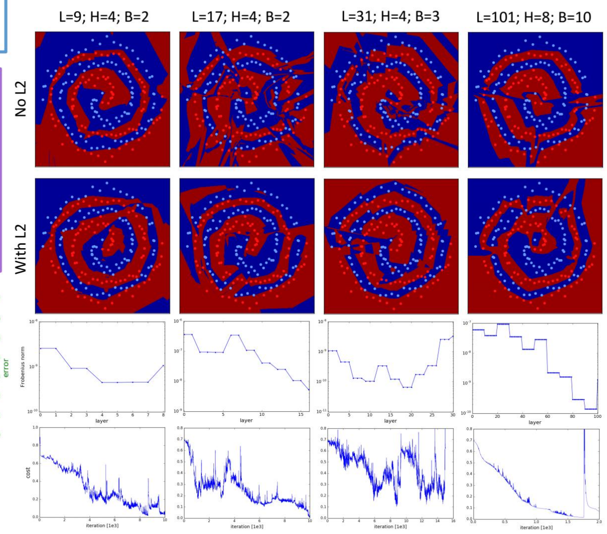
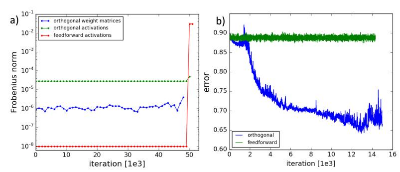


Fig 4. Classification boundaries for Orth/ReLU architectures on the spiral dataset. Gradient magnitude and cost function during training shown for those networks with L2 regularization.

MNIST

We test orthogonal networks on the MNIST dataset of handwritten digits which is a standard benchmark for various algorithms. We chose this dataset because of the small size of the images and because we were surprised to learn that the best performing single feedforward network (test error 0.35%) performs as well as the best single convolutional network (test error 0.35%) [7]. Our primary goal was not to achieve good performance on MNIST, but to demonstrate once again that orthogonal networks can be trained at larger depths than standard networks. As shown in Fig. 5, we find that standard networks cannot be trained at a depth of 50 units because of the vanishing gradient problem. In contrast, gradients of our orthogonal networks are stable at this depth and the cost function and classification error decrease during the training. We hope to achieve good performance on MNIST and other real world datasets with orthogonal networks in future work.



c)

Error	Feedforward Network	Orthogonal Network
Train	89%	64%
Validation	89%	64%
Test	89%	64%

Fig 5. a) Magnitude of the gradients for the feedforward network and orthogonal networks. Number of hidden units = 4. b) Error vs training iteration for feedforward and orthogonal networks. c) Training, validation, and test error for feedforward and orthogonal networks.

References

1. M. Nielsen, Neural Networks and Deep Learning, 2017.
2. Y. Bengio, Learning Deep Architectures for AI, Vol. 2, No. 1 (2009).
3. K. He, X. Zhang, S. Ren, and J. Sun, arXiv:151203385, 2015.
4. G. Huang, Z. Liu, L. Maaten, and K. Weinberger, arXiv:1608:06993.
5. S. Zagoruyko and N. Komodakis, arXiv: 1706.00388.
6. A. Veit, M. Wilber, and S. Belongie, CoRR, abs/1605.06431, 2016.
7. Y. LeCunn, http://yann.lecun.com/exdb/mnist/