Deep Orthogonal Neural Networks

Ben Norasowski | bnn@stanford.edu

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 that 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 gradients flow 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 an orthogonal and preserving non-linear operation. We find that our deep orthogonal neural networks are able to train at larger depths than standard feedforward networks.

Definitions

We design a non-linear operation, $\phi$, which preserves norm for both forward and back propagation. The operation changes the signs of elements of the input according to the following:

\[
\begin{align*}
    x_i &= \pm x_i & &\text{if} & & 0 < x_i < \frac{\pi}{2} \\
    x_i &= 0 & &\text{if} & & x_i \leq 0 \\
    x_i &= \pm x_i & &\text{if} & & x_i \geq \frac{\pi}{2}
\end{align*}
\]

Then define

\[
\begin{align*}
    \mathbf{Q} &= (\mathbf{Q}_1 \mathbf{Q}_2 \ldots \mathbf{Q}_n) \in \mathbb{R}^{D \times D} & &\text{where} & & \mathbf{Q}_i \in \mathbb{R}^{d_i \times d_i}.
\end{align*}
\]

We then define forward propagation through an orthogonal layer as

\[
\begin{align*}
    \mathbf{y} &= \mathbf{Q} \mathbf{x} + \mathbf{b}
\end{align*}
\]

where $Q$ is an orthogonal matrix.

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 networks with 20 layers and the same number of hidden units are 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 network 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 rely on width.

Gradient Analysis

Let $I$ be the softmax cross entropy cost function.

\[
\begin{align*}
    \nabla_c \mathcal{L} (\mathbf{W}, \mathbf{b}) &= \mathbf{Q} \nabla_c \mathcal{L} (\mathbf{W}, \mathbf{b}) \\
    &= \mathbf{Q} \nabla_c \mathcal{L}(\mathbf{W}, \mathbf{b}) \\
    &= \mathbf{Q} \nabla_c \mathcal{L}(\mathbf{W}, \mathbf{b})
\end{align*}
\]

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

The above does not preserve the norm of the $\mathbf{a}^l$ during forward propagation. However the resulting change in $\mathbf{a}^l$ is additive change and therefore does not result in exponential decay of the gradients $d\mathbf{a}^l/d\mathbf{a}^l$ (see Fig. 2).

Training Algorithm

Training a deep 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 2 onto tangent space at $Q$

\[
\pi_z(Z) = QA + (I - QQ^T)Z
\]

Geodesic equation in direction $\mathbf{u} \in \mathcal{U}$ from $Z$:

\[
\dot{\mathbf{Q}}(t) = Q(t) \mathbf{u}
\]

Parallel transport of $A$ along the geodesic:

\[
\Delta = Q(t) \mathbf{u}
\]

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

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.

Fig. 2. a) Classification errors for standard feedforward networks with 10 layers and 2 hidden units trained on the MNIST dataset. Error is 100% with 1 layer. Error drops to near 0% with 5 layers. See inset for mean classification error for each of the layers. b) Classification errors for deep orthogonal networks with 20 layers and 2 hidden units trained on the MNIST dataset. Error is 100% with 1 layer. Error drops to near 0% with 5 layers. See inset for mean classification error for each of the layers. c) Magnitude and cost function of the standard feedforward and orthogonal networks.

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

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

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

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.