Implementing weight normalization to accelerate training of deep neural networks

Abstract— In this project, we would like to try to implement and critique the weight normalization method explained in this publication- "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks" by Tim Salima tim@openai.com and Diederik P. Kingma dpkingma@openai.com". The core idea presented in this paper is to reparameterize the weight vector we use for simple backpropagation to include more parameters. The authors claim that this simple technique will help optimize the gradient descent by conditioning the minimization problem and speed up stochastic gradient descent. We would like to understand, implement, verify and try to modify the authors claims. Also, if we see that the results are promising, we would like to implement this idea and contribute it to the YANN toolbox, so that it helps others to use and evaluate this method.

Index Terms— Weight Normalization, back propagation, gradient descent.

1 Introduction

Most of the neural networks that exist today use first order gradient based optimization for learning. However, the first order gradient relies on the curvature of the objective function. If the condition number of the Hessian matrix of the objective at the optimum is low, the problem is said to exhibit pathological curvature, and first-order gradient descent will have trouble making progress. Thought neural networks differ in their architectures, the fundamental units (neurons) parameterized by weights and bias remain the same. There are several approaches that focus on conditioning the gradient of neural networks. Alternatively, we can use gradient descent without preconditioning but change the parameterization of the model.

2 Weight normalization

The output of a standard neuron can be represented by:

$$y = \emptyset\,(w.x + b) \quad (1)$$

where [w, b] represents weights and bias.
We reparameterize the weight vector in terms of a scalar g and another vector parameter v.

$$w = g\,/\,||v||\ *\ v \quad (2)$$

This reparameterization has the effect of fixing the Euclidean norm of the weight vector w. The norm of w is equal to g and is independent of vector v. By decoupling the norm of the weight vector (g) from the direction of the weight vector ($v/||v||$), we speed up convergence of our stochastic gradient descent optimization

3 Gradients

Training a neural network in the new parameterization is done using standard stochastic gradient descent methods. Here we differentiate through (2) to obtain the gradient of a loss function L with respect to the new parameters v.
r. This shows that weight normalization accomplishes two things: it scales the weight gradient by $g/||v||$, and it projects the gradient away from the current weight vector. Both effects help to bring the covariance matrix of the gradient closer to identity and benefit optimization

$$\nabla_v L = \frac{g}{\|v\|} M_w \nabla_w L, \text{ with } M_w = I - \frac{ww'}{\|w\|^2}$$

4 Experiments

This method has been tried on different datasets to like CIFAR-10, MNIST, etc and found that the training speed is increased. It has been found to work well for various scenarios like Supervised learning, generative modelling, and reinforcement learning.

5 Objectives

As a part of this project we would like to accomplish the following tasks:

1. Understand and implement the core idea presented in the paper.
2. Train the neural networks using the approach presented here and perform a comparative study to validate the experimental results.
3. Use this method to train a neural network on other datasets to see the improvements.
4. Contribute an implementation to the Yann toolbox if we see that the results are promising.

6 References

[1] Complete paper: Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks (https://arxiv.org/abs/1602.07868)
[2] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville. Recurrent batch normalization. arXiv preprint arXiv:1603.09025, 2016.
[3] CIFAR-10 dataset
https://www.cs.toronto.edu/~kriz/cifar.html