CENTER FOR
Brains
Minds+
Machines

CBMM Memo No. 148 **July 13, 2024**

For HyperBFs AGOP is a greedy approximation to gradient descent

Yulu Gan
and
Tomaso Poggio
Center for Brains, Minds, and Machines, MIT, Cambridge, MA, USA

Abstract

The Average Gradient Outer Product (AGOP) provides a novel approach to feature learning in neural networks. We applied both AGOP and Gradient Descent to learn the matrix M in the Hyper Basis Function Network (HyperBF) and observed very similar performance. We show formally that AGOP is a greedy approximation of gradient descent.

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

1 Introduction

The Average Gradient Outer Product (AGOP), defined as $\frac{1}{n}\sum_{x\in X}(\nabla f(x))(\nabla f(x))^T$, where $\nabla f(x)$ is the gradient of a predictor, was recently proposed in [1] as a foundational mathematical observation. The claim was that this framework characterizes feature learning across diverse neural network architectures and machine learning models. In the original paper [2], Radhakrishnan et al. had proposed Recursive Feature Machines (RFM) for feature learning, utilizing AGOP to update the feature matrix $M = W^T W$, in an extension of kernel machines, proposed long ago with the name of Hyper Basis Function Networks (HyperBF) [3]. M is a positive semi-definite, symmetric feature matrix providing a weighted distance for a kernel K. It is given by $K_M(x,z) = \exp(-\gamma|x-z|_M)$.

To explore ideas related to kernel methods and optimization-free learning, we utilized the HyperBF method on an image classification problem. HyperBF extends the concept of Radial Basis Function (RBF) networks by replacing the Euclidean distance measure with a Mahalanobis-like distance. Following the suggestions in [3] (see Appendix 3.1), we updated the feature matrix M using gradient descent and compared it with the Recursive Feature Machines (RFM) techniques of Belkin et al. [2], which updates M using AGOP.

In Table 1, we compared the two methods for updating M: the AGOP method and gradient descent on W as in the footnote. Despite their different update mechanisms, the empirical results of these two methods were remarkably similar. The primary distinction lies in their update techniques: "moving centers" uses GD, while "fixed centers" uses AGOP.

Table 1: **Experimental results on CIFAR10 [4] and MINIST [5].**

	CIFAR10	MNIST
	Acc ↑	
RFM	46.30	96.56
Ours (AGOP)	45.78	96.88
Ours (GD)	46.60	97.34

The comparable performance raises questions about the underlying relationship between these two techniques. In fact,we found that AGOP can be regarded as a greedy approximation of gradient descent. A detailed argument is below.

2 Proof

Let us consider a general scenario described by the following expression:

$$H(f(x)) = \frac{1}{2}(y - f(x))^2, f^*(x) = h\left(W^{(t)}x\right) \tag{1}$$

In this context, h is an activation function, and $W^{(t)}$ represents the weight matrix at time step t. The MSE is used as the objective function.

The gradient of this expression is given by:

$$\frac{\partial H(f)}{\partial W} = \nabla_W h\left(W^{(t)}x\right)\left(y - hW^{(t)}x\right)x^T \tag{2}$$

Gradient descent as in [3] can then be formulated as:

$$W^{(t+1)} = W^{(t)} + \eta\nabla_W h\left(W^{(t)}x\right)\left(y - hW^{(t)}x\right)x^T \tag{3}$$

Assume an initial condition $W^{(0)} = 0$, we have:

$$W^{(t)} = C^{(t)}x^T \tag{4}$$

Now, if we examine the gradient outer product and substitute $W^(t)$ with Eq. 4, the gradient outer product can be reformulated as:

$$
\begin{aligned}
\nabla_x f(x) \nabla_x f(x)^T &= \nabla h \left(W^{(t)} x \right) W^{(t)} \cdot \left(\left(\nabla h W^{(t)} x \right) W^{(t)} \right)^T \\
&= W^{(t)} \left(\nabla h W^{(t)} x \right)^T \left(\nabla h W^{(t)} x \right) W^{(t)} \\
&= x C^{(t)} \left(\nabla h W^{(t)} x \right)^T \left(\nabla h W^{(t)} x \right) C^{(t)} x^T \\
&= \left(x x^T \right) (\cdots) \\
&\propto x x^T
\end{aligned} \tag{5}
$$

Therefore, $\nabla_x f(x) \nabla_x f(x)^T$ is proportional to xx^T. Furthermore,

$$
\begin{aligned}
&W^{(t)^T} W^{(t)} = x C^{(t)^T} C^{(t)} x^T \propto x x^T \\
&\nabla f(x) \nabla f(x)^T \propto W^{(t))^T} W^{(t)}
\end{aligned} \tag{6}
$$

Hence, $\nabla_x f(x) \nabla_x f(x)^T$ serves as an estimator for $W^{(t)^T} W^{(t)}$.

References

[1] Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

[2] Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism of feature learning in deep fully connected networks and kernel machines that recursively learn features. *arXiv preprint arXiv:2212.13881*, 2022.

[3] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

3 Appendix

3.1 Learn M using Gradient Descent

A full description of the method can be found in the [3]. Here are the main equations:

$$f^*(\boldsymbol{x}) = \sum_{\alpha=1}^{n} c_\alpha G\left(\|\boldsymbol{x}-\boldsymbol{t}_\alpha\|^2_{\boldsymbol{W}}\right), G(\cdot) = \exp\left(-\frac{\|x-z\|^2_W}{L}\right), \|x-z\|^2_W = (x-x_i)^T W^T W (x-x_i) \tag{7}$$

$$\frac{\partial H\left[f^*\right]}{\partial c_\alpha} = -2\sum_{i=1}^{N} \Delta_i G\left(\|\boldsymbol{x}_i-\boldsymbol{t}_\alpha\|^2_W\right) \tag{8}$$

$$\frac{\partial H\left[f^*\right]}{\partial t_\alpha} = 4c_\alpha \sum_{i=1}^{N} \Delta_i G'\left(\|\boldsymbol{x}_i-\boldsymbol{t}_\alpha\|^2_{\boldsymbol{W}}\right) \boldsymbol{W}^T\boldsymbol{W}\left(\boldsymbol{x}_i-\boldsymbol{t}_\alpha\right) \tag{9}$$

$$\frac{\partial H\left[f^*\right]}{\partial \boldsymbol{W}} = -4\boldsymbol{W}\sum_{\alpha=1}^{N} c_\alpha \sum_{i=1}^{N} \Delta_i G'\left(\|\boldsymbol{x}_i-\boldsymbol{t}_\alpha\|^2_{\boldsymbol{W}}\right)(x_i-t_\alpha)(x-t_\alpha)^T, \tag{10}$$

where

$$\Delta_i = y_i - \sum_{d=1}^{N} C_\alpha \exp\left(-\frac{\left(x_i-t_\alpha\right)^\top \boldsymbol{W}^\top \boldsymbol{W}\left(x_i-t_\alpha\right)}{L}\right) \tag{11}$$

$$G' = \exp\left(-\frac{\|x_i-t_\alpha\|^2_w}{L}\right) \tag{12}$$

3.2 Recursive Feature Machine

The RFM is introduced in [2] and detailed in Algorithm 1.

Algorithm 1 Recursive Feature Machine (RFM)

Input: X, y, K_M, T ▷ Training data: (X, y), kernel function: K_M, and number of iterations: T
Output: α, M ▷ Solution to kernel regression: α, and feature matrix: M
$M = I_{d\times d}$ ▷ Initialize M to be the identity matrix
for $t = 1$ to T **do**
    $K_{\text{train}} = K_M(X, X)$ ▷ $K_M(X,X)_{i,j} := K_M(x_i, x_j)$
    $\alpha = yK^{-1}_{\text{train}}$
    $M = \frac{1}{n}\sum_{x\in X}^{\text{train}}(\nabla f(x))(\nabla f(x))^T$ ▷ $f(x) = \alpha K_M(X,x)$ with $K_M(X,x)_i := K_M(x_i, x)$
end for