Neuronal Death in Neural Networks with Group Sparsity Regularization

CS 230 Project, Final Report, Winter 2018

Nicholas Dwork (SUNet ID: 05806332)

1 Code Repository

The code for this project can be found here:

`https://github.com/ndwork/sparseNNs.git`

2 Introduction

In the brain, neurons that are deemed useless die away through a process called neuronal death [1]. This process conserves resources (e.g. space and energy consumption) while maintaining functionality. We might hope that we could include a similar process during the training of neural networks. That is, rather than retaining all the neurons in the initial structure of the network, we would like to remove neurons that are unnecessary. This process would also conserve resources (e.g., memory and computational cost required by the network) by reducing the number of nodes in each layer of the neural network. Additionally, eliminating neurons would reduce the number of parameters in the network and could prevent over-fitting the parameters to the training data.

Neuronal death can be achieved during network training by utilizing a regularization function that encourages group sparsity, where the parameters of each neuron are combined into a group. The L_2, L_1 norm is such a regularization function; when used, a neural network is trained by solving the following optimization problem:

$$\underset{w}{\text{minimize}} \;\; J(w,x) + \lambda_2 \sum_g \|w_g\|_2^2, \tag{1}$$

where J is the cost function, w_g are the parameters for the g^{th} neuron, w is a vector of all the parameters (equal to the concatenation of all the w_g vectors), and x is the training dataset [2, 3].

When J is differentiable, proximal algorithms are popular for solving problems of this type [4]. For example, problem (1) can be solved using the proximal gradient algorithm. However, this would require knowledge of the gradient of J in each iteration. As is often the case with neural networks, the size of the training dataset may be so large that computing the gradient is too computationally expensive to be feasible in every iteration. Instead, the gradient can be estimated using a mini-batch of data in each iteration of a stochastic proximal gradient optimization algorithm (SPG) [5]. The iterations of SPG for $k \in \{1, \ldots, K\}$ are

$$w^{(k+1)} = \text{prox}_{t_k L_2, L_1} \left(w^{(k)} - t_k \nabla_x J \left(w^{(k)}, x_d \right) \right),$$

where x_d is the data of the d^{th} mini-batch, t_k is the step size for the k^{th} iteration, and $\text{prox}_{t_k L_2, L_1}$ is the proximal operator of the scaled L_2, L_1 norm. By the separable sum rule of proximal operators, $\text{prox}_{t_k L_2, L_1}$ is the sum of the scaled L_2 proximal operator applied to each group x_g. That is, the proximal operator for the L_2, L_1 norm is $\text{prox}_{L_2, L_1}(w) = \sum_g \text{prox}_{L_2}(w_g)$, where

$$\text{prox}_{t_k\, L_2}(w_g) = \begin{cases} (1 - t_k / \|w_g\|_2)\, x & \text{if } \|w_g\|_2 \geq t_k \\ 0 & \text{otherwise} \end{cases}.$$

The optimization with the group sparsity regularization determines the sparsity pattern (i.e., it identifies those neurons whose parameters are all 0). Once the sparsity pattern is determined, the network is "polished". That is, the network is pruned of all dead neurons (those with all weights and bias equal to 0) and then retrained using Stochastic Gradient Descent on the unregularized problem.

3 Test Problem

The results of this project were generated with the CIFAR-10 dataset [6]. It is comprised of $60,000$ color images of size 32×32 where each image is labeled as one of ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The training set consists of $50,000$ images and the test set consists of $10,000$ images.

The neural network used for classification (shown in figure 1) consists of three convolutional layers followed by three fully connected layers. The convolutional layers are each followed by a soft-plus activation function (with a smoothing parameter of 100)[1]; after activation, the convolutional layers are each followed by a 2×2 average pooling operation. Note that each of these functions is differentiable, adhering to the requirements of SPG. The final fully connected layer does not have an activation function applied to it; instead, a softmax is applied to its output. Cross-entropy is used as the loss metric during training.

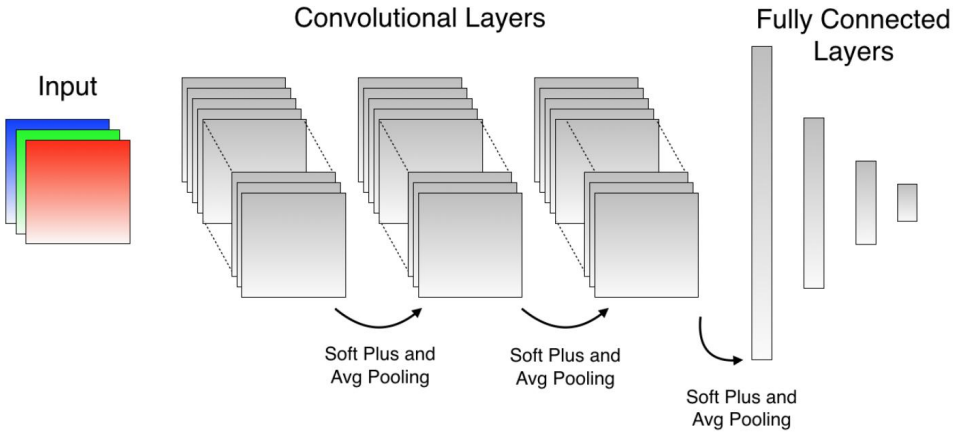

Figure 1: Classification network for CIFAR-10 dataset: three convolutional layers followed by three fully connected layers.

4 Results

Without any regularization, the network achieves 100% accuracy on the training set and 73% accuracy on the test set. With a regularization of 100, 356 of the 1310 neurons were removed from the network. The accuracy, after polishing, is 100% on the training set and 75% on the test set. This shows that group-sparsity regularization is able to eliminate neurons that are not useful for the classification process.

5 Conclusion and Future Work

This paper presents a method that is able to automatically remove neurons while maintaining functionality for the test problem described. This reduces the time required for the network to perform classification, makes it easier to implement with a hardware solution, and reduces the probability of over-fitting. The method will need to be tested on additional datasets to further validate is utility.

When imposing group sparsity, additional gains in the training time can be made by pruning as the weights become sparse [7]. That is, those neurons with weights that are 0 can be removed from the network during the training process. I leave this expansion as future work to be done after this class project's completion.

In mammals, thousands of new neurons are created in the hippocampus of an adult human every day through a process called neurogenesis [8]. Through the process of learning, only those neurons which are deemed useful are retained; the others experience neuronal death [1]. Continuing the analogy of the introduction, we might hope to include this new adaptability into the training of neural networks, perhaps in reinforcement learning algorithms. Again, I leave this expansion as future work.

[1] The soft-plus activation function is a smooth approximation to the relu activation function.

6 Acknowledgements

The author would like to thank Surag Nair and Daniel O'Connor for their guidance and many useful insights throughout this project.

References

[1] Lee J Martin. Neuronal cell death in nervous system development, disease, and injury. *International journal of molecular medicine*, 7(5):455–478, 2001.

[2] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[3] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[4] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[5] Lorenzo Rosasco, Silvia Villa, and Bang Công Vũ. Convergence of stochastic proximal gradient algorithm. *arXiv preprint arXiv:1403.5074*, 2014.

[6] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[7] Shijian Tang and Jiang Han. A pruning based method to learn both weights and connections for LSTM.

[8] TJ Shors, ML Anderson, DM Curlik II, and MS Nokia. Use it or lose it: how neurogenesis keeps the brain fit for learning. *Behavioural brain research*, 227(2):450–458, 2012.