NoiseOut: A Simple Way to Prune Neural Networks

A summary of the paper: M. Babaeizadeh et al., *NoiseOut: A Simple Way to Prune Neural Networks*, 29th Conference on Neural Information Processing Systems (NIPS 2016).

Neural networks have recently achieved state-of-the-art results using deep learning in various fields such as image recognition, natural language processing, speech recognition and more. However, neural networks are usually overparameterized, which results in an excessive usage of memory and computations. One way to deal with this issue is by pruning the network. The technique presented here works by removing excessive neurons and parameters, by merging neurons based on their correlations, while ideally maintaining the same accuracy.

Pruning a single neuron

The NoiseOut algorithm works by merging two neurons that are highly correlated. Pseudo-code for the NoiseOut algorithm can be seen in Algorithm 1. By merging the highly correlated neurons, it's possible to keep the signals in the network close to the original, thereby also maintaining the original accuracy. In the ideal case of a correlation of 1 between the neurons, the network would yield exactly the same result as before the merge. However, in the non-ideal case the accuracy might change a bit. If further training cannot compensate for a potential decrease in accuracy, the algorithm cannot compress the network further, and the algorithm is terminated.

The scheme for pruning a single neuron was not explained in the concerned paper, but as we found it substantial to the presented material, we chose to include an explanation found in some later work by same authors (see Scheme 1) [1].

procedure Train(X,Y):
    W $\leftarrow$ *initialize_weights*()
    for *each iteration* **do**
        $Y_N \leftarrow$ *generate_random_noise*()
        Y' $\leftarrow$ *concatenate*(Y, Y_N)
        W $\leftarrow$ *back_prob*(X, Y')
        while *cost(W)* $\leq$ *threshold* **do**
            A, $B \leftarrow$ *find_most_correlated_neurons*(W, X)
            $\alpha, \beta \leftarrow$ *estimate_parameters*(W, X, A, B)
            $W' \leftarrow$ *remove_neurons*(W, A)
            $W' \leftarrow$ *adjust_weights*(W', B, α, β)
            $W \leftarrow W'$;
        end
    end
    return W

Algorithm 1: NoiseOut. X is inputs, and Y is expected output.

1. For each i,j,l calculate $\rho\left(h_i^{(l)}, h_j^{(l)}\right)$
2. Find $u, v, l = \arg\max \left|\rho\left(h_i^{(l)}, h_j^{(l)}\right)\right|$
3. Calculate $\alpha, \beta := \arg\min \left(h_u^{(l)} - \alpha h_v^{(l)} - \beta\right)$
4. Remove neuron u in layer l
5. For each neuron k in layer $l+1$:
   - Update the weight $w_{v,k}^{(l)} = w_{v,k}^{(l)} + \alpha w_{u,k}^{(l)}$
   - Update the bias $b_k^{(l+1)} = b_k^{(l+1)} + \beta w_u^{(l)}$

Scheme 1: Pruning of single neuron. $h_i^{(l)}$ is the activation of the ith neuron in the lth layer, and $\rho(\cdot, \cdot)$ is the correlation.

Encouraging correlations between neurons

Correlations between the neurons is a key factor in pruning the network, although there is no guarantee that the back-propagation results in correlations in the hidden layer. This paper suggests an adjustment to the cost-function which encourages higher correlations by adding additional output nodes (*noise outputs*) before the back-propagation. The noise will differ in each iteration based on the chosen noise model. Three different noise models have been investigated, namely; Binomial, Gaussian, and Constant noise distributions,

and compared to the *non-noise* case. They were tested on a 2 layer multilayer perceptron (MLP) (2-2-1) and a 6 layer MLP (2-2-2-2-2-2-1), to see which noise model contributes to the highest correlations between the neurons. All noise models yields higher correlation than the *non-noise* case.

Experiments

The NoiseOut algorithm was implemented with Keras and applied to two well-known networks, LeNet-300-100 and LeNet-5, on the MNIST dataset.
The LeNet-300-100 network is a fully connected neural network with 300 and 100 neurons in the hidden layers. It achieves an error rate of 3.05 % on the MNIST dataset. After the pruning it achieves over 96 % reduction of the parameters, while keeping the same error rate.
The LeNet-5 network consists of two convolutional layers and one fully connected hidden layer. It achieves an error rate of 0.95 % on the MNIST dataset. It also achieves a parameter reduction of over 98 %, again while maintaining the same error rate. This is a reduction of the total number of weights in the network by a factor of 44.
For the SVHN (Street View House Numbers) dataset a deep convolutional neural network with over 1 million parameters was used. It achieved an accuracy of 93.84 % on the test set and a parameter reduction of 85.39 % for the Gaussian noise case.

Conclusion

The NoiseOut scheme for reducing parameters in neural networks, based on the correlation between neurons, is presented. It is demonstrated how the introduction of noise outputs can increase the correlation between neurons in the hidden layers, however, a further explanation of this scheme is not included. It is then shown how pruning according to this scheme can maintain high accuracy for various datasets and network architectures.

References

[1] M. Babaeizadeh et al., *A Simple yet Effective Method to Prune Dense Layers of Neural Networks* https://openreview.net/forum?id=HJIY0E9ge 4 November 2016