Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla

presented by
Chaitanya Polumetla
Overview

• Introduction
• Need for OBD
• The Idea
• Authors Proposal
• Why OBD could work?
• Experiments
• Results
• Conclusion
• Drawbacks
• References
Introduction

- Real world Neural Network problems generally have large size and are complicated
- Increase in number of parameters in system cause overfitting problem affecting the generalization performance
Need for techniques like OBD

- Networks with excessive weights over train data, as a result they have poor generalization
- Thus arises a need for a technique that reduces the size of the network without affecting validation
- Once such technique is Optimal Brain Damage
The Idea

- Reduce network complexity/size by pruning which could improve generalization.
- This can be done by removing unwanted weights from the network by freezing them i.e., setting them to ‘0’.
- Minimize cost function composed of both the training error and the measure of network complexity.
Authors Proposal

• Take a network, delete half (or more) of the weights in the network

• Wind up with a network which works either better or as well as the original net
Optimal Brain Damage

- Derive a theoretically sound technique for weight removal order using the derivative of the error function:

\[ \delta E = \frac{1}{2} \sum_i g_i \delta u_i + \frac{1}{2} \sum_i h_{ii} \delta u_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta u_i \delta u_j + O(\|\delta U\|^3) \]

- Here \( \delta u_i \) are the components of \( \delta U \)

- the \( g_i \) are the components of the gradient \( G \) of \( E \) with respect to \( U \)

- and the \( h_{ij} \) are the elements of Hessian matrix \( H \) of \( E \) with respect to \( U \)
Equation Cont..

- Assumes that Hessian matrix is diagonal
- Assumes parameter deletion will be performed after training has converged
- Assumes cost function is nearly quadratic
- Hence the equation reduces to:

\[ \delta E = \frac{1}{2} \sum_i h_{ii} \delta u_i^2 \]
Computing the 2nd Derivatives

- Network expressed as:

\[ x_i = f(a_i) \quad a_i = \sum_j w_{ij} x_j \]

- \( x_i \) is the state of unit \( i \), \( a_i \) is total input (weighted sum)
- \( f \) is the squashing function
- \( w_{ij} \) is connection going from input \( j \) to unit \( i \)
2nd Derivatives contd..

- Diagonals of Hessian:

\[ h_{kk} = \sum_{i,j} \frac{\partial^2 E}{\partial w_{ij}^2} \quad \frac{\partial^2 E}{\partial w_{ij}^2} = \frac{\partial^2 E}{\partial a_{ij}^2} x_j^2 \]

- Second Derivatives are back propagated from layer to layer:

\[ \frac{\partial^2 E}{\partial a_i^2} = f'(a_i)^2 \sum_l w_{li}^2 \frac{\partial^2 E}{\partial a_i^2} + f''(a_i) \frac{\partial E}{\partial x_i} \]

\[ \frac{\partial^2 E}{\partial a_i^2} = f'(a_i)^2 - 2(d_i - x_i) f''(a_i) \]
Goal

- To find a set of parameters whose deletion will cause the least increase in error ‘E’
- Problem is insoluble in general case since matrix H could be enormous and difficult to compute
- Hence the simplifying assumptions which reduce the equation
The Procedure

• Choose a reasonable network architecture
• Train the network until local minimum is obtained
• Compute the second derivatives for each parameter
• Compute the saliencies: $s_k = h_{kk} u_k^2 / 2$
• Delete the low-saliency parameters
Why OBD could work?

- For fixed amount of training data, networks with too many weights do not generalize well.
- On the other hand networks with too few weights do not represent the data well.
- Best case is a trade off b/w training error and network complexity.
The Data Set

• The data set used by authors for their experiments is the “Handwritten Digit Recognition” dataset

• Information gathered by United States Postal Service (USPS)

• Can be accessed at: http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
Authors Experiments

• The results shown were obtained using back-propagation applied to the handwritten digit recognition

• initial network had $10^5$ connections controlled by 2578 parameters

• database had 9300 training examples and 3350 test examples
Results for OBD vs Magnitude

- (a) Objective function vs Parameters for OBD and Magnitude based parameter deletion
- (b) Predicted and actual objective function vs number of parameters
- Predicted value is the sum of saliencies of the deleted parameters
Comparison of MSE with Retraining versus w/o Retraining

- Objective function vs number of parameters, without retraining (upper) and after retraining (lower)
- (a) training set (b) test set
- performance on training and test set (after retraining) stays almost same when up to 1500 parameters (60%) are deleted
My Implementation

- Used Multilayered Perceptron for the training with back-propagation
- Integrated with Weka
- Could not run on the whole data set due to memory limitations
- Code paralyzed by bugs
- Implementation gave similar results to MultiLayer Perceptron
- This could be due to bugs or small network
Other Notable Techniques

- Other neural net pruning technique similar to OBD is the Optimal Brain Surgeon
- No need to re-train after pruning
- No restrictive assumptions like OBD
- More effective and accurate (authors claim)
Authors Conclusions

• Decrease in the number of parameters by factor of four when OBD is used interactively

• Decrease in the number of parameters by more than two when OBD is used as automatic pruning tool

• Significant improvement in network's speed

• Slight improvement in accuracy
Possible Drawbacks of OBD

• Might remove wrong weights
• High probability that OBD would select ineffective parameters to delete in small networks
• Hessian matrix might not be diagonal at all as assumed
References


Questions?

Thank You