A Hybrid Approach to Complexity Optimization of Neural Networks

Hyunjin Lee†, Taechang Jee‡, Hyeyoung Park§, Yillbyung Lee†

The category of the paper and code:

Session 2 Neural Network Models and Algorithms

2.3 Neural network architectures

Hyunjin Lee
Laboratory for Artificial Intelligence, Department of Computer Science, Yonsei University. 134 Shinchon-dong, Soedaemun-gu, Seoul, 120-749, Korea
dryad@csai.yonsei.ac.kr
kt2751@dreamwiz.com
A Hybrid Approach to Complexity Optimization of Neural Networks

Hyunjin Lee†, Taechang Jee‡‡, Hyeyoung Park§§§, Yillbyung Lee†

† Dept. of Computer Science, University of Yonsei University
134 Shinchon-dong, Seodaemun-gu, Seoul, Korea
dryad, yblee@csl.yonsei.ac.kr
‡‡ Dept. of R&D Group, LG-EDS System
236-1 Hyoseong2-dong, Kyeong-gu, Inchon, Korea
taejee@dreamwiz.com
§§§ Lab. for Mathematical Neuroscience, Brain Science Institute, RIKEN
2-1 Hirosawa, Wako, Saitama, Japan
hypark@brain.riken.go.jp

Abstract

In this paper, we propose a hybrid approach to the optimization of complexity of feed-forward neural networks in order to achieve good generalization performances and compact structures. By applying an adaptive regularization method to the network learning, we expect to acquire a good generalization performance of the network. By applying a pruning method to the trained networks, we expect to get a compact structure as well as the good generalization performance. We confirm the performance of the proposed method through experiments on a benchmark data set.

1. Introduction

The optimization of the complexity of neural networks is a problem of significant theoretical and practical importance. From a theoretical point of view, a network model that is too simple, or too inflexible, will have a large bias, while one has too much flexibility in relation to the particular data set will have a large variance. In order to find the optimal balance between bias and variance, we need to have a way for controlling the effective complexity of neural network model. From a practical point of view, the complexity of neural networks is closely related to the number of adaptive parameters, which determines the memory and computational resources of neural networks systems.

The simplest method for the complexity optimization in practice is to compare several network models having different number of hidden units based on a criterion, and to select the best one. The criterion can be given by an estimate of the generalization performance, and there are three approaches to the estimation; the validation error approach, the algebraic generalization approach and the evidence approximation approach[11]. However, in order to compare several network models, we need to train each network, which is very computationally costly.

Alternatively, there are some stepwise selection methods such as pruning and growing. The pruning method starts with a relatively large network and prunes out the least significant connections through learning, repeatedly. On the contrary, the growing method starts with a small network and add units during the learning process, with the goal of arriving at an optimal network structure. Especially, the pruning method can leave essential parameters in the network, thus can give a compact structure and some explanation about the input-output mapping of the network[10].

On the other hand, there is somewhat indirect approach to controlling the complexity of a model by using regularization. The regularization term in an error function gives some penalty to the high complexity of a network. An important thing in this approach is the regularization parameter that determines the strength of penalty. In the adaptive regularization method, the regularization parameter is adaptively optimized with respect to the generalization performance. Therefore, the three methods used for the estimation of the generalization error mentioned above can also be used in this approach.

Larsen et. al[5][6] used the average of validation errors obtained by K-fold cross-validation to adapt the regularization parameter. Moody proposed the generalized final prediction error (GPE) and the
effective number of parameters[8], and apply them to the weight regularization. Mackay[7] explained the regularization method using the Bayesian framework, and calculated the optimal value of regularization parameter based on the Bayesian evidence.

In this paper, we propose a combination of the pruning method and the adaptive regularization method to design a compact network efficiently with good generalization performance. Our method is based on the optimal brain surgeon (OBS) with regularization term for pruning[3] and Bayesian evidence method for adaptive regularization. The proposed method is expected to improve the generalization, to decrease the amount of computation, and to facilitate interpretation.

The remainder of this paper is organized as follows. Section 2 describes the adaptive regularization method based on Bayesian evidence. Section 3 presents the pruning method for remove the least significant connections. The proposed method is presented in section 4. We also show the superiority of the proposed method through computational experiments in section 5. Conclusion and discussions for future works are made in section 6.

2. Adaptive Regularization based on Bayesian Evidence

The Technique of regularization encourages smoother network mappings by adding a penalty term $E_\alpha$ to an error function $E_D$. Thus the total cost function $C$ can be defined by

$$C = \beta E_D + \alpha E_w.$$  \hspace{0.5cm} (1)

Here $\alpha, \beta$ are the regularization parameters, which control the relative strength of the penalty term $E_\alpha$ to $E_D[2]$. One of the simplest forms of the penalty term is called weight decay that consists of sum of squares of adaptive parameters in the network[1]. In the adaptive regularization, the regularization parameter $\alpha, \beta$ is adaptively optimized with respect to the generalization performance. Even though there are various methods to measure the generalization performance, we use the Bayesian evidence for the measure.

Mackay[7] used the concept of the evidence of stochastic models to measure the generalization performance of the models. The evidence of a network model specified by a parameter vector $\mathbf{w}$ is the probability of data $D$ given the network model

$$p(D | \mathbf{w}) = \int p(D | \mathbf{w}) p(\mathbf{w}) d\mathbf{w}.$$  \hspace{0.5cm} (2)

If we assume the prior probability density function $p(\mathbf{w})$ is sufficiently flat, the evidence $p(D | \mathbf{w})$ is mainly dependent on $p(\mathbf{w} | D)$, which can be written as

$$p(\mathbf{w} | D) = \int p(\mathbf{w}, \alpha, \beta | D) d\alpha d\beta$$

$$= \frac{1}{p(D)} \int p(D | \mathbf{w}, \beta) p(\mathbf{w} | \alpha) p(\alpha) p(\beta) d\alpha d\beta.$$  \hspace{0.5cm} (3)

Under the assumption that the prior probability densities of regularization parameter $\alpha$ and $\beta$ are determined by

$$p(\alpha) = \frac{1}{\alpha}, \quad p(\beta) = \frac{1}{\beta},$$  \hspace{0.5cm} (4)

respectively, we obtain the negative logarithm of the evidence,

$$-\ln p(\mathbf{w} | D) = \frac{N}{2} \ln E_\alpha + \frac{W}{2} \ln E_w + const.$$  \hspace{0.5cm} (5)

On the other hand, we need the optimal value of $\alpha^*, \beta^*$. By approximating the probability density $p(\mathbf{w} | D)$ of Equation (3) to

$$p(\mathbf{w} | D) \approx p(\mathbf{w} | \alpha^*, \beta^*, D),$$  \hspace{0.5cm} (6)

we obtain

$$-\ln p(\mathbf{w} | D) = \beta^* E_D + \alpha^* E_w + const.$$  \hspace{0.5cm} (7)

By contrasting Equation (7) with (5), we can get a formula of the optimal regularization parameter, which can be written by

$$\alpha^* = \frac{W}{2 E_w}, \quad \beta^* = \frac{N}{2 E_\alpha}.$$  \hspace{0.5cm} (8)

This Bayesian approach has three important advantages. First, no ‘test set’ or ‘validation set’ is involved, so all available training data can be devoted to both model fitting and model comparison. Second, regularization constants can be optimized on-line. Third, the Bayesian objective function is not noisy, in contrast to the cross validation measure[1].

3. Pruning Method

Pruning algorithms is to train an oversized network and then remove the unnecessary parts. Several schemes have been proposed for pruning of the network architecture. The two most widely used schemes for pruning of feed-forward networks are Optimal Brain Damage (OBD) and OBS. Both schemes are based on weight ranking according to the saliency defined as the change in training error when the particular weight is pruned. In OBD, the
saliency is estimated as the simple deletion of a parameter, while the OBS scheme includes adapting all parameters in local quadratic approximation[1, 3, 4].

Pedersen, Hansen and Larsen[9] derived a modification of the OBS so that it can handle networks trained by using the regularized cost function. Let us define a cost function with weight decay by

$$ C(w) = E(w) + \frac{1}{2} w^T D w $$

where $E(w)$ is a mean squared error to training data sets, and $D$ is a regularization parameter matrix. The OBS saliency of a weight is defined as the change in training error as the weight is eliminated and the remaining weights retrained to the new minimum of (9). Table 1 shows the computational procedure of the saliency.

Table 1 : A Computational Procedure for Saliency

1. Expand the cost function to second order around an extremum.
2. Find the change of weights after eliminating of j'th weight, $w_j$, and retraining of the remaining weights in quadratic approximation.
3. After adjustment of weights, calculate the associated change in training error.

First, the extremum of the cost function be denoted $w_o$. The expansion of the cost function around this point leads

$$ C(w) = C(w_o) + \frac{1}{2} \delta w^T (A + D) \delta w, $$

where $w = w_o + \delta w$. The first order term vanishes by the assumed optimality of $w_o$. $A$ is the second derivative matrix of the training error; the Hessian of the cost function is $H = A + D$. Secondly, when $j$'th unit vector is eliminated, we derive the constrained extremum by using a Lagrange multiplier:

$$ \tilde{C}_j(w) = C(w) + \lambda (\delta w + w_o)^T e_j. $$

The extremum is given by $\delta w_j = -\lambda H^{-1} e_j$, with

$$ \lambda_j = \frac{w_j^T e_j}{e_j^T H^{-1} e_j}. $$

The saliency of the j'th weight is calculated by

$$ \delta E_j(w) = \lambda_j w_j^T D H^{-1} e_j + \frac{1}{2} \lambda_j^2 e_j^T H^{-1} A H^{-1} e_j, $$

where $D$ is a matrix of regularization, $\lambda$ is the Lagrange multiplier, and $A$ is the second derivative matrix of the training error. $H$ is the Hessian of the cost function with regularization.

4. Proposed Method

The proposed complexity optimization method is a combination of adaptive regularization method and OBS pruning method, which are described above. In this section, we describe the procedures required for the proposed Bayesian optimization of the regularization parameters and the generalized OBS pruning. Figure 1 shows an overall structure of the proposed method.

![Figure 1. Overall Structure of the Proposed Method](image)

Table 2 shows the detail procedure of the proposed methods.

Table 2 : The Proposed Method to Optimization of Complexity of Neural Network

1. Initialize objective function parameters $\alpha, \beta$ and the weights $w$.
2. Take one step of the Levenberg-Marquardt algorithm to minimize the objective function

$$ C(w) = \beta E_0 + \alpha E_w. $$

$E_0$ is the sum of the squared errors, $E_w$ is the sum of squares of network weights.
3. Compute new estimates for the objective function parameters

$$ \alpha = \frac{W}{2 E_w}, \quad \beta = \frac{N}{2E_0}. $$
W is the number of weight and N is the number of training data.
4. The iterate steps 2 through 3 until convergence.
5. The saliency for weight $j$ is defined by

$$\delta E_j(w) = \frac{\alpha}{\beta} \lambda_j w_j^T H^{-1} e_j + \frac{1}{2} \lambda_j^2 e_j^T H^{-1} A H^{-1} e_j$$  \hspace{1cm} (16)$$

where $\lambda$ is Lagrange multiplier and $H$ is the Hessian of the regularize criterion.
6. Eliminate a weight parameter which has minimal saliency given by (16). Compute new estimates for the objective function parameters (15).
If error is larger than a pre-defined threshold value then go back to step 2, else go back to step 5. This repeats until proposed error is matched, and also the architecture is simpler.

5. Experimental Results

We apply our algorithm to MONK problem which is a well-known benchmark problem. It is an artificial robot domain, in which robots are described by six different attributes[12]:

- $x_1$ head_shape \( \subseteq \) round, square, octagon;
- $x_2$ body_shape \( \subseteq \) round, square, octagon;
- $x_3$ is_smiling \( \subseteq \) yes, no;
- $x_4$ holding \( \subseteq \) sword, balloon, flag;
- $x_5$ jacket_color \( \subseteq \) red, yellow, green, blue;
- $x_6$ has_tie \( \subseteq \) yes, no.

The learning tasks of the three MONK’s problems are of binary classification, each of them being given by the following logical description of a class.

- **Problem M1:**
  (head_shape=body_shape)or (jacket_color=red).
  From 432 possible examples, 124 were randomly selected for the training set.

- **Problem M2:**
  Exactly two of the six attributes have their first value.
  From 432 examples, 169 were selected randomly.

- **Problem M3:**
  (Jacket_color is green and holding a sword) or (jacket_color is not blue and body_shape is not octagon).
  From 432 examples, 122 were selected randomly.

In the experiment, 10 fully connected neural networks used as the starting networks. Each of these networks consists of 17 input units, 5 hidden units, 1 output unit. Each hidden units and output unit have one bias. Thus, the total number of weight of a network is 96.

We compared the proposed method with the simple weight decay method. The Table 3 shows the performance of the two methods. It is confirmed that the proposed method gives better accuracy and more compact network than weight decay method. On MONK1 and MONK2 data, we obtained more compact networks by the proposed method. On MONK3 data, OBSWD has 100% recognition for training data but couldn’t reach 100% recognition for test data. But, proposed method has 100% recognition to test data.

**Table 3 : The Accuracy and Number of Weights Determined by OBS with Weight Decay(OBSWD) and by Proposed Hybrid Approach(HA) on three MONK’s Problems.**

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (%)</th>
<th># of connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONK1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSWD</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>HA</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>MONK2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSWD</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>HA</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>MONK3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSWD</td>
<td>100</td>
<td>94.68</td>
</tr>
<tr>
<td>HA</td>
<td>96.47</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 2-(a) shows a network structure obtained by the proposed method for the MONK 1 data. The number of connections is reduced into 12 from 96. By classification rule, it is well known that the network’s node 1, 2, 3, 4, 5, 6 and 12 is most significant. In Figure 2-(a), it is shown that the network retain essential connections with 100% accuracy rate to training and test data sets.

Figure 2-(b) shows a networks structure obtained by the proposed method for the MONK2 data. The number of connections is reduced into 15 from 96. The proposed method can retain essential connections of network with 100% accuracy rate to training and test data sets.

Figure 2-(c) shows a networks structure for the MONK 3 data. The number of connections is reduced into 11 from 96. The accuracy of network is 95.08% to training data and 100% to test data. The classification rule for the MONK 3 data cannot be represented by networks node exactly. However, it can be roughly told that proposed method retains essential connections of network.
method for optimizing the complexity of neural networks. By combining two different approaches, we can obtain a network with good generalization ability and compact structure. This work is a preliminary study on a total framework of the complexity optimization for neural networks, which has computational efficiency as well as good optimizing performance. We expect to improve the optimizing performance through combination of various methods with different characteristics, and this paper shows an example of combination. As a future works, we need to sophisticate and improve each optimization method, and develop a more computationally efficient combination method.

7. References


6. Conclusions and Future Works

In this paper, we proposed a hybrid method of the adaptive regularization method and the pruning