1-1-2009

A neural network pruning approach based on compressive sampling

Abdesselam Bouzerdoum
University of Wollongong, bouzer@uow.edu.au

Son Lam Phung
University of Wollongong, phung@uow.edu.au

Jie Yang
University of Wollongong, jy962@uow.edu.au

A neural network pruning approach based on compressive sampling

Abstract

The balance between computational complexity and the architecture bottlenecks the development of Neural Networks (NNs), An architecture that is too large or too small will influence the performance to a large extent in terms of generalization and computational cost. In the past, saliency analysis has been employed to determine the most suitable structure, however, it is time-consuming and the performance is not robust. In this paper, a family of new algorithms for pruning elements (weighs and hidden neurons) in Neural Networks is presented based on Compressive Sampling (CS) theory. The proposed framework makes it possible to locate the significant elements, and hence find a sparse structure, without computing their saliency. Experiment results are presented which demonstrate the effectiveness of the proposed approach.

Keywords

neural, sampling, compressive, approach, pruning, network

Disciplines

Physical Sciences and Mathematics

Publication Details

Yang, J., Bouzerdoum, A. & Phung, S. (2009). A neural network pruning approach based on compressive sampling. Proceedings of International Joint Conference on Neural Networks 2009 (pp. 3428-3435). New Jersey, USA: IEEE.

A Neural Network Pruning Approach based on Compressive Sampling

Jie Yang, Abdesselam Bouzerdoum, Son Lam Phung

Abstract— **The balance between computational complexity and the architecture bottlenecks the development of Neural Networks (NNs). An architecture that is too large or too small will influence the performance to a large extent in terms of generalization and computational cost. In the past, saliency analysis has been employed to determine the most suitable structure, however, it is time-consuming and the performance is not robust. In this paper, a family of new algorithms for pruning elements (weighs and hidden neurons) in Neural Networks is presented based on Compressive Sampling (CS) theory. The proposed framework makes it possible to locate the significant elements, and hence find a sparse structure, without computing their saliency. Experiment results are presented which demonstrate the effectiveness of the proposed approach.**

NOMENCLATURE

γ	Bias vector between hidden and output layer
θ	Bias vector between input and hidden layer
A	Network targets
B	Output from the reduced network
C	Input matrix for the hidden layer
d_1	Number of input dimensions
d_2	Number of output dimensions
N_{hn}	Number of initial hidden neurons
N_{rn}	Number of remaining neurons after pruning
N_s	Number of training samples
P	Network inputs
V	Original weight matrix between hidden and output layer
W	Original weight matrix between input and hidden layer
Z	Output from the original network

I. INTRODUCTION

HOW to design an exact topology for Neural Networks (NNs) is one of the most important issues faced by researchers over the past two decades [1]–[7],[18]–[19]. The problem seems to be an awkward predicament in that it is difficult to find the balance between the computational complexity and the generalization ability of a network. A too large network architecture may result in poor generalization on the test data even if it can obtain high accuracy on the training data. On the other hand, a too small structure requires more training time to converge to a local minimum, which may not yield satisfactory performance. A variety of approaches have been developed to deal with this problem, which can be broadly categorized as follows:

- Network Pruning, which is based on the saliency analysis on each element (weights or hidden neurons) [2]–[7];
- Network Construction, which begins with a small network and incrementally adds hidden neurons during the training process [18];
- Other algorithms such as Evolutionary Pruning [19].

In this paper we are concerned with pruning methods; some traditional pruning algorithms are discussed briefly in Section II. Generally speaking, Network Pruning is often cast as three sub-procedures: (i) define and quantify the saliency for each element in the network; (ii) eliminate the least significant elements; (iii) re-adjust the remaining topology. To this end, the following questions may be raised:

1) What is the best criterion to describe the saliency, or significance of elements?
2) How to eliminate those unimportant elements with minimal increase in error?
3) How to make the method converge as fast as possible?

To overcome these difficulties, we offer another insight into Network Pruning in this paper. Our approach is inspired by *Compressive Sampling* (also known as Compressed Sensing) theory, which addresses sparse signal representation [8]–[10]. The method can help in recovering signals that have a sparse representation from a number of measurements/projections of dimensionality lower than the number of samples required by the Shannon/Nyquist Sampling theory. Thus, if we consider the topology of a neural network as a sparse structure, then CS can be employed to reconstruct this topology.

In this paper, we propose a family of new algorithms to obtain the optimal topology for pruning weights and hidden neurons based on CS. The main advantage of our approach is that the proposed algorithms are capable of iteratively building up the sparse topology, while maintaining the training accuracy of the original larger architecture. The remainder of the paper is organized as follows. The next section presents a brief review of traditional pruning algorithms and Compressive Sampling theory. Then Section III details the development of the new network pruning approach. By describing the link between pruning NNs and CS and introducing two definitions for different sparse matrices, the algorithms for pruning weights and hidden neurons are given in Section III-B and C, respectively. The

Jie Yang, Abdesselam Bouzerdoum, and Son Lam Phung are with the School of Electrical, Computer and Telecommunications Engineering, and ICT Research Institute, University of Wollongong, Wollongong, Australia (email: {jy962, a.bouzerdoum, phung}@uow.edu.au).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

implementation issues and experimental results are discussed in Section IV, followed by concluding remarks.

II. Background

This section introduces basic notions about pruning algorithms and Compressive Sampling theory, which will be used in the remainder of the paper.

A. Traditional Algorithms for Pruning NNs

The general framework for Network Pruning can be described as follows:

1) Set a large enough architecture for the NNs and train with any learning method (e.g. back propagation algorithm), until the stopping criterion is met;
2) Compute the saliency of each element and eliminate the least important ones;
3) Retrain the pruned network. If the change of output between the original and pruned network is small enough, then go to step 2; otherwise stop and output the network architecture.

Roughly, the methods for pruning can be classified into two categories: weight pruning and hidden neuron pruning. Examples of weight pruning algorithms include Optimal Brain Damage (OBD) [2], Optimal Brain Surgeon (OBS) [3], and Magnitude-based pruning (MAG) [5]. On the other hand, Skeletonization (SKEL) [6], non-contributing units (NC) [7] and Extended Fourier Amplitude Sensitivity Test (EFAST) [14] all implement hidden neuron pruning.

B. Compressive Sampling

Recently, a siginificant research effort has been devoted to the problem of *Compressive Sampling* (CS) [8]–[10]. This theory supposes that if we allow for a degree of residual error ϵ, CS guarantees the success of recovering the given signal under some conditions from a number of projections (measurement vectors). We refer the reader to [8]–[10] for a more detailed discussion on CS and its wide applications.

According to the number of measurement vectors (also known as measurement samples in CS), the CS problem can be categorized into *Single-Measurement Vector* (SMV) [11] or *Multiple-Measurement Vector* (MMV) [12], [13]. Mathematically, the SMV problem is expressed as follows. Given a measurement sample $y \in R^m$ and a dictionary $D \in R^{m \times n}$ (the columns of D are referred to as the atoms), we seek a vector solution satisfying:

$$(P): min \|x\|_0 \quad s.t. \;\; y = Dx \tag{1}$$

where $\|x\|_0$ (known as l^0-norm) is the cardinality of x, or the number of nonzero elements in x. Several alternative algorithms or strategies have been developed including Greedy Algorithms (such as Orthogonal Matching Pursuit (OMP) [11] or Matching Pursuit (MP) [20]) and Non-convex local optimization like FOCUSS [21] algorithm. In this paper, we are concerned with the Greedy Algorithms, whose convergence is demonstrated by the following theorem [17]:

Theorem 1: There exists a time function $\beta(t) \in (0,1)$, which depends only on the dictionary, for any sample y, such that the residual error calculated with the OMP or MP algorithms decays as:

$$\left\|r^t\right\|^2 \leq \beta(t) \left\|r^{t-1}\right\|^2 \tag{2}$$

where $r^t = y - Dx^t$ denotes the reconstruction error after t iterations. More precisely, we have:

$$\left\|r^t\right\|^2 \leq \beta(t-1) \left\|r^{t-1}\right\|^2 \leq ... \leq \beta^t(0) \left\|r^0\right\|^2 = \beta^t(0) \|y\|^2 \tag{3}$$

Furthermore, for the OMP algorithm we have the following theorem [11]:

Theorem 2: Given an arbitrary d-sparse signal $x \in R^n$, $n \geq d$ and a random $m \times n$ linearly independent matrix D. OMP can represent x with probability exceeding $1 - \delta$ when the following condition is satisfied: $m \geq Kd \log(n/\delta)$, where K is an absolute constant, and $\delta \in (0, 0.36)$.

While, the SMV problem aims to find a sparse signal representation, the MMV problem aims to find a joint-sparse representation of several signals, or a sparse matrix representation [13]. The MMV can be stated as follows (to avoid confusion, we use the upper-case letters Y and X to emphasize that they are matrices rather than vectors):

$$(Q): min \|m(X)\| \quad s.t. \;\; Y = DX \tag{4}$$

where $Y \in R^{m \times k}$, $X \in R^{n \times k}$ and $m(X)$ is the matrix norm.

III. Problem Formulation and Methodology

A. Overview of Methods

In this section, we formulate the problem of Network Pruning as a Compressive Sampling problem. Before we explain our main idea, it is necessary to introduce some basic definitions.

Definition 1: (***S-sparse (I) Matrix***) given an arbitrary matrix $X \in R^{n \times k}$, if $\|x_i\|_0 \leq S$ where x_i is any column in X, then X is called an ***S-sparse (I) Matrix***, which we denote as $S1(X)$.

Definition 2: (***S-sparse (II) Matrix***) given an arbitrary matrix $X \in R^{n \times k}$, if $\|X\|_0 \leq S$, where $\|X\|_0 = \left\|\left(\|x_1\|_0, \|x_2\|_0, \|x_3\|_0, \ldots, \|x_n\|_0\right)^T\right\|_0$ is the number of rows that contain nonzero elements, then X is called an ***S-sparse (II) Matrix***, which we denote as $S2(X)$.

Without loss of generality, consider a trained two-layer feed-forward network. Given a set of training input patterns, which are stored in a matrix P, and the desired output patterns, stored in a matrix A, then the mathematical model for training the NN can be expressed in the form of the following expansion:

$$min \|A - Z\| \, s.t. \begin{cases} C = f_1(WP + \theta) \\ Z = f_2(VC + \gamma) \end{cases} \tag{5}$$

where Z denotes the network output matrix, C the hidden unit output matrix, W is the hidden unit weight matrix, V is the output layer weight matrix, f_1 and f_2 denote the activation functions, and θ and γ are the bias terms. Our

aim is to find a minimal topology in terms of the number of weights or hidden neurons while minimizing the increase in error given by $\|A - Z\|$. When pruning weights from W or V, the behavior acts like setting, from a mathematical viewpoint, the relating elements in W or V to 0. Then the goal of finding the smallest number of weights in NNs within a range of accuracy can be equated to finding an ***S-sparse (I) Matrix*** W or V. Therefore, the problem of pruning weights can be cast as follows:

$$\begin{cases} min_W[S1(W)] & s.t.\ C = f_1(WP + \theta) \\ min_V[S1(V)] & s.t.\ Z = f_2(V(f_1(WP + \theta)) + \gamma) \end{cases} \tag{6}$$

On the other hand, pruning hidden neurons can be considered as a special case of pruning weights in that if all weights from a certain hidden neuron are pruned, the neuron would be removed as well. This process is equivalent to finding a V with most of its rows are zero, i.e. an ***S-sparse (II) Matrix*** V. Consequently, We define the following optimization problem:

$$min_V[S2(V)] \quad s.t.\ Z = f_2(V(f_1(WP + \theta)) + \gamma) \tag{7}$$

Note that $WP + \theta = [W, \theta][P, 1]^T = \tilde{W}\tilde{P}$, and $[V, \gamma][(f_1(WP + \theta), 1]^T = \tilde{V}\tilde{C}$, so the problem in (6) can be rewritten as:

$$\begin{cases} min_{\tilde{W}}[S1(\tilde{W})] & s.t.\ [f_1^{-1}(C)]^T = (\tilde{P})^T(\tilde{W})^T \\ min_{\tilde{V}}[S1(\tilde{V})] & s.t.\ [f_2^{-1}(Z)]^T = (\tilde{C^*})^T(\tilde{V})^T \end{cases} \tag{8}$$

Where $C^* = f_1(\tilde{W}\tilde{P})$ denotes the actual input matrix of the hidden layer for the pruned network. Similarly, Equation (7) for pruning hidden neurons is changed to:

$$min_{\tilde{V}}[S2(\tilde{V})] \quad s.t.\ [f_2^{-1}(Z)]^T = (\tilde{C})^T(\tilde{V})^T \tag{9}$$

Comparing the modified optimization problem with (4), we can see here $(\tilde{P})^T$, $(\tilde{C})^T$ and $(\tilde{C^*})^T$ serve as the ditionary D and $[f_1^{-1}(C)]$ and $[f_2^{-1}(Z)]$ play the role of the signal matrix Y in (4). Note that one can replace Z with A in (8) or (9) since Z is very close to A after training the original network; the experimental results in Section $IV-C$ show similar performance between the two methods. In doing so, the process of pruning NNs can be regarded as finding different sparse solutions for weight matrix W or V. Fig. 1 illustrates the difference between S-sparse (I) and S-sparse (II) network topologies.

B. Pruning Weights

Obviously, the problem of finding an ***S-sparse (I) Matrix*** can simply be calculated from the sparse representations for Multi-SMVs simultaneously. This algorithm, named $SMVSI$ in our paper, is summarized in Table I.

Without loss of generality and simplicity, we focus on the OMP [11] algorithm as the SMV method in $SMVSI$. Consequently, the procedure for pruning weights based on Algorithm 1 is given in Table II, which we denote as CSP1 (Compressive Sampling based Pruning 1):

Remark 1: An important question that arises is "will the CSP1 algorithm converge?" Here, what we need to point out

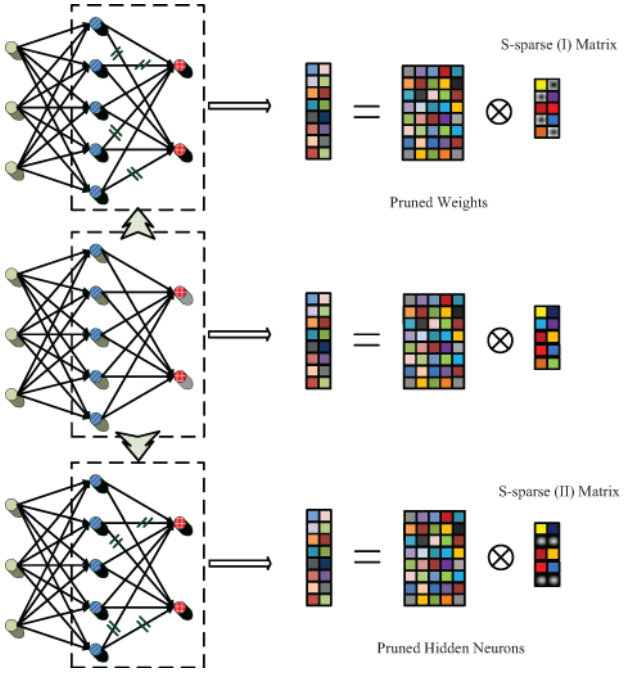

Fig. 1. Pruned NNs with S-sparse (I) and S-sparse (II) weight Matrices

TABLE I

ALGORITHM 1: $SMVSI$ ALGORITHM FOR ***S-sparse (I) Matrix***

Input: the signal matrix $Y \in R^{m \times k}$, the dictionary $D \in R^{m \times n}$, Maximal iteration M
Output: an $n \times k$ matrix X
Procedure:
1): For $y_i \in Y$, $i \in [1, k]$ execute any traditional SMV method using (y_i,D) with M iterations and obtain x_i;
2): Replace the i^{th} column of X by x_i.

is that the convergence of CSP1 is only influenced by the number of samples and orthogonality of data. We take the process of calculating $\tilde{W}$ as example. Firstly, we note that the CSP1 algorithm is a procedure for rebuilding each column $[(\tilde{W})^T]_i \in [(\tilde{W})^T]$ using OMP:

$$min \left\| [(\tilde{W})^T]_i \right\|_0 \quad s.t.\ [[f_1^{-1}(C)]^T]_i = (\tilde{P})^T[(\tilde{W})^T]_i \tag{10}$$

where $[[f_1^{-1}(C)]^T]_i$ denote the corresponding ith column in matrix $[[f_1^{-1}(C)]^T]$. Thus, assuming that the columns of $(\tilde{P})^T$ (the dictionary) are linearly independent and K is a positive constant. According to $Theorem$ 2, suppose that $\left\| [(\tilde{W})^T]_i \right\|_0 \leq d$ when it satisfies $N_s \geq Kd\log(N_{hn}/\delta)$, after d iterations, CSP1 is guaranteed to find the sparsest solution in each column $[(\tilde{W})^T]_i \in [(\tilde{W})^T]$ with probability exceeding $1 - \delta$. Therefore, convergence of CSP1 is guaranteed. Unfortunately, the above claim is built on the success of pursuit algorithms, depending on the number of samples and orthogonality of data, and thus convergence is not always guaranteed. However, according to $Theorem$ 1, the OMP method, with the residual error decreasing, is still known to perform very well [11].

Remark 2: The computational complexity of the above

TABLE II

ALGORITHM 2: CSP1

Input: the trained network and Maximal iteration $M1$ and $M2$.
Output: a pruned network with $\tilde{W}$ and $\tilde{V}$.
Procedure:
1): Compute $C = f_1(WP + \theta)$;
2): Find $\tilde{W}$ by $(\tilde{W})^T = SMVSI([f_1^{-1}(C)]^T, (\tilde{P})^T, M1)$;
3): Compute $(\tilde{V})^T = SMVSI([f_2^{-1}(Z)]^T, (\tilde{C^*})^T, M2)$.

algorithm depends on the number of weights (how large the original network is) as well as the remaining weights. Table III summarizes the computational complexity for all algorithms in terms of the number of floating operations (flops). Note that the existing methods including MAG, OBD to OBS only remove one element (weight) in one pruning iteration, which is costly in term of computation. On the contrary, CSP1 focuses on the remaining weights rather than the eliminated ones, which speeds up the pruning process, as can be seen from Table III (where N_s, N_{hn}, and N_{rn} denote the number of training samples, initial hidden neurons and remaining neurons after pruning, respectively).

TABLE III

COMPARISON OF COMPUTATIONAL COMPLEXITY OF DIFFERENT PRUNING ALGORITHMS.

Algorithm	Computation Cost	Flops
MAG	$O(N_{hn}N_s)$	$O(N_{hn} - N_{rn})$
OBD	$O(N_{hn}N_s)$	$O(N_{hn} - N_{rn})$
OBS	$O(N_{hn}^2 N_s)$	$O(N_{hn} - N_{rn})$
$CSP1$	$O(N_{hn}log(N_s)) - O(N_{hn}N_s)$	$O(N_{rn})$

C. Pruning Hidden Neurons

It is worth noting that we cannot apply CSP1 algorithms directly to calculate an ***S-sparse (II) Matrix***. Fortunately, methods used in SMV have already been extended to solve this problem (we refer to them as $MMVSII$ algorithm in this paper) including MMV (Orthogonal) Matching Pursuit and (Regularized) M-FOCUSS Algorithm [12], [13]. In particular, we propose two important properties of the above pursuit methods by showing the sparse solution as well as their convergence.

Theorem 3: If X is a matrix solution obtained by any $MMVSII$ pursuit method, then X is an ***S-sparse (II) Matrix***.

Proof: As for the MMV (Orthogonal) Matching Pursuit, the algorithms select only one atom from the dictionary which has the largest inner product value [12], [13] with the residual error at each iteration. Then it is obvious that the solution is a ***T-sparse (II) Matrix*** at Tth iteration. On the other hand, the (Regularized) M-FOCUSS Algorithm provides a solution that minimizes $\|Y - DX\|$ as well as $(\sum x_{ij}^2)^{1/2}$, with x_{ij} denoting the element in the ith row and the jth column in X. It can be shown that the result is necessarily sparse, which could be extended easily from [21]. ■

Theorem 4: For all of the above $MMVSII$ pursuit methods, the objective function $\|Y - DX\|$ decreases, i.e., $\left\|Y - D(X)^{t+1}\right\| = k\left\|Y - D(X)^t\right\|$, where $k \in (0, 1)$ is a constant [12].

Since the comparison between the above algorithms is done in [12], here we are concerned with the M-FOCUSS algorithm for its efficiency. We list the framework for pruning neurons based on the M-FOCUSS algorithm in Table IV (interested readers can easily implement this algorithm using any $MMVSII$ pursuit method by simply replacing the M-FOCUSS algorithm):

TABLE IV

ALGORITHM 3: CSP2 (COMPRESSIVE SAMPLING BASED PRUNING 2)

Input: the trained network and the parameter δ.
Output: a pruned network with $\tilde{V}$.
Procedure:
Compute $(\tilde{V})^T = M - FOCUSS([f_2^{-1}(Z)]^T, (\widetilde{C})^T, \delta)$, where δ is the minimal value used to stop iterations.

Remark 3: As for the convergence of the CSP2 algorithm, it is guaranteed by $Theorem$ 4. In particular, [13] proves that M-FOUCSS can recover any matrix using $(\mu^{-1} + 1)/(1 + \sqrt{k})$ atoms at most, where μ only depends on the dictionary and k is the number of columns in the dictionary. This means that the optimal upper bound for the remaining number of neurons would be less than $(\mu^{-1} + 1)/(1 + \sqrt{d_2})$.

Remark 4: If we use CSP2 between the input and hidden layers, the process is transferred into a feature selection for input data. In other words, we have the following optimization problem:

$$min_{\tilde{W}}[S2(\tilde{W})] \quad s.t.\ [f_1^{-1}(C)]^T = (\tilde{P})^T(\tilde{W})^T \qquad (11)$$

We leave this for the future development.

IV. EXPERIMENT RESULTS AND ANALYSIS

Experiment are conducted using two different benchmark problems taken from from Proben1 [4]: the Cancer1 classification problem and the Flare1 function approximation problem. The Cancer1 problem is created based on the diagnosis of breast cancer to classify a tumor as either benign or malignant. Flare1 is the prediction of solar flares that will occur during the next 24 hours. Table V below presents the two data sets.

TABLE V

DATASET FOR EXPERIMENTS

	Description		
	Types	Training examples	Test examples
Cancer1	Classification	350	175
Flare1	Approximation	533	267

A. Experiment Results

In this paper, we compare our algorithms with those of Neural Network Simulator (SNNS), a simulator for NNs which can be downloaded from *http://www.ra.cs.uni-tuebingen.de/SNNS/*. All the traditional algorithms, such as MAG, OBS, OBD, NC, SKETL, are available in SNNS.

The CSP1 method introduces two parameters, $M1$ and $M2$, corresponding to 2 sub-procedures as shown in Table II; Since d_1 (the dimension for input data) and N_{hn} (the maximum number of hidden units) are the row dimensions of $(\tilde{W})^T$ and $(\tilde{V})^T$ in (8), respectively, these two parameters have the following scale:

$$M1 \leq d_1, \quad M2 \leq N_{hn}; \tag{12}$$

otherwise, the solutions will not be sparse any more. Consequently, we set the parameters in our CSP1 algorithm as follows: $M1 = [d_1/4, d_1/3, d_1/2, 2d_1/3]$ and $M2 = [N_{hn}/10, N_{hn}/8, N_{hn}/6, N_{hn}/4]$. In CSP2 algorithm, there is only one user set parameter δ controlling the stopping criterion. Generally speaking, the smaller δ we set, the sparser the solution. However, it is a time-consuming process to compute the solution for a small δ. Thus, we set the parameter group as $\delta = [e^{-5}, e^{-10}, e^{-15}]$.

In our experiments, we test for all of the above different parameter combinations for CSP1 and CSP2. In other words, one whole process consists of 4×4 loops in CSP1 and 3 loops in CSP2, respectively, and we take the best result as the output. Furthermore, all the following experiments begin with a fully-connected 2-layer network with 128 hidden neurons. We run all the algorithms 20 times to collect the statistics shown in Tables VI and VII.

TABLE VI
COMPARISONS FOR DIFFERENT ALGORITHMS ON CANCER1 PROBLEM

Traing Epochs=500	MAG	OBS	OBD	CSP1
Weight	1236	1236	1245	666
MSE	0.01442	0.01846	0.03680	0.0172
Time(s)	1.41	24.49	39.68	0.145
Traing Epochs=500	NC	SKETL	EFAST	CSP2
Hidden Neurons	104	107	6	9
MSE	0.03136	0.03406	0.03129	0.0174
Time(s)	180.31	25.76	13.68	6.31

TABLE VII
COMPARISONS FOR DIFFERENT ALGORITHMS ON FLARE1 PROBLEM

Traing Epochs=50	MAG	OBS	OBD	CSP1
Weight	3454	3245	2359	807
MSE	0.06213	0.07123	0.05856	0.0036
Time(s)	0.53	86.14	21.56	0.219
Traing Epochs=50	NC	SKETL	EFAST	CSP2
Hidden Neurons	127	126	6	5
MSE	0.07965	0.07725	0.01638	0.0034
Time(s)	33.57	6.68	9.12	13.58

Fig. 2 presents the Receiver Operating Characteristic (ROC) curves of the CSP1 and CSP2 algorithms for the classification problem.

From these data, the following observations can be made:

1) Both CSP1 and CSP2 achieve test errors comparable with that of the other algorithms, as well as the reduced network topology and computational cost. For instance, in the cancer1 and flare1 problems, CSP1 produces a network less than half the size when compared with its counterparts;

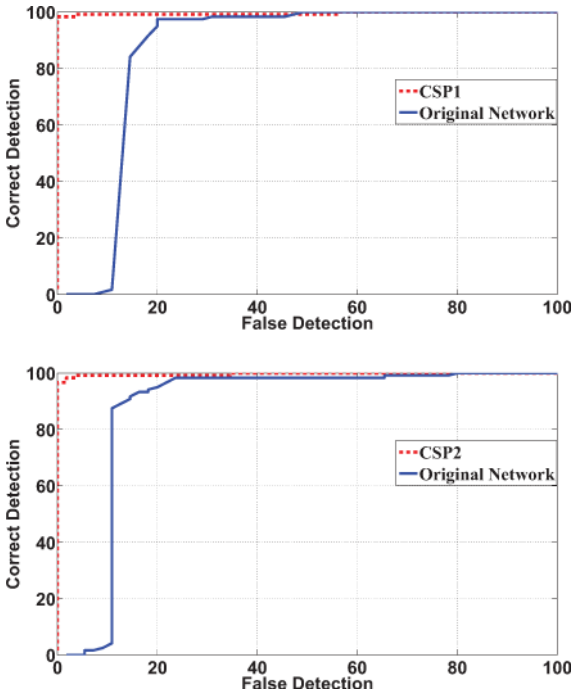

Fig. 2. Detection Rate vs. False Alarm Rate

2) Although CSP2 obtains more hidden units than EFAST after training with the cancer1 problem, its MSE and computation time is significantly better;
3) The ROC curves in Fig. 2 indicate clearly that the generalization performances of the pruned network using the presented algorithms are better than the original network. In fact, CSP1 and CSP2 lead to 99.26% and 98.29% total classification accuracy, respectively.

B. Selection of Algorithm Parameters

In this section we discuss the parameters impacting the performance of CSP1 and CSP2, based on the Cancer1 problem. In CSP1, 20 trials are run with the above configurations; Fig. 3 show the different results. We first note that the greater a parameter (either d_1 or N_{hn}), the larger the structure of the remaining network, and the longer the execution time. Furthermore, CSP1 achieves worst results when $M_2 = N_{hn}/4$ compared with its counterparts. It is expected that the CSP1 algorithm, having a larger topology, will be trapped in local minima as the training data is over-fitted. In addition, the minimal network we obtain only contains 282 weights, however, its error on test data is about 0.02283; on the other hand the minimal error is 0.017172 but the corresponding number of weights is over 666. So user can select different coefficients according to the specific purpose.

Fig. 4 presents the results obtained by CSP2. From these results we note that the smaller the parameter δ, the higher the probability that we can obtain a better test error. This can be explained by the fact that the CSP2 algorithm, for small parameter, may have a more sparse solution, and hence achieves a smaller topology. However, we also notice that

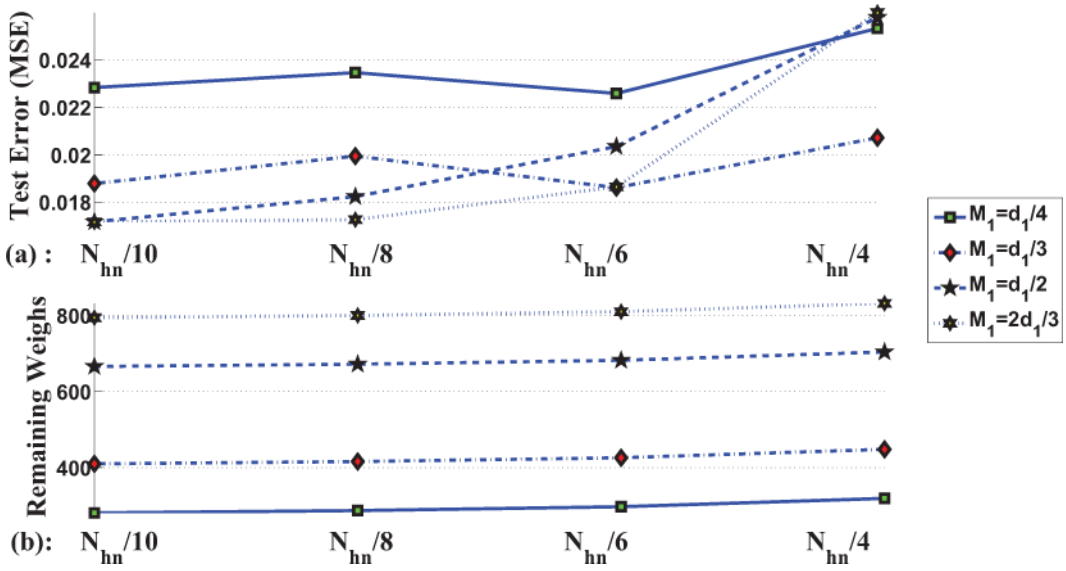

Fig. 3. Different sparse levels and values of $M1 = [d_1/4, d_1/3, d_1/2, 2d_1/3]$ and $M2 = [N_{hn}/10, N_{hn}/8, N_{hn}/6, N_{hn}/4]$ for the CSP1 Algorithm

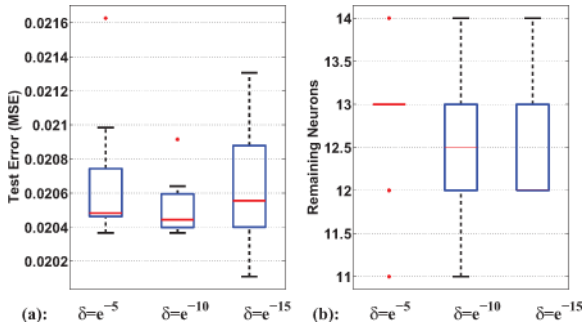

Fig. 4. For the CSP2 algorithm with different value of δ, (a) Error Comparison, (b) Remaing Topology

it may require a higher computational cost to calculate the topology. For the Flare1 problem, the results are not shown, but the same conclusions can be drawn.

C. Discussion

In this section, we discuss some potential methods for improving the performance of the proposed algorithms. Firstly, the scale of the coefficients for CSP1 is set as $M2 = d_1/2$ and $M1 \in [1, N_{hn}/10]$, and the maximum number for iterations in CSP2 is given by $Iterations = 100$. The ***generation curves*** for both methods are shown in Fig. 5 and 6. We firstly note that the curves of "Error with A" overlaps "Error with Z", confirming that the result is the same using A or Z in Eqs. (8) and (9) since Z is very close to A after training. Secondly, the different behavior for pruning NNs between CSP1 and CSP2 algorithm is also found: CSP1 can be regarded as reconstructing the original network by adding weights at each iteration in that the number of weights is increasing in Fig. 5; on the other hand, CSP2 begin with a network which has the same szie as the original one and then deletes some hidden neurons. In Fig. 5, we notice that from *Episode* 2 the error on test data obtained from CSP1 did not change greatly. Actually, it is better to stop the CSP1 algorithm here in that it corresponds to the smallest number of remaining weights. The same phenomenon can be found in Fig. 6 as well because the topology seems to maintain the same from *Episode* 2 in CSP2. Another way we can benefit from them is that we save the computational cost.

V. Conclusions

We have extended the application of Compressive Sampling (CS) to design the topology of Neural Networks (NNs). Specifically, using the pursuit methods in CS, a novel pruning mechanism for NNs has been presented in this paper. We regard the different input or output for layers of NNs as a dictionary in CS, and then the goal for finding a minimal topology in NNs is simply changed into locating a sparse structure. The key difference between our proposed algorithm and the existing techniques is that we only need to focus on the remaining elements; our method can lead to a quick convergence and a better topology. The empirical work demonstrates that our algorithm is an effective alternative to traditional pruning methods in terms of (generalization) accuracy and computational complexity.

There are still some open questions in the study of NNs topology based on Compressive Sampling. For instance, is it possible to obtain the optimal architecture without training the NNs first since all the algorithms need to obtain a trained topology before pruning? However, it is very time-consuming to train a large network architecture. Moreover, we notice that our method is an off-line mechanism, which means the performance depends on the volume of available data.

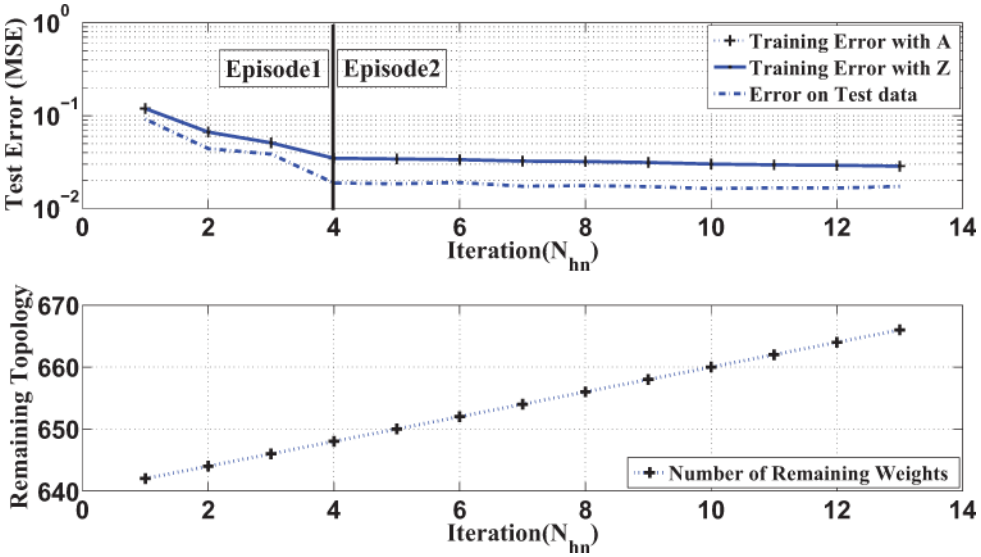

Fig. 5. Generation Curve in CSP1 with $M2 = d_1/2$ and $M1 \in [1, N_{hn}/10]$

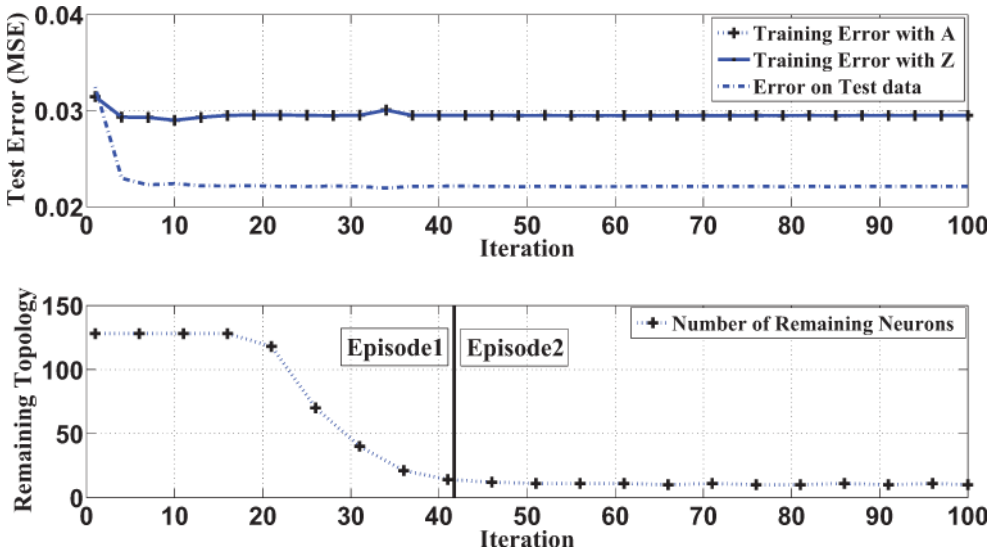

Fig. 6. Generation Curve with maximum iterations in CSP2

It would be interesting to extend the proposed methods to online training.

REFERENCES

[1] R. Reed, "Pruning algorithms—a survey," *IEEE Transactions on Neural Networks*, vol. 4, pp. 740-747, May. 1993.

[2] Y. Le Cun, J.S. Denker,and S.A. Solla, "Optimal brain damage," *Adv. Neural Inf. Process. Syst*, vol. 2, pp. 598-605, 1990.

[3] B. Hassibi and D. G. Stork, "second-order derivatives for network pruning: optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 5, pp. 164-171, 1993.

[4] L. Prechelt, *Proben1—A set of neural networks benchmark problems and benchmarking rules.* Univ.Karlsruher,Karlsruher,Germany,Tech. Rep, 21/94, 1994.

[5] M. Hagiwara, "Removal of hidden units and weights for backpropagation networks," *Proc. IEEE Int. Joint Conf. Neural Netw.*, pp. 351-354, 1993.

[6] M.Mozer and P.Smolensky, "Skeletonization: a technique for trimming the fat from network via relevance assessment," *Advances in Neural Information Processing Systems*, vol. 1, pp. 107-115, 1991.

[7] J. Sietsma and R. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67-79, 1991.

[8] Candes.E.J., Romberg.J, Tao.T., "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform.Theory*, vol. 52, pp. 489-509, 2006.

[9] Haupt, J.; Nowak, R., "Signal Reconstruction From Noisy Random Projections," *IEEE Trans. Inform.Theory*, vol. 52, pp. 4036-4048, 2006.

[10] Rauhut, H.; Schnass, K.; Vandergheynst, P.; "Compressed Sensing and Redundant Dictionaries," *IEEE Trans. Inform.Theory*, vol. 54, pp. 2210 - 2219, May 2008.

[11] Tropp, J.A. and Anna C. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," *IEEE Trans. Inform.Theory*, vol. 53, pp. 4655-4666, 2007.

[12] Cotter, S.F, Rao, B.D, Kjersti Engan, Kreutz-Delgado, K, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2477-2488, July. 2005.

[13] J. Chen and X. Huo, "Theoretical Results on Sparse Representations of Multiple-Measurement Vectors," *IEEE Transactions on Signal Processing*, vol. 54, No. 12, pp. 4634-4643, December 2006.

[14] Lauret, P, Fock, E, Mara, T.A, "A node pruning algorithm based on a Fourier amplitude sensitivity test method," *IEEE Transactions on Neural Networks*, vol. 17, pp. 273-293, March 2006.

[15] S Amaril, "Information geometry on hierarchical of probability distributions," *IEEE Trans on Inf Theory*, vol. 47, No. 5, pp. 1701-1711, 2001.

[16] Y H Liu, S W Luo , A J Li, "Information geometry on pruning of neural network," *Intel Conf on Machine Learning and Cybernetics*, Shanghai. 2004.

[17] Remi Gribonval and Pierre Vandergheynst, "On the Exponential Convergence of Matching Pursuits in Quasi-Incoherent Dictionaries," *IEEE Transactions on Inf Theory*, vol. 52, pp. 255-261, 2006.

[18] J.Moody and P.J.Antsaklis, "The dependence identification neural network construction algorithm," *IEEE Trans on NEURAL NETWORK*, vol. 7, pp. 3-15, Jan. 1996.

[19] D. White and P. Ligomenides, "GANNet: A genetic algorithm for optimizing topology and weights in neural network design," *in International Workshop on Artificial Neural Networks, in New Trends in Neural Computation,*, Berlin, Germany:Springer-Verlag, pp. 332-327,1993

[20] S. Mallat and Z. Zhang, "Matching pursuits with time frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, pp. 3397-3415, 1993.

[21] Rao, B.D.; Engan, K.; Cotter, S.F.; Palmer, J.; Kreutz-Delgado, K.;, "Subset selection in noise based on diversity measure minimization," *IEEE Transactions on Signal Processing*, vol. 51, pp. 760-770, 2003.