Towards Efficient Learning of Neural Network Ensembles from Arbitrarily Large Datasets

Kang Peng, Zoran Obradovic and Slobodan Vucetic[1]

Abstract. Advances in data collection technologies allow accumulation of large and high dimensional datasets and provide opportunities for learning high quality classification and regression models. However, supervised learning from such data raises significant computational challenges including inability to preserve the data in computer main memory and the need for optimizing model parameters within given time constraints. For certain types of prediction models techniques have been developed for learning from large datasets, but few of them address efficient learning of neural networks. Towards this objective, in this study we proposed a procedure that automatically learns a series of neural networks of different complexities on smaller data chunks and then properly combines them into an ensemble predictor through averaging. Based on the idea of progressive sampling the proposed approach starts with a very simple network trained on a very small sample and then progressively increases the model complexity and the sample size until the learning performance no longer improves. Our empirical study on a synthetic and two real-life large datasets suggests that the proposed method is successful in learning complex concepts from large datasets with low computational effort.

1 INTRODUCTION

For a long time the machine learning community has focused on learning from datasets of moderate size ranging from less than a hundred to several thousand. In this scenario, the amount of available data is often not sufficient for optimal learning of the underlying relationships. Consequently, a major research challenge is to design a learning process that gains the most from the available data in terms of model selection, learning, and accuracy estimation. More recently, advances in data collection techniques allowed accumulation of large and high dimensional datasets in domains such as geosciences, bioinformatics, network intrusion detection, and credit card fraud detection. While the abundance of data provides an opportunity to learn high-quality classification and regression models, it also creates significant computational difficulties related to storage, processing, and learning from such large datasets. In such data-rich scenarios, the emphasis thus shifts to development of procedures for cost-effective learning from arbitrarily large datasets.

As one of the most powerful machine learning algorithms, neural networks are suitable for learning highly complex concepts given sufficiently large data using the back-propagation algorithm (and its variants) in training time that scales linearly with data size [1]. A common problem in learning neural networks is how to determine appropriate model architecture that fits the complexity of the dataset at hand. In the case of single-layer feedforward neural networks, the problem is reduced to determining the number of hidden nodes. Networks with small number of hidden nodes may not have sufficient representational power to model the complexity and diversity inherent in the data, while networks with too many hidden neurons are very costly to train and could overfit the data. As reviewed in [2], this model selection problem can be formulated as searching for the maximum on a performance surface defined over all possible network architectures. This complex surface can be very large, nondifferentiable, noisy and multimodal, which makes such search ineffective and inefficient. Existing solutions include manual trial-and-error procedures and more advanced automatic methods such as constructive learning [3], network pruning [4], and evolutionary learning [2, 5]. However, although successful in learning from moderate-sized data, these methods were not designed to address issues related to learning from arbitrarily large datasets.

When a dataset is too large to fit into computer main memory learning a single neural network model on all available data becomes extremely inefficient due to the need for accessing the data from secondary memory. This problem can be often avoided by reducing redundancies common in real-life data, i.e. selecting a data sample that is as small as possible but still sufficient for learning high-quality predictor. However, in practice determining such a minimal sample size n_{min} is not trivial. A common strategy for this purpose is *progressive sampling* [6], which was originally proposed for learning of decision trees. It builds a series of models on progressively larger samples until prediction performance no longer improves. The worst case efficiency of progressive sampling with geometric schedule is about the same as learning on the entire data set, while major efficiency improvements were reported on large scale experiments where the learning accuracy on progressively increasing samples reached a plateau quickly.

However, progressive sampling alone is often insufficient while using neural networks to learn highly complex concepts from arbitrarily large datasets. As samples become much larger it might be appropriate to increase the model complexity to achieve better representation abilities. This issue is automatically dealt with if using learning algorithms such as decision trees. But for neural networks the model complexity (e.g. the number of hidden neurons) cannot be adjusted by the back-propagation algorithm. For example, the learning curve [6] of single-layer feedforward neural networks with 5 hidden neurons may saturate earlier (i.e. resulting in a smaller n_{min}) at a significantly lower level than those with 10 hidden neurons. At the same time neural networks with 10 hidden neurons might be able to learn more accurate models for complex concepts.

[1] Center for Information Science and Technology, Temple University, Philadelphia, PA, USA, {kangpeng, zoran, vucetic}@ist.temple.edu

Ensemble methods [7, 8] provide another solution for learning from arbitrarily large datasets. Typically individual models are built on small samples or subsets of the original data, which can be fit into available main memory, and then combined as the final model. It was shown that such ensembles could achieve performance comparable to a single predictor built on all available data [8]. In addition they can readily benefit from parallel and distributed computing environment. However, similar problem remains for efficient learning: one still has to determine the sample size and number of hidden neurons for individual networks, and furthermore, the ensemble size; finding the best combination of these three parameters is still a non-trivial problem.

In this work the objective is to address the difficulties involved in learning neural networks with a single hidden layer of sigmoidal units from arbitrarily large datasets. We proposed an iterative procedure based on integration of progressive sampling and ensemble methods. The proposed approach starts with a very simple network trained on a very small sample and then progressively increases model complexities and sample sizes until the learning performance no longer improves. Our empirical study on three real life large datasets suggests that the proposed method is successful in learning complex concepts from large datasets with low computational effort.

2 METHODOLOGY

Given an arbitrarily large dataset, our goal is to develop a cost-effective procedure to learn an ensemble of neural networks of close-to-optimal accuracy with close-to-minimum computational cost. We describe an ensemble using three parameters: E as the ensemble size, H as the number of hidden neurons, and N as the training sample size of individual networks. Our basic assumption is that the accuracy of such ensembles increase with each of the three parameters but saturates after a certain point. Since the ensemble learning time using the back-propagation algorithm scales linearly with E, H, and N, we can use their product $H*N*E$ as a good estimate of the required computational cost.

In the space of all possible triples (H, N, E) there should exist an optimal combination of $H = H_O$, $N = N_O$ and $E = E_O$ that results in an ensemble with the highest possible accuracy and minimal $H*N*E$. However, searching for such an optimal combination cannot be done efficiently since it is at least as difficult as searching for optimal architecture and training sample size for a single neural network. Thus, we approximate this optimal solution through an iterative procedure (Figure 1) that learns a series of neural networks with possibly different H and N. Starting with a simple network (H = 1) trained on a small sample (N = 40), it gradually includes more complex networks trained on larger samples. In this way it greedily explores the space of H-N in a cost-effective fashion.

The inputs for the proposed procedure are a large dataset $\boldsymbol{D}$ and certain computational constraints such as maximal allowed sample size (main memory) $\boldsymbol{N_{max}}$ and execution time $\boldsymbol{T_{max}}$. Its output is an ensemble of neural networks when it converges or the specified computational constraints are met. The dataset $\boldsymbol{D}$ is initially divided into 3 disjointed sets $\boldsymbol{D_{TR}}$, $\boldsymbol{D_{VS}}$ and $\boldsymbol{D_{TS}}$. The samples for training are drawn from $\boldsymbol{D_{TR}}$ only, $\boldsymbol{D_{VS}}$ is used for model evaluation and selection during the learning process, while $\boldsymbol{D_{TS}}$ is used for evaluation of the final ensemble predictor. For simplicity, we assume that examples are stored in a random order in $\boldsymbol{D}$. To draw a sample of size N, the procedure reads N examples sequentially from the current file pointer. If the end of file is encountered the file is rewinded. Note that if $\boldsymbol{D}$ is large, only a fraction of data would be used by the procedure.

Procedure in Figure 1 is not restrictive to the type of learning problem and can be applied to both classification and regression problems. The only difference is in the performance measure used: percentage of correct classification for classification and coefficient of determination R^2 for regression. The R^2 is defined as 1 - $MSE/VAR(y)$, where $VAR(y)$ is variance of the target variable y and MSE is the estimated mean squared error. Other measures are also possible, e.g. cost-based measures for classification problems.

Input: A large dataset $\boldsymbol{D}$, upper limit for sample size (main memory) $\boldsymbol{N_{max}}$, upper limit for execution time $\boldsymbol{T_{max}}$
Output: A neural network ensemble

Divide $\boldsymbol{D}$ into 3 disjoint sets: $\boldsymbol{D_{TR}}$, $\boldsymbol{D_{VS}}$, $\boldsymbol{D_{TS}}$
$H_1 := 1$, $N_1 := 40$, $i := 0$
Repeat
- $i := i + 1$
- Draw a sample $\boldsymbol{S_i}$ of size N_i from $\boldsymbol{D_{TR}}$
- Build neural network NN_i of H_i hidden neurons on $\boldsymbol{S_i}$
- Identify the best ensemble out of networks NN_1, NN_2, …, NN_i, and store its accuracy on $\boldsymbol{D_{VS}}$ in ACC_i
- **if** ACC_i is significantly higher than ACC_{i-1} **then**
  $H_{i+1} := H_i$, $N_{i+1} := N_i$
  else
  $H_{i+1} := H_i + I_H$ **if** $N_{i-1} = N_i$
  $N_{i+1} := N_i * F_A$ **if** $H_{i-1} = H_i$
  end

until convergence or $\boldsymbol{N_i} \geq \boldsymbol{N_{max}}$ (or execution time $\geq \boldsymbol{T_{max}}$)
Identify the best ensemble out of the available networks

Figure 1. The procedure for learning from large dataset given certain computational constraints.

After network NN_i is built at the i-th iteration, a total of 2^i-1 ensembles (including individual networks) can be constructed out of the available i networks through averging or majority voting. The ensemble with the highest accuracy on $\boldsymbol{D_{vs}}$ is selected and its accuracy stored in ACC_i as the performance for this iteration. However, to prevent examination of the exponentially increasing number of ensembles after each iteration we use the assumption that a network trained with more examples (note that such a network is of the same size or larger than any network trained previously) should result in a more accurate ensemble. Based on the assumption, we evaluate only i ensembles consisting of networks NN_j, NN_{j+1}, …, NN_i for j ranging between 1 and i.

The values of H and N are automatically adjusted for the $i+1$-th network (iteration) based on the learning performance in the previous 3 iterations: ACC_{i-2}, ACC_{i-1} and ACC_i. If the improvement from ACC_{i-1} to ACC_i is statistically significant, the current values of H and N will be used in the next iteration. The reasoning is that as long as the accuracy is improving the network complexity and training size should be kept at the minimum. However, the improvement from ACC_{i-1} to ACC_i may be insignificant or even negative. In this case the procedure explores networks with higher complexity trained on larger number of examples. We distinguish two different scenarios: (1) If the improvement from ACC_{i-2} to ACC_{i-1} was insignificant or negative and N was increased in the previous iteration, since it is likely that increasing N further may not help, number of hidden neurons H is increased in the following iteration; Conversely, (2) if H was increased in the previous iteration and the accuracy is not improved significantly, the training size N is increased in the following iteration. In our current implementation, H is always

increased by a fixed amount of I_H ($I_H > 0$), while N is multiplied by a constant factor of F_A ($F_A > 1$). Although it might be more efficient to progressively increase H by multiplying it with a constant factor, we found it difficult to select an appropriate factor that prevents too fast increase in network complexity.

To decide if ACC_i is significantly improved over ACC_{i-1}, we examine if their 90% confidence intervals (CIs) overlap. For classification problems the 90% CI for ACC_i is calculated as $ACC_i \pm 1.645 \cdot \text{sqrt}(ACC_i \cdot (1-ACC_i)/|\boldsymbol{D_{VS}}|)$ [9]. For regression problems it is calculated as $ACC_i \pm 1.645 \cdot \text{SE}(ACC_i)$, where $\text{SE}(ACC_i)$ is the standard error estimated by bootstrapping [10]. A total of 1000 bootstrap replicated samples were drawn from $\boldsymbol{D_{VS}}$, accuracies (R^2) were calculated on each of them, and one standard deviation of the observed accuracies was used as $\text{SE}(ACC_i)$.

To detect convergence, we measure the ratio between the standard deviation and mean of the accuracies during the latest N_C iterations. If this ratio is smaller than a pre-specified small threshold q_C ($q_C > 0$), i.e. the accuracy variation is sufficiently small during the last N_C iterations, the procedure stops. It also terminates if the current training sample size N_i is larger than the pre-specified upper limit $\boldsymbol{N_{max}}$, or if the cumulative execution time exceeds $\boldsymbol{T_{max}}$.

According to our assumption, a network built on a sample should have higher accuracy than those trained on smaller samples. In practice, this does not always hold due to the instability of back-propagation algorithms that produce solutions at different local minima. Thus, it is in general hard to estimate the expected performance given a certain combination of H and N if only a single network is built. Consequently, the decisions for adjusting H and N by the procedure might be misleading. An intuitive but effective method was used to alleviate this problem. If the accuracy of a network is significantly less accurate than the networks trained previously with comparable N and H, the procedure will discard it and train another one with different initial weights on the same sample. Here the significance of accuracy comparison is also estimated based on their 90% confidence interval overlap. However, if the re-training is repeated 3 times and the problem remains, the procedure proceeds by incrementing N and training a new network. Although in the worst case, we have to repeat training 3 times for each combination of H and N, in practice it occurs infrequently when H and N are fairly large. Therefore, the re-training is not likely to consume too much computational time.

3 EXPERIMENTAL RESULTS

Three large datasets were used in this study. The *Waveform* dataset is artificially generated benchmark dataset with 21 continuous attributes and 3 target classes [11]. The highest achievable accuracy on this dataset using a Bayes optimal classifier is 86.8%. For this study, we generated 100,000 examples from each of the 3 classes. The *Covertype* dataset [12] has 54 attributes and 7 target classes, and consists of 581,012 examples. Out of the 54 attributes, 40 are binary representing soil type, another 4 binary ones represent wilderness area, and the remaining 10 are continuous topographical attributes. Seven classes represent forest cover types. We transformed the first 40 binary attributes into 7 new real value attributes as in [13] to reduce the dimensionality. Thus the transformed dataset had 21 attributes. The originally reported neural network accuracy was around 70% [12]. The *MISR* dataset [14] has 36 continuous attributes representing radiance values measured by *MISR* satellite and one continuous target variable representing retrieved aerosol optical depth. For this study we used *MISR* data consisting of 45,449 examples retrieved over land for the 48 contiguous United States during a 15-day period of summer 2002.

Each dataset was divided into 3 disjointed subsets $\boldsymbol{D_{TR}}$ for training, $\boldsymbol{D_{VS}}$ for model selection, and $\boldsymbol{D_{TS}}$ for testing. For *Waveform* and *Covertype* datasets, the number of model selection and testing examples $|\boldsymbol{D_{VS}}|$ and $|\boldsymbol{D_{TS}}|$ were 10,000; while for *MISR* datasets $|\boldsymbol{D_{VS}}|$ and $|\boldsymbol{D_{TS}}|$ were 5,000 due to somewhat smaller size of the dataset. Samples for training of individual neural networks were randomly drawn from $\boldsymbol{D_{TR}}$ without replacement. 75% of each sample was used for neural network training and the remaining 25% as the early stopping set. The resilient back-propagation algorithm [15] was used with default learning parameters and for maximum of 300 epochs. The reported accuracy measure for *MISR* experiments is coefficient of determination R^2.

We first examined the effect of number of hidden neurons (H) and sample sizes (N) on learning individual neural networks and their ensembles. For each combination of H and N, we trained 20 neural networks on randomly sampled data from $\boldsymbol{D_{TR}}$ and tested their accuracy on $\boldsymbol{D_{TS}}$. For each pair (H, N) in Table 1 we report (1) accuracy (mean and standard deviation) of individual networks; and (2) accuracy of an ensemble of the 20 networks ($E = 20$).

Several interesting conclusions could be made from the obtained results. First, ensemble accuracy was higher than that of the individual neural networks over the whole range of pairs (H, N) with the difference being more pronounced at small sample sizes. An interesting phenomenon occurs on *MISR* data where ensembles of worse-than trivial but complex predictors resulted in considerably higher accuracy. Second, it was evident that problem complexity has a major influence on the needed training data size and model complexity; while an ensemble of neural networks with 10 hidden nodes each trained on 400 examples seems to be sufficient for achieving optimal accuracy on *Waveform* data, accuracy on *Covertype* and *MISR* data continually increased over the whole range of data sizes and numbers of hidden neurons. It is worth observing that on *Covertype* data the ensemble with $H = 40$ and $N = 12{,}800$ achieved 7% higher accuracy than originally reported at [12].

Probably the most interesting result is that, given a fixed computational effort measured as $H*N*E$, neural network ensembles with $E = 20$ components appear to be inferior to individual neural networks. For example, on *Covertype* and *MISR* data single networks with 10 hidden nodes trained on 12,800 examples had significantly higher accuracy than ensembles of 20 networks each trained with 800 examples for any choice of H. Additional study of this phenomenon (data not shown) confirmed that, for fixed $H*N*E$, individual neural networks are surprisingly competitive to neural network ensembles with any $E > 1$.

The cost-efficient learning procedure described in Figure 1 has several parameters that should be selected prior to the experimental evaluation. We performed a preliminary exploratory study over the *Waveform* dataset to decide on the appropriate parameter choice. Based on this study, for all datasets we used 4 as the increment amount I_H for H and 1.5 as the multiplication factor F_A for N. The two parameters for convergence detection were fixed to $N_C = 5$ and $q_C = 0.0025$. It is worth noting that the exploratory study indicated that performance of the proposed procedure was fairly robust to the parameter choice. In all

experiments, only the sample size (main memory) upper limit of N_{max} = 20,000 was used.

Table 1. The effect of number of hidden nodes *H* and sample size *N* for single neural networks and ensembles of 20 components. The accuracy (%) is reported for *Waveform* and *Covertype* and, R^2 for *MISR* dataset.

	H		
N	1	10	40
50	51.5±8.1 / 75.3	75.7±3.3 / 84.8	77.2±2.9 / 85.5
100	49.9±9.8 / 81.6	79.3±1.9 / 86.2	80.9±1.6 / 86.3
200	52.9±7.4 / 83.9	81.9±1.0 / 86.4	83.0±1.0 / 86.3
400	56.1±4.5 / 83.9	83.9±0.5 / 86.7	84.3±0.4 / 86.6
800	57.7±3.1 / 83.6	85.0±0.5 / 86.9	85.2±0.6 / 86.8
1600	59.8±2.4 / 84.4	85.6±0.3 / 86.8	85.9±0.2 / 86.7
3200	59.2±1.5 / 83.9	86.2±0.3 / 87.0	86.2±0.2 / 86.8
6400	59.6±1.6 / 83.9	86.4±0.2 / 87.1	86.4±0.2 / 86.9
12800	60.1±1.2 / 84.1	86.3±0.2 / 87.1	86.5±0.2 / 86.9

(a) *Waveform*

	H		
N	1	10	40
50	29.3±18.7 / 43.1	51.2±4.0 / 62.6	51.5±3.9 / 62.7
100	38.7±15.2 / 55.6	57.6±3.4 / 67.4	56.7±2.7 / 66.6
200	49.0±09.9 / 63.3	61.2±2.1 / 69.0	61.1±2.5 / 68.8
400	41.4±19.7 / 66.0	64.5±2.0 / 70.3	64.9±2.0 / 70.4
800	46.7±20.4 / 68.1	66.9±1.7 / 70.6	67.9±0.9 / 71.1
1600	46.2±20.4 / 65.0	69.2±0.8 / 71.4	69.7±0.6 / 72.3
3200	53.6±13.7 / 67.0	70.8±0.7 / 72.6	71.3±0.4 / 73.5
6400	53.7±14.9 / 68.7	72.1±0.6 / 73.3	72.8±0.6 / 74.8
12800	52.2±15.0 / 68.5	73.1±0.4 / 74.0	74.9±0.6 / 77.1

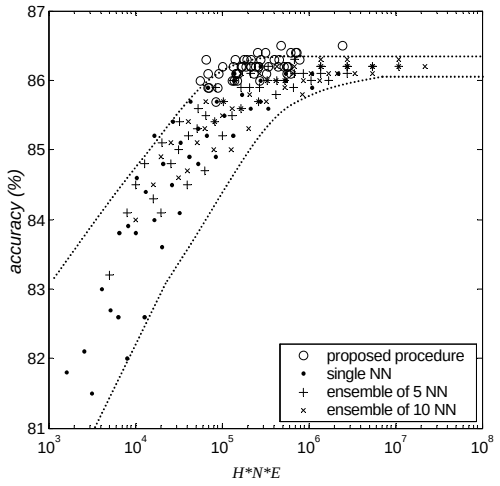
(b) *Covertype*

	H		
N	1	10	40
50	0.40±0.15 / 0.54	-0.12±0.41/0.35	-3.39±2.02/-0.08
100	0.43±0.11 / 0.59	0.23±0.21/0.53	-0.60±0.63/0.31
200	0.56±0.05 / 0.63	0.38±0.14/0.64	-0.07±0.18/0.53
400	0.61±0.08 / 0.65	0.57±0.08/0.66	0.23±0.13/0.63
800	0.63±0.06 / 0.67	0.59±0.04/0.65	0.30±0.30/0.64
1600	0.66±0.05 / 0.70	0.65±0.08/0.73	0.60±0.16/0.72
3200	0.68±0.04 / 0.71	0.75±0.04/0.80	0.64±0.16/0.78
6400	0.71±0.01 / 0.73	0.79±0.02/0.82	0.77±0.06/0.83
12800	0.72±0.01 / 0.74	0.82±0.03/0.84	0.80±0.03/0.85

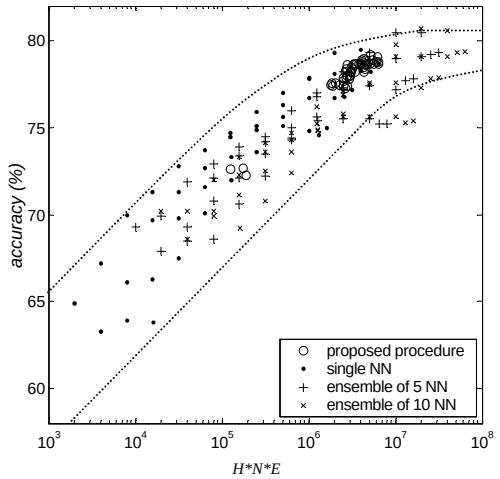
(c) *MISR*

The procedure was repeated 50 times on each dataset until its convergence or until the training sample size exceeded the upper limit N_{max}. For comparison, we also trained ensemble predictors of size $E \in \{1, 5, 10\}$ using neural networks with $H \in \{1, 5, 10, 20, 40, 80\}$ trained with $N \in \{200, 400, 800, 1600, \ldots, 204800\}$ examples. Since training an ensemble of neural networks based on the resilient backpropagation algorithm has time complexity of O(*H*N*E*), the product *H*N*E* provides a good estimate of the total computation effort. In Figure 2 we show the scatter plot of prediction accuracy (percent correct or R^2) vs. *H*N*E* for each triple (*H, N, E*); each ensemble predictor is represented as a single point on the figure. For ensembles constructed by our approach, we summed products *H*N* for all the networks trained during the learning process, and represented the resulting accuracy vs. sum(*H*N*) as a circle at the same figure.

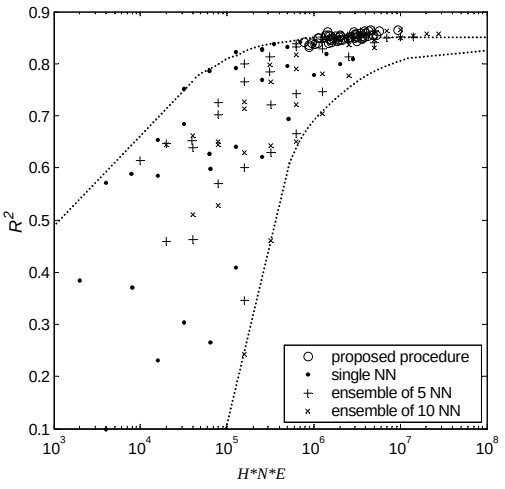
From Figure 2, a similar behavior can be observed for all three datasets, i.e. the ensembles occupy an area that resembles the shape of a learning curve [6]. At relatively small values of *H*N*E*, the highest achievable accuracy is gradually improved with increase in *H*N*E*. Starting from a certain critical value of *H*N*E* that is largely dependent on the dataset properties, a saturation region is reached where only marginal accuracy improvements are possible with increase in *H*N*E*. This critical



(a) *Waveform*



(b) *Covertype*



(c) *MISR*

Figure 2. Evaluation of the proposed procedure.

value and the corresponding triple (*H, N, E*) that results in an ensemble with maximum accuracy can be considered as the optimal tradeoff between accuracy and computational costs of learning. We refer to the optimal choice of (*H, N, E*) as the *oracle* solution. Successful cost-effective learning procedure should be

able to produce an accurate ensemble with a comparable computational effort.

As can be seen from Figure 2, most circles representing the ensembles constructed using our procedure are located close to the upper boundary of the learning curve area near the critical *H*N*E* region. More specifically, all 50 runs of our procedure achieved near-optimal accuracy on *Waveform* and *MISR* datasets after the computational effort that is of the same order of magnitude as the *oracle* solution. In the case of *Covertype* data, the proposed procedure resulted in ensembles slightly inferior to the oracle solution (that exceeded 80%), but with near one order of magnitude smaller computational effort. This behavior could be explained by the sample size limit of N_{max} = 20,000 set as the procedure termination criterion. We note that oracle solution corresponded to an ensemble of 10 neural networks with 80 hidden nodes trained with 204,800 examples. We also observe that in several runs for *Covertype* data the procedure converged too early with the resulting accuracy below 75%. This result indicates that additional improvements of the proposed procedure might be necessary.

In Table 2 we show the summary information based on the 50 runs of the proposed procedure. N_{total} is the total number of examples used for training all individual networks including those that are not selected for the resulting ensemble predictor. Note that for *MISR* data N_{total} is much larger than its small original size ($|\boldsymbol{D}_{TR}|$ = 35,449) since it was scanned 3 times during learning. *Accuracy* is the prediction accuracy estimated on $\boldsymbol{D}_{TS}$ (R^2 is used for *MISR* dataset). For *Waveform* data the achieved accuracy reached previously reported 86.8% score of the optimal Bayes classifier. For *Covertype* data the obtained accuracy was much higher than the previously reported 70%, and in most cases exceeds 77%. The remaining 3 columns provide description of the component networks included in the resulting ensemble. This confirms that our procedure successfully adapted to the inherent data complexity by constructing relatively simple neural networks on small training datasets for *Waveform* data, and relatively complex networks on much larger training datasets for more difficult *Covertype* and *MISR* datasets.

Table 2. Summary of 50 runs of the proposed procedure on 3 datasets.

Dataset	N_{total}	*Accuracy*	*E*	*N*	*H*
waveform	12753	86.1±0.1%	6-12	649-2950	10-24
covertype	86309	78.0±1.5%	2-6	9983-14022	31-37
MISR	100815	0.85±0.01	4-8	7854-14187	18-26

N_{total} – total number of examples used, *Accuracy* – prediction accuracy on $\boldsymbol{D}_{TS}$, *E* – final ensemble size *N* – individual sample size, *H* – number of hidden neurons for single NN

4 CONCLUSIONS

In this study we proposed a procedure for cost-effective learning of an ensemble of single-layer feedforward neural network predictors from arbitrary large datasets. It builds a series of networks on samples much smaller than the original data and thus avoids the computational overhead associated with learning a complex network using all available data. The differences between our procedure and progressive sampling are in automatically adjusting model complexity and utilizing the previously built networks to guide the learning process. As our experimental study suggested, the proposed approach could learn predictors with near-optimal accuracy with high probability while requiring only modest computational effort that is a function of the inherent complexity of the learning task at hand.

We are currently exploring several avenues of research aimed at achieving more robust and computationally cheaper procedures for learning from very large datasets. The procedure reported in this study has a number of free parameters that need to be appropriately selected prior to its application. Our next task is to avoid parameter selection process by determining the default values that work over a large range of domains, and by developing parameter selection techniques adaptable to the specific application. As an example, it would be beneficial if model complexity and training data size increments could be decreased when the learning process is close to convergence. While the current procedure corresponds to the static learning scenarios, we are exploring extensions that would allow its use to data mining applications over data streams with concept drifts.

ACKNOWLEDGEMENTS

This research was supported in part by the NSF grant IIS-0219736 to Z. Obradovic and S. Vucetic and Temple University New Directions Studies grant to Z. Obradovic. We thank Dr. Amy Braverman for help with obtaining MISR data.

REFERENCES

[1] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[2] G.F. Miller, P.M. Todd and S.U. Hegde, 'Designing Neural Networks Using Genetic Algorithms', *Proc. 3rd Int'l Conf. on Genetic Algorithms and Their Applications*, J. D. Schaffer, Ed. San Mateo, CA, Morgan Kaufmann, 379–384, 1989.

[3] S.E. Fahlman and C. Lebiere, 'The Cascade-Correlation Learning Architecture', *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann, San Mateo, CA, 524-532, 1990.

[4] B. Hassibi, D.G. Stork and G.J. Wolff, 'Optimal Brain Surgeon and General Network Pruning', *Proc. IEEE Int'l Conf. on Neural Networks*, San Francisco, **1**, 293-299,1993.

[5] X. Yao, 'Evolving Artificial Neural Networks', *Proc. of the IEEE*, IEEE Press, **87(9)**,1423-1447,1999.

[6] F. Provost, D. Jensen and T. Oates, 'Efficient Progressive Sampling', *Proc. 5th Int'l Conf. on Knowledge Discovery and Data Mining*, 23-32, 1999.

[7] L. Breiman, 'Pasting Small Votes for Classification in large databases and on-line', *Machine Learning*, **36**, 85-103, 1999.

[8] N.V. Chawla, T.E. Moore, K.W. Bowyer, L.O. Hall, C. Springer and W.P. Kegelmeyer, 'Bagging is a Small Dataset Phenomenon', *Proc. Int'l Conf. of Computer Vision and Pattern Recognition (CVPR)*, 684-689, 2000.

[9] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[10] B. Efron, and R.J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall, New York, 1993.

[11] L. Breiman, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.

[12] J. Blackard, *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types*, PhD dissertation, Colorado State University, Fort Collins, 1998.

[13] S. Vucetic and Z. Obradovic, 'Performance Controlled Data Reduction for Knowledge Discovery in Distributed Databases', *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 29-39, 2000.

[14] A. Braverman, and L. DiGirolamok, '*MISR* Global Data Products: A New Approach', *IEEE Trans. Geoscience and Remote Sensing*, **40(7)**, 1626-1636, 2002.

[15] M. Riedmiller and H. Braun, 'A Direct Adaptive Method for Faster Backpropogation Learning: the RPROP Algorithm', *Proc. IEEE Int'l. Conf. on Neural Networks*, **1**, 586-591, 1993.