An Incremental Neural Network Construction Algorithm for Training Multilayer Perceptrons

Oya Aran, Ethem Alpaydın
Department of Computer Engineering,
Boğaziçi University TR-34342
Istanbul, Turkey
{aranoya, alpaydin}@boun.edu.tr

***Abstract*— The problem of determining the architecture of a multilayer perceptron together with the disadvantages of the standard backpropagation algorithm, directed the research towards algorithms that determine not only the weights but also the structure of the network necessary for learning the data. We propose a Constructive Algorithm with Multiple Operators using Statistical Test (MOST) for determining the architecture. The networks that are constructed by MOST can have multiple hidden layers with multiple hidden units in each layer. The algorithm uses node removal, addition and layer addition and determines the number of nodes in layers by heuristics. It applies a statistical test to compare different architectures. The results are promising and near optimal.**

I. Introduction

Multilayer networks with fixed topology trained using standard *backpropagation* based on *gradient descent* are the most common use of neural network models. These networks are only useful with the appropriate network architecture. The standard back propagation algorithm finds the network weights using gradient descent procedure but the network architecture is found by trial and error. The optimal architecture is a network large enough to learn the underlying function, and as small as possible to generalize well. A network smaller than the optimal architecture can not learn the problem, but on the other hand a larger network will overlearn the data with a poor generalization performance. The generalization performance of neural networks can be viewed as the *bias/variance dilemma* [1]. The trade off between the bias and the variance is the key factor in the generalization performance of a neural network. A small network will have a high bias and will fail to learn the underlying function generating the data. If we use a large network, then the bias will be close to zero (it will even be zero if there are enough number of units in the network; in that case, the network will interpolate the data) but then a high variance will be introduced. The optimal architecture is the one that balances the bias and the variance so that the network can generalize the data, ignoring the noise.

All algorithms that determine the network architecture have to start with an initial architecture. The initial architecture is determined by the nature of the algorithm used. In constructive approach, the algorithm starts with a small network and constructs the network by adding nodes and connections [2]. The simplest network is the network with no hidden units. In problems where prior information exists, the initial state can be different. In pruning approach [3], in contrast to the constructive approach, the algorithm starts with a large network and removes the unnecessary nodes and connections. The search must be terminated when the generalization performance of the network begins to decrease. Some algorithms continue until all training examples are correctly classified but these algorithms fail to learn noisy data and generate larger networks.

The search strategy determines how to reach to the best architecture starting with the initial architecture in general. In particular, it determines the next state and how to move to the next state from the current state. In some algorithms there is only one next state. The disadvantage of this type of algorithms is that, finding a good architecture for any kind of problem can be impossible since there is only one possible next state. In the multi-valued case, there are candidate next states and the algorithm chooses one among the candidates. The disadvantage of these algorithms is their high computation time.

Constructive approach is generally preferred to the pruning approach with a number of advantages. Specifying the initial network is easier in constructive methods. In pruning methods, one has to decide how big the initial network must be whereas it is easy to found an initially small network. Since constructive methods start with smaller networks, the computation time is less and they are likely to find smaller networks.

II. Determining the network architecture

The problems in backpropagation [2], [4] and the need for finding the appropriate architecture resulted in algorithms that learns the necessary architecture from data. Constructive algorithms can solve some of the problems of the standard backpropagation but also introduce some other problems. The main disadvantage of the constructive algorithms is their weakness on noisy data. The different training techniques used in constructive algorithms try to overcome these problems and to reduce both time and space complexity.

The simple way of training the new network is to train only the newly added unit and freezing the previous weights. The assumption in this approach is that, the existing units in the network are already trained and are useful in obtaining the target function. Cascade-correlation algorithm uses weight freezing in training the network [4]. Although this kind of training reduces the time and space complexity of the whole process, re-

search on weight freezing [5], [6] shows that, in general, it fails to find the desired solution. When an extra degree of freedom is introduced by adding a new unit in the network, freezing the existing weights only allows finding the solution in an affine subset of the weight space [5]. On the other hand, algorithms using weight freezing can sometimes find good solutions with a high generalization power, also with a huge decrease in the learning time.

When a new unit is added to the network, its appropriate weights are initialized to random values and the training of the whole network continues with the old weights. The idea here is that, the information learned so far will be useful in reaching the final network. Dynamic Node Creation algorithm uses this kind of training [5]. This kind of training decreases the training time but it may cause the algorithm to get stuck in local minima.

Training of the new network can be done with all the weights initialized to random values. This kind of training is the same with the standard backpropagation training. These algorithms have the advantages and disadvantages of backpropagation.

III. Constructive algorithms

An early work on the problem is Projection Pursuit Regression (PPR) [7], proposed in 1989. PPR is a statistical technique for multivariate data analysis using a two layer feedforward network with linear output units.

In Group Method of Data Handling (GMDH) type of algorithms [8] the number of incoming connections to a hidden unit is fixed but the sources of these incoming connections can change. It can be any combination of input units and other hidden units. The algorithm selects the next architecture among these different combinations. The Upstart algorithm [9] is a constructive algorithm for binary classification problems. The algorithm starts without hidden nodes and tries to separate the data. If separation is not possible, then corrector nodes are added. The generated network is very much similar to a network with one hidden layer by defining all the corrector nodes as hidden units. Cascade-correlation method [4] constructs a network with multiple hidden layers. The algorithm starts with an initial network and incrementally adds one-unit hidden layers to the network until a satisfying solution is found. The inputs are directly connected to the outputs and to the hidden units. A hidden unit is connected to the inputs, to the outputs and to all the preceeding hidden units. The calculated input side weights are also frozen. Dynamic Node Creation (DNC) [5] starts with an initial network and incrementally adds hidden nodes to the network until a satisfactory solution is found. Hidden nodes are added one at a time and to the same hidden layer. The whole network is re-trained after each hidden node addition. In DNC, a new hidden node is added to the network when the average error curve begins to flatten out too quickly. In Grow and Learn (GAL) [10] algorithm, the network grows when it learns class definitions. In the ”sleep” phase of the algorithm, the units that are no longer necessary are removed to reduce the complexity.

An algorithm that adds, deletes units and layers is proposed in 1994 [11]. The algorithm applies an intelligent generate and test procedure, explores different alternatives and selects the most promising one. A relatively new algorithm, Constructive Algorithm for Real Valued Examples (CARVE), was proposed in 1998. CARVE [12] uses convex hull methods for the determination of network weights. The algorithm starts with an empty hidden layer into which thresholds units are added one at a time until the layer is complete. Feedforward Neural Network Construction Using Cross Validation [13] uses cross validation for adding units to a single hidden layered network. The network with more hidden units is only accepted if the total accuracy on training and cross validation samples is higher than that of the previous network.

IV. MOST

Many of the algorithms proposed so far make some important assumptions on the architecture such as the number of layers. In general, an architecture with a single hidden layer is assumed and the number of nodes in the network is determined. Architectures with no hidden layers or more than one hidden layer are discarded. Cascade correlation constructs a network with more than one hidden layer but it has another assumption: each layer consists of a single hidden unit. Another problem is that comparing the error values of two networks once is unreliable. Statistical tests over multiple runs must be used for comparing two architectures. MOST uses 5×2 cv F Test [14]. Another problem is incrementing the number of hidden nodes in the network one by one since one hidden node difference can be statistically insignificant.

Constructive Algorithm with Multiple Operators using Statistical Tests (MOST) makes no assumptions on the number of layers of the network and overcomes the problem of one hidden node addition by applying multiple operators. One hidden node additions or removals are used for finetuning the network.

MOST starts with an initial network, which is a network with no hidden layers. Then it tries to apply the next applicable operator. The pseudocode of the MOST algorithm is given in Figure 1. The application of operators, in order of precedence (to prefer simple networks), is as follows:

1) Remove a percentage of hidden units from a layer.
2) Remove a hidden unit from a layer.
3) Add a hidden unit to a layer.
4) Add a percentage of hidden units to a layer.
5) Add a new layer between the output and the layer below the output. When we add a new layer, the number of hidden units in all the layers are redetermined.

An initial value for the minimum number of hidden units, *MINHIDDEN*, in a layer is specified since the performance of a linear perceptron is better in most of the problems than a multilayer perceptron with small number of hidden units. This initial value can be changed in the algorithm as a result of statistical comparisons. When we apply an operator, if the calculated value for the number of hidden units in a layer is smaller than *MINHIDDEN*, then *MINHIDDEN* is used. Similarly, a maximum number of hidden units for each layer is also specified.

When a new layer is to be added to the network, the hard point is to determine the number of hidden units in each layer. A popular heuristic is using the average of the number of hidden nodes in the upper and the lower layers. But this can lead to wrong results if the input dimension is very large or small.

We applied four heuristics in order to be able to cover different architectures in the search space.

1) Number of nodes in the upper layer
2) (Number of nodes in the upper layer) × 2
3) Average of the nodes in the upper and lower layers
4) (Average of the nodes in the upper and lower layers) / 2

Heuristics one and three are the base heuristics. Heuristics two and four are useful when heuristics one and three are either too small or too large. When adding hidden layers, we first determine the number of nodes in the newly added layer, which is the layer before the outputs. The simplest of the four heuristics is chosen and the number of hidden nodes in the preceding layer is then calculated. The algorithm starts from the first operator and applies that operator if it is applicable. If there is no hidden layer in the current network, then first four operators are not applicable. The algorithm adds a new layer and tries the above mentioned heuristics starting from the simple one to the complex one. In order to prevent the algorithm to go in a loop, if a simple architecture is selected and after then a complex one is selected due to statistical significance, then the algorithm never selects a simpler architecture again. MOST applies 5×2 cv F test to compare the current and the candidate architecture. If one of them is significantly better than the other, that network is selected. If the two networks are not significantly different, then the current network is compared with the last network before the hidden node addition. If the candidate network is significantly better, and it is simple than the current network, candidate network is selected. If there is no significant difference then selection is done by comparing the complexities of the candidate network and the network before the hidden node addition. The complexity measure is defined as the number of connections in the network (Figure 2).

1) Start with an initial network, N_1, $N_1 = LP$
2) For all applicable operators i
   - $N_2 = Operator_i(N_1)$
   - if preferable(N_2, N_1) then $N_1 = N_2$ and start new loop at step 2

Fig. 1. The MOST algorithm

Function preferable(new, current)

- C_t = candidate network, C_{t-1} = current network
- C_{t-2} = last network before the hidden layer addition
- If $E(C_t) < E(C_{t-1})$ then preferable=TRUE
- Else if $E(C_t) = E(C_{t-1})$ and $E(C_t) < E(C_{t-2})$ and $comp(C_t) < comp(C_{t-1})$ then preferable=TRUE
- Else if $E(C_t) = E(C_{t-1})$ and $E(C_t) = E(C_{t-2})$ and $comp(C_t) < comp(C_{t-2})$ then preferable=TRUE
- Else preferable=FALSE

Fig. 2. Deciding between two architectures. C_t is the network, $E(C_t)$ is the error of C_t and $comp(C_t)$ is the complexity of C_t

V. Experiments

Datasets used in experiments are given in Table I. In all regression datasets, MOST finds an architecture with three hidden units. This is because the optimal solution has three or five hidden nodes in those datasets and since the *MINHIDDEN* parameter is initially three in MOST. The search for `sin` dataset is given in Figure 3. The search trees of other regression datasets are similar.

TABLE I
DATASETS USED IN EXPERIMENTS

	Type	No. of out.	No. of inp.	Train. set	Test set
sine (artifical)	reg	1	1	500	500
california [15]	reg	1	8	10320	10320
boston [15]	reg	1	13	400	106
pum8fh [16]	reg	1	8	4096	4096
pum8nh [16]	reg	1	8	4096	4096
ocr (AT&T)	cls	10	256	600	600
optdigits [17]	cls	10	64	3823	1797
pendigits [17]	cls	10	16	7494	3498

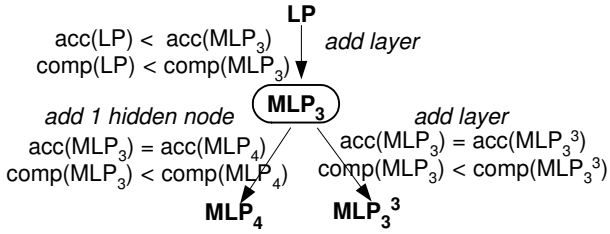

Fig. 3. MOST search tree for `sin` dataset. MOST starts with LP and applies adding a hidden layer since there is no other operator. It selects MLP with 3 hidden units as the candidate network. After appliying 5×2 cv F test, MLP3 is found to be more accurate than LP and accepted. Next operator is adding a single hidden node. MLP4 is not accepted since it is more complex than MLP3 with no improvement in the accuracy. Next operator is adding a hidden layer. MLP3-3 is not accepted with the same reasons as in MLP4 at the search steps.

In classification datasets MOST generates a larger search tree. In `ocr` and `optDigits` datasets, MOST finds an architecture as well as the optimal architecture. The search tree for `ocr` is given in Figure 4. In `penDigits` dataset, the architecture found by MOST has two hidden layers and this architecture performs better than the single hidden layer network with 20 hidden units. The results of all algorithms on all datasets are given in Table II. Newman-Keuls [18] range test is applied to these results and the test results are given in Table III.

VI. Conclusions

The optimal neural network architecture is the architecture which generalizes the underlying function of a given dataset. Too complex or too simple architectures fail to learn this underlying function. Determining the architecture by trial and error takes too much time and can eliminate some architectures that can be successful. Determining the architecture for a given problem in the learning process is the desired goal.

In MOST, there is no assumption on the number of layers in the network. The resulting network can have no, one, or many hidden layers. In choosing the candidate network, more

TABLE II

OVERALL RESULTS. NUMBER OF HIDDEN UNITS: AVERAGE ERROR ± ST. DEV.

	MLP	cascade	DNC	MOST
sine	3: 0.04± 0.12	3: 0.04± 0.00	5: 0.00± 0.00	3: 0.03± 0.13
california	10: 0.29± 0.02	4: 0.31± 0.01	15: 0.24± 0.01	3: 0.29± 0.01
boston	5: 0.77± 0.26	6: 0.32± 0.09	8: 0.84± 0.45	3: 0.97± 0.48
pum8fh	8: 0.41± 0.01	5: 0.39± 0.00	5: 0.44± 0.06	3: 0.43± 0.05
pum8nh	5: 0.38± 0.04	10: 0.34± 0.01	10: 0.37± 0.02	3: 0.42± 0.04
ocr	20: 0.03± 0.01	0: 0.04± 0.00	6: 0.12± 0.02	11: 0.04± 0.01
optdigits	20: 0.04± 0.00	7: 0.06± 0.00	12: 0.08± 0.00	21: 0.03± 0.00
pendigits	20: 0.03± 0.00	20: 0.03± 0.00	11: 0.04± 0.01	$\frac{19}{20}$: 0.02± 0.01

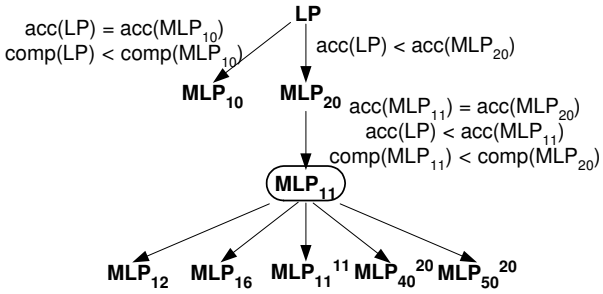

Fig. 4. MOST search tree for `ocr` dataset. MOST starts with LP and applies adding a hidden layer since there is no other operator. It selects MLP with 10 hidden units as the candidate network. MLP10 has the same accuracy as LP but with high complexity so it is not accepted. Next candidate network is MLP20. The accuracy of MLP20 is higher than LP, so it is selected. Next operator is removing a percentage of hidden nodes. MLP11 is accepted since it has the same accuracy as MLP20 and simpler and also its accuracy is higher than LP. The algorithm tries other operators but none of them is accepted.

TABLE III

NEWMAN-KEULS RANGE TEST RESULTS. A<B MEANS THAT A IS STATISTICALLY SIGNIFICANTLY HAS LESS ERROR THAN B. A=B MEANS THAT THE TWO CLASSIFIERS ARE THE SAME.

`sin`	MOST=MLP=cascade=DNC
`california`	DNC < MLP < MOST=cascade
`boston`	cascade < MOST=MLP=DNC
`pum8fh`	MOST=MLP=cascade=DNC
`pum8nh`	cascade < MLP=DNC < MOST
`ocr`	MOST=MLP=cascade < DNC
`optDigits`	MOST=MLP < cascade < DNC
`penDigits`	MOST < MLP=cascade < DNC

than one architecture is considered and the one that significantly increases the accuracy is selected, if any. The results of the algorithms are promising. MOST algorithm finds near optimal results and is useful since the only extra parameter it needs is the confidence percentage value and it can easily be set. The disadvantage of the algorithm is that the time complexity is high since it performs 5×2 cross validation for each architecture.

As future work, some other statistical tests that do not need 5×2 cross validation (e.g., McNemar's test that needs one run) can be applied in order to reduce the time complexity of the algorithm. But then the test results will be less reliable (higher type I error and less power). The effect of the confidence of the test can be examined in detail. Some other heuristics that are used to determine the candidate architectures can be applied so that the search space gets larger.

REFERENCES

[1] Geman, S., E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, Vol. 4, pp. 1–58, 1992.

[2] Kwok, T.-Y. and D.-Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems", *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, pp. 630–645, 1997.

[3] Reed, R., "Pruning algorithms – a survey", *IEEE Transactions on Neural Networks*, Vol. 4, pp. 740–747, 1993.

[4] Fahlman, S. E. and C. Leibiere, "The cascade-correlation learning architecture", *In Advances In Neural Information Processing Systems*, Vol. 2, pp. 524–532, 1990.

[5] Ash, T., "Dynamic node creation in backpropogation networks", *Connection Science*, Vol. 1, No. 4, pp. 365–375, 1989.

[6] Hwang, J., S. You, S. Lay, and I. Jou, "What's wrong with the cascade-correlation learning: A projection pursuit learning perspective", *IEEE Transactions on Neural Networks*, Vol. 7, No. 2, pp. 278–289, 1996.

[7] Friedman, J. H. and W. Stuetzle, "Projection pursuit regression", *Journal of the American Statistics Association*, Vol. 76, No. 376, pp. 817–823, 1981.

[8] Tenorio, M. F. and W. T. Lee, "Self-organizing network for optimum supervised learning", *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, pp. 100–110, 1990.

[9] Frean, M. R., "The upstart algorithm: a method for constructing and training feedforward neural networks", *IEEE Transactions on Neural Networks*, Vol. 2, pp. 198–209, 1990.

[10] n, E. A., "GAL: Networks that grow when they learn and shrink when they forget", Technical Report TR-91-032, ICSI, Berkeley, CA, 1991.

[11] Nabhan, T. M. and A. Y. Zomaya, "Toward generating neural network structures for function approximation", *Neural Networks*, Vol. 7, No. 1, pp. 89–99, 1994.

[12] Young, S. and T. Downs, "CARVE - a constructive algorithm for real valued examples", *IEEE Transactions on Neural Networks*, Vol. 9, No. 6, pp. 1180–1190, 1998.

[13] Setiono, R., "Feedforward neural network construction using cross validation", *Neural Computation*, Vol. 13, pp. 2865–2877, 2001.

[14] Alpaydın, E., "Combined 5 x 2 cv f test for comparing supervised classification learning algorithms", *Neural Computation*, Vol. 11, pp. 1885–1892, 1999.

[15] "Data, software and news from the statistics community", `http://lib.stat.cmu.edu/`.

[16] "Collections of data for developing, evaluating, and comparing learning methods", `http://www.cs.toronto.edu/~delve/`.

[17] Blake, C. L. and C. J. Merz, "UCI repository of machine learning databases", `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[18] Newman, D., "The distribution of the range in samples from a normal population, expressed in terms of an independent estimate of standard deviation", *Biometrika*, Vol. 31, pp. 20–30, 1939.