

A VARIABLE STRUCTURE LEARNING ALGORITHM FOR MULTILAYER PERCEPTRONS

KARL MATHIA

Newport Corp.

750 National Court, Richmond, California 94804

kmathia@newport.com

ABSTRACT

Choosing the 'best' neural network structure for a given problem is often a difficult task. A data-driven, automated design method has therefore been an appealing concept for years, and research resulted in network pruning as well as constructive algorithms. The Variable Structure Learning (VSL) algorithm proposed here is a fully automatic structure learning algorithm for Multilayer Perceptrons. VSL attempts to converge to the minimum complexity network for a given problem by combining both network construction and pruning concepts. A normalized and continuously monitored learning metric enables structure convergence and avoids the scaling problem.

INTRODUCTION

A common challenge with artificial neural networks (ANNs) is to choose a sufficiently complex network structure for learning a desired mapping, e.g. for function approximation, classification, etc., where 'network complexity' refers to network parameters such as the number of layers and neurons, as well as the connection pattern between layers. An ANN designed too small may not be able to learn a mapping as represented by the training data, while excess network complexity can result in overfitting and poor generalization [2]. *A Priori* knowledge about the desired mapping can point to a 'good' initial ANN structure [5][6], but may not be available.

Data-driven, automated structure learning is therefore an appealing network design concept, and past research resulted in several network pruning and network construction algorithms. See, for example, [1][3][4][8][10]. Simulations of variable structure ANNs demonstrated superior learning performance over fixed structure ANNs [7]. Based on similar ideas, the Variable Structure Learning (VSL) algorithm proposed here is a fully automatic structure learning algorithm for Multilayer Perceptrons, a popular feedforward ANN. The error backpropagation algorithm [9], which per se is not a structure learning algorithm, is utilized for weight training between structural changes.

Unlike other algorithms, VSL combines both network construction and pruning methods while maintaining a continuously decreasing learning metric (mean squared error). This is possible by adding neurons without 'forgetting' already learned knowledge. The normalized and decreasing learning metric, monitored after every learning epoch, supports the objective to converge to a small (ideally the minimum complexity) structure for learning a given problem. It also avoids the well-known scaling problem via a unified treatment of different problem complexities.

STRUCTURE LEARNING FRAMEWORK

An artificial neural network is uniquely specified by a 4-tuple of network parameters $\Psi = \{\mathcal{W}, n, g, \mathcal{G}\}$ [3][4]:

- directed connectivity graph, \mathcal{G}
- number of hidden neurons, q
- transfer function of the hidden neurons, g
- connection weights, \mathcal{W} .

Note that more than one tuple (Ψ_m, Ψ_n, \dots) can implement the same ANN function or mapping. The network parameters in Ψ are available as free variables for neural network learning. Adjusting the connection weights in a fixed network structure is most common, while structure learning usually refers to adjusting the directed connectivity graph and the number of hidden neurons. Constructive algorithms ‘grow’ an ANN, i.e. increase network complexity, while pruning algorithms ‘shrink’ it, i.e. decrease network complexity.

The goal of MLP training for regression problems is to approximate an unknown mapping or function, represented by data, via an iterative, supervised learning process. Let $f_\Psi \in \mathfrak{R}^{m \times n}$ be the function implemented by an ANN with network parameters. During learning one or more parameters are adjusted, resulting in a series of tuples Ψ_k , $k=1,2,\dots,K$, and associated ANNs

$$f_{\Psi_1}, f_{\Psi_2}, \dots, f_{\Psi_K}. \quad (1)$$

The learning process is terminated after K iterations when a success criterion $\|f_{des} - f_\Psi\| \leq \varepsilon$ is satisfied, i.e. after the learning process converged to a prespecified limit ε . Although the unknown, desired function $f_{des} \in \mathfrak{R}^{m \times n}$ is only represented by experimental data, the distance $\|\cdot\|$ can be estimated from the data. The mean squared error (MSE) in Equation 6 below between

$$d = f_{des}(x) \text{ and,} \quad (2)$$

$$y = f_\Psi(x) \quad (3)$$

is a common estimate, where $x \in \mathfrak{R}^m$ is the ANN input and $y, d \in \mathfrak{R}^n$ represent the actual and desired ANN output, respectively.

VARIABLE STRUCTURE LEARNING

Variable Structure Learning (VSL) combines both constructive and pruning methods. It adds and trains hidden neurons to an initially small ANN until a prespecified success criterion is satisfied. The network size is then minimized by successively pruning the ‘least significant’ hidden neurons while maintaining that success criterion. Here the well-known multilayer perceptron (MLP) and the error backpropagation (EBP) algorithm are used to demonstrate how VSL extends an established neural network learning algorithm to a structure learning algorithm. The VSL concept is also intended for enhancing and extending other ANN architectures and weight training algorithms.

Network Parameters and Learning Variables. Learning variables (here: VSL variables) are free network parameters which are adjusted during training in order to achieve a given learning objective. Many ANN learning algorithms,

like EBP, adjust only the connection weights. VSL trains both the connection weights and the number of hidden neurons.

Directed Connectivity Graph. The directed connectivity graph \mathbf{G} defines how all input, output, and hidden neurons are interconnected. The pruning of connection weights is one example of structure learning. Here \mathbf{G} is not considered an independent VSL variable. Rather, \mathbf{G} is a consequence of the number of neurons in the fully connected hidden layer. It was shown that one hidden layer (Figure 1) is sufficient for a universal approximator [11].

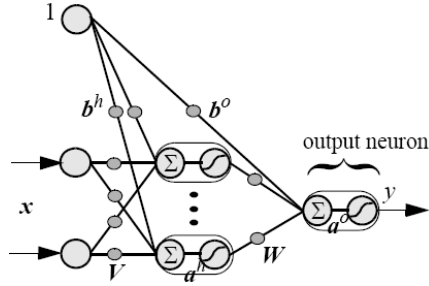


Figure 1. MLP with one hidden layer.

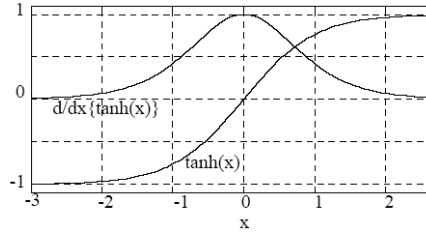


Figure 2. Hyperbolic tangent.

Number of Hidden Neurons. The number of hidden neurons is a VSL variable, while the number of input and output neurons are fixed and dictated by the data being approximated. VSL begins with one hidden neuron and adds or prunes fully connected neurons as needed. Of course, any available *a priori* knowledge about the unknown function can be used to specify the initial MLP.

Transfer Function of Hidden Neurons. Here the transfer function from activation value to neuron output is not a VSL variable and remains unchanged. For pruning the ‘least significant’ hidden neuron, i.e. the neuron with the smallest weight values, it is convenient to identify that neuron by the smallest contribution to the overall MLP mapping. A small neuron output in response to a small input is achieved using a transfer function with $g(0)=0$. Thus the hyperbolic tangent (Figure 2) replaces the common sigmoid with $g(0)=0.5$:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}, \quad (4)$$

$$\frac{d}{dx} \tanh(x) = \frac{d}{dx} \left\{ \frac{1 - e^{-2x}}{1 + e^{-2x}} \right\} = [\operatorname{sech}(x)]^2, \quad (5)$$

Connection Weights \mathbf{W} . The connection weights of neurons are probably the most common learning variable for ANNs. The VSL algorithm trains the weights of hidden and output neurons using EBP. Weight and structure updates occur sequentially and are independent (Figure 3), so a different weight training algorithm could also be utilized.

Learning Metric for Structure Learning. A critical element of the VSL algorithm is the learning metric, which is continuously decreasing if the entire training set is used for batch training [2]. Its decreasing and normalized profile,

representing learning error and learning ‘time scale’ (Figure 3) provides the information for

- deciding if and when adding or pruning neurons (evaluate learning),
- convergence of both weight and structure (when to terminate VSL),
- avoiding the well-known scaling problem (unified treatment of problems with different complexities).

The VSL and EBP learning metric is the well-known mean squared error (MSE) between desired output vector and actual ANN output vector taken over n output neurons,

$$E = \frac{1}{n} \mathbf{e}^T \mathbf{e} = \frac{1}{n} (\mathbf{d} - \mathbf{y})^T (\mathbf{d} - \mathbf{y}), \quad (6)$$

The MSE is normalized with respect to the initial (maximum) error E_{max} and monitored after every k -th epoch:

$$\bar{E}_k = \frac{E_k}{E_{max}} = \frac{E_k}{E_1}, \quad \bar{E}_k \in [0, 1]. \quad (7)$$

E_{max} is obtained after the first EBP iteration. Normalization allows to specify the success criteria, the minimum error E_{min} , as a percentage of the initial error. For similar reasons it is also convenient to normalize the epoch number with respect to the maximum number of epochs, k_{max} , specified by the user:

$$\bar{k} = \frac{k}{k_{max}}, \quad \bar{k} \in [0, 1]. \quad (8)$$

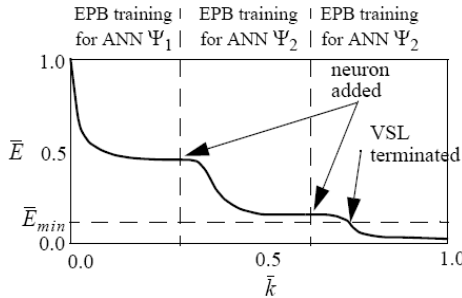


Figure 3. Example of a VSL metric.

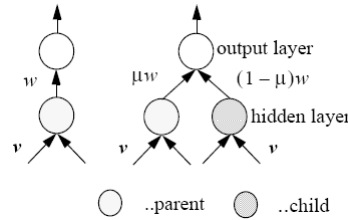


Figure 4. Adding a neuron.

Rules for Structure Learning. The VSL rules, based on the VSL metric and its rate of change $d\bar{E}/d\bar{k}$, determine when to add or prune hidden neurons. For the remainder of the paper the notation used is E and k without ‘bar’.

- **Rule 1:** the constructive phase is in progress if E is greater than E_{min} and decreasing,

$$E > E_{min} \quad \text{and} \quad \Delta E / \Delta k < 0. \quad (9)$$

Action: continue weight training.

- **Rule 2:** the ANN structure is insufficient or weight training is stuck in a local minimum if E converges to a value greater than E_{min} ,

$$E > E_{min} \quad \text{and} \quad \Delta E / \Delta k \rightarrow 0. \quad (10)$$

Action: add neuron to hidden layer.

- **Rule 3:** constructive VSL phase is successful if E drops below the desired minimum value E_{min} :

$$E < E_{min}. \quad (11)$$

Action: terminate constructive VSL phase; initiate VSL pruning phase.

- **Rule 4:** the pruning phase is in progress if

$$E < E_{min}. \quad (12)$$

Action: prune the least significant neuron; re-train all weights.

- **Rule 5:** the pruning phase is completed when E increases to

$$E > E_{min}. \quad (13)$$

Action: restore last ANN with $E < E_{min}$ and terminate VSL.

The initial VSL parameters are the initial structure and VSL limits:

- Initial ANN structure (number of hidden neurons), default is $q=1$.
- Maximum admissible number of learning epochs, k_{max} .
- Success criterion, E_{min} .
- Epoch size P for EBP learning.

The VSL algorithm determines all other parameters.

Adding Hidden Neurons. Neurons are added per Rule 2 by splitting off a ‘child’ neuron from a ‘parent’ neuron. The child inherits the connection weights such that the current ANN mapping does not change, i.e. the child is a *clone* of the parent with exception of its output weight:

$$\text{Parent: } \{(\mathbf{v}, \mathbf{b}^h), \mu w\} \quad (14)$$

$$\text{Child: } \{(\mathbf{v}, \mathbf{b}^h), (1 - \mu)w\}, \quad (15)$$

with $0 < \mu < 1$. The vector of input weights \mathbf{v} is cloned, while the sum of the parent’s and child’s output weight is equivalent to the parent’s original output weight (Figure 4). Note that, although the network parameters change ($\Psi_k \neq \Psi_{k+1}$), the associated ANN mappings do not: $\mathbf{f}_{\Psi_k} = \mathbf{f}_{\Psi_{k+1}}$.

Pruning Hidden Neurons. Hidden neurons are pruned per Rule 4. The ‘least significant’ hidden neuron, i.e. with the smallest weights, is pruned, i.e.

$$\min_{\text{hidden neurons}} \{\|\mathbf{v}\|_1 \cdot \|\mathbf{w}\|_1\}. \quad (16)$$

All other weights in the MLP remain unchanged and weight training resumes. The pruning process is repeated until the VSL metric returns to $E > E_{min}$. In this case \mathbf{f}_{Ψ_k} , the ANN at iteration k , is replaced with the previously saved $\mathbf{f}_{\Psi_{k-1}}$, the final ANN. The learning process is completed and VSL is terminated. Pruning an insignificant neuron usually has little impact on the ANN mapping, and re-training is fast and computationally inexpensive.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Table 1: XOR truth table.

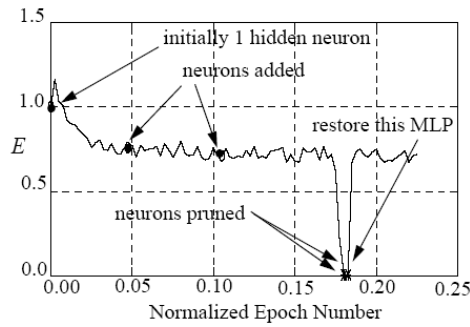


Figure 5. XOR problem: VSL metric for pattern training (not batch training).

EXAMPLES

The VSL algorithm was implemented in Matlab and tested using three function approximation problems and one classification problem.

XOR Problem. The purpose of this simple benchmark test was to verify that VSL converges to the minimum ANN structure possible. For the XOR problem this structure is well known: two hidden neurons. The truth table for the XOR problem is given in Table 1, with inputs x_1 and x_2 , and output y .

The XOR problem received remarkable attention after Minsky and Papert published ‘Perceptrons’ in 1969. The authors showed that a single perceptron *cannot* learn the XOR problem, due to a lack of linear separability of the 0’s and 1’s in the x_1/x_2 -plane. It was not until the 1980’s when it was discovered that an MLP with two hidden neurons, i.e. a 2-2-1 structure [2], can perform that task. The VSL algorithm converges to exactly this structure after six VSL iterations, as is shown in Figure 5 (VSL metric) and Figure 6.

Strain Prediction for a Titanium Workpiece. The goal for the MLP was used to predict the strain on a workpiece of titanium, based on the workpiece size, i.e. width, thickness, and volume. Other inputs are temperature, the allowable ultimate strain, and the measured strain rate. Therefore a six-to-one mapping was to be approximated. The training set contained 25 I/O vectors (Figure 7). The VSL algorithm converged to two hidden neurons, which is a surprisingly small network size. Intuitively suggested MLPs had up to two hidden layers and over 20 neurons, e.g. a 6-15-5-1 structure. This result demonstrates one of the potential benefits of structure learning: avoiding oversized neural network architectures. The learning curve is shown in Figure 8. In response to the initially flat MSE the VSL algorithm adds four neurons. It is then determined that, once the problem is learned, only two neurons are sufficient.

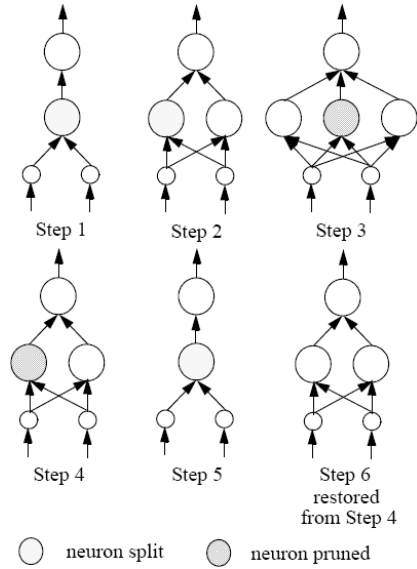


Figure 6. XOR problem: MLP structure during VSL.

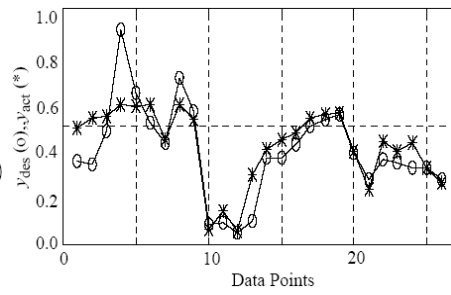


Figure 7. Titanium problem: desired (o) and learned mapping (*).

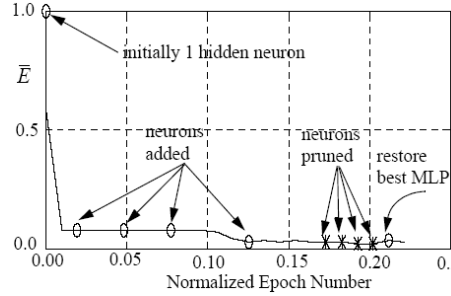


Figure 8. VSL for the Titanium problem: normalized squared error.

Classification Problem: Non-Destructive Evaluation, NDE. The non-destruction evaluation of composite materials is represented as a classification problem. A laser beam created a heat shockwave within the material. The time history of the reflected shockwave contains information about invisible damages within the material. The 16-point Fourier transform of 776 recorded signals was associated with different classes (damaged, not damaged, etc.), resulting in 776 16-to-3 mappings. The learning curve is shown in Figure 9. The changing structure during the VSL process is illustrated in Figure 10. The VSL algorithm converged to an MLP with two neurons. Intuitively a larger MLP might have been chosen by a human designer.

CONCLUSIONS

Variable Structure Learning, a data-driven, automated design method for Multilayer Perceptrons, was introduced. It was shown that VSL automates the process of choosing a ‘good’ (ideal: minimum complexity) neural network structure for a given problem. VSL starts with a small initial network and adds and prunes artificial neurons based on a success criteria monitored during the learning process. The VSL algorithm was tested with different sets of data, and it was shown that it potentially converges to the smallest structure possible. Further research will include generalization tests and comparison with other structure learning algorithms.

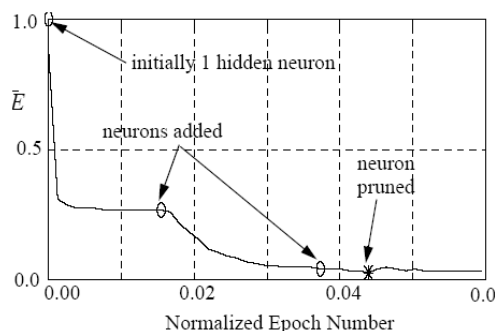


Figure 9. VSL for the NDE problem: normalized squared output error.

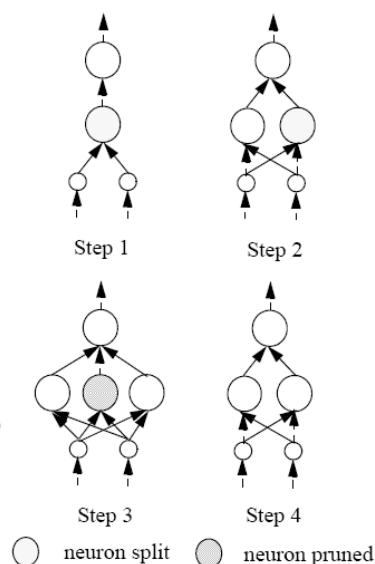


Figure 10. VSL for the NDE problem: MLP during learning.

REFERENCES

- [1] F.L. Luo, *Applied Neural Networks for Signal Processing*, Cambridge Univ. Press, Cambridge, Mass., 1999.
- [1] T. Ash, "Dynamic Node Creation in Backpropagation Networks," *Connection Science*, Vol. 1, No. 4, pp. 365-375, 1989.
- [2] S. Haykin, *Neural Networks - A Comprehensive Foundation*, IEEE Press/Macmillan College, New York, 1994.
- [3] T.Y. Kwok and D.Y. Yeung, "Objective Functions for Training New Hidden Units in Constructive Neural Networks," *IEEE Trans. Neural Networks*, 8(5), pp. 1131-1148, Sep 1997.
- [4] T.Y. Kwok and D.Y. Yeung, "Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems," *IEEE Trans. Neural Networks*, 8(3), pp. 630-645, May 1997.
- [5] G.G. Lendaris and K. Mathia, "Using A Priori Knowledge to Prestructure ANNs", *Australian Journal of Intelligent Information Processing Systems*, Vol. 1, No. 1, pp. 25-30, May 1994.
- [6] G.G. Lendaris, M. Zwick, and K. Mathia, "On Matching Network Structure to Problem Domain Structure", *Proc. World Cong. Neural Networks '93*, Portland, Oregon, pp. 488-493, July 1993.
- [7] B.Y. Pearce, "Implementation of a Variable Structure Neural Network," Masters Thesis, University of Alabama, Auburn, March 2001.
- [8] Reed R., "Pruning Algorithms - A Survey," *IEEE Transactions on Neural Networks*, Vol. 4, No. 5, pp. 740-747, 1993.
- [9] Werbos P.J., "Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. Thesis, Harvard University, Cambridge, Mass, 1974.
- [10] Y. Zhao and C.G. Atkeson, "Some Approximation Properties of Projection Pursuit Learning Networks," in *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, and R.P. Lippmann, Eds., pp.936-943, Morgan Kaufman, San Mateo, CA, 1992.
- [11] T.C. Lee, *Structure Level Adaptation for Artificial Neural Networks*, Kluwer Academic Publishers, Norwell, Massachusetts, 1991.