A Constructive Learning Algorithm for an HME

Kazumi Saito and Ryohei Nakano
NTT Communication Science Laboratories
2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
Telephone No. +81 774 95 1822, Fax No. +81 774 95 1839
e-mail {saito,nakano}@cslab.kecl.ntt.jp

ABSTRACT

A Hierarchical Mixtures of Experts (HME) model has been applied to several classes of problems, and its usefulness has been shown. However, defining an adequate structure in advance is required and the resulting performance depends on the structure. To overcome this problem, a constructive learning algorithm for an HME is proposed; it includes an initialization method, a training method and an extension method. In our experiments, which used parity problems and a function approximation problem, the proposed algorithm worked much better than the conventional method.

1. Introduction

A Hierarchical Mixtures of Experts (HME) model has been applied to several classes of problems, and its usefulness has been shown [5, 6, 10]. However, defining an adequate structure in advance is required and the resulting performance depends on the structure.

To overcome this problem, we can consider two approaches: using pruning algorithms [8] or constructive algorithms [7]. Pruning algorithms will play an important role in improving generalization performance of networks, particularly for problems that involve many irrelevant input variables. However, a large amount of computation time must be spent to train larger than desirable initial networks; moreover, we generally do not know what size is suitable for an initial network. Compared to pruning algorithms, we believe that constructive algorithms are more suitable for the present problem. Although several algorithms have been proposed for feed-forward networks or classification trees (e.g.[3, 1]), little work has been done in an HME context.

This paper proposes a constructive learning algorithm for HME. Section 2 explains the proposed algorithm, which includes an initialization method, a training method and an extension method. In Section 3 the proposed method is evaluated based on experiments that used parity problems and a function approximation problem.

2. Constructive Algorithm

2.1. Basic definition

Let $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$ be a set of examples, where $\mathbf{x}_t$ denotes an n-dimensional input vector and y_t a target value corresponding to $\mathbf{x}_t$. An HME consists of several Expert Networks (ENs) and Gating Networks (GNs) expressed as a tree where leaves are ENs and the other nodes are GNs. In this paper, we only consider binary HMEs, because an arbitrary tree can be reduced to a functionally equivalent binary tree. Hereafter, the weight vector of EN_i is expressed by $\mathbf{w}_i = (w_{i0}, \cdots, w_{in})^T$, and its output value is defined by $u_i(\mathbf{x}_t, y_t; \mathbf{w}_i) = \exp\bigl(-\frac{1}{2}(y_t - \mathbf{w}_i^T \mathbf{x}_t)^2\bigr)$, where $\mathbf{a}^T$ means the transposed vector of $\mathbf{a}$. The weight vector of GN_i is expressed by $\mathbf{v}_i = (v_{i0}, \cdots, v_{in})^T$, and since the soft-max function reduces to the sigmoidal function in a binary HME, the output value of GN_i can be defined by $g_i(\mathbf{x}_t; \mathbf{v}_i) = \bigl(1 + \exp(-\mathbf{v}_i^T \mathbf{x}_t)\bigr)^{-1}$. Note that w_{i0} and v_{i0} mean bias terms and x_{t0} is always set to 1. The simplest HME consisting of EN_1, EN_2, and

GN_1 is expressed as (GN_1, EN_1, EN_2), and its output value is defined by $g_1 u_1 + (1 - g_1)u_2$. Then, an arbitrary HME can be expressed by a corresponding list structure such as (GN_1, EN_1, (GN_2, EN_2, EN_3)), and its output value can be recursively defined by $g_1 u_1 + (1 - g_1)(g_2 u_2 + (1 - g_2)u_3)$.

2.2. Relation matrix and objective function

In order to describe the structure of variously extended HMEs, we introduce a matrix $\mathbf{R} = (r_{ij})$; here, r_{ij} represents a relation between EN_i and GN_j. An element only takes one of three values $\{1, -1, 0\}$: $r_{ij} = 1$ indicates that the product term $g_j u_i$ appears in the HME; $r_{ij} = -1$ indicates that the term $(1 - g_j)u_i$ appears; $r_{ij} = 0$ indicates that EN_i and GN_j have nothing to do with each other. For example, the relation matrix for (GN_1, EN_1, EN_2) is $\mathbf{R}^{(2)} = (1, -1)^T$, and the relation matrix for (GN_1, EN_1, (GN_2, EN_2, EN_3)) is $\mathbf{R}^{(3)} = ((1, -1, -1)^T, (0, 1, -1)^T)$.

Hereafter, for the HME$^{(c)}$ whose number of ENs is c, a vector consisting of all weight vectors is simply expressed as $\mathbf{\Phi}^{(c)} = (\mathbf{w}_1^T, \cdots, \mathbf{w}_c^T, \mathbf{v}_1^T, \cdots, \mathbf{v}_{c-1}^T)^T$, where $N^{(c)} = (2c - 1)(n + 1)$ is the dimension of $\mathbf{\Phi}^{(c)}$. Then, by referring to the relation matrix $\mathbf{R}^{(c)}$, the product term with respect to EN_i can be expressed as

$$h_i(\mathbf{x}_t, y_t; \mathbf{\Phi}^{(c)}) \;=\; u_i(\mathbf{x}_t, y_t; \mathbf{w}_i) \times \prod_{\{j | r_{ij}^{(c)} = 1\}} g_j(\mathbf{x}_t; \mathbf{v}_j) \times \prod_{\{j | r_{ij}^{(c)} = -1\}} (1 - g_j(\mathbf{x}_t; \mathbf{v}_j)). \tag{1}$$

Now, the objective function of the HME$^{(c)}$ can be defined by the logarithmic likelihood

$$L(\mathbf{\Phi}^{(c)}) \;=\; \sum_{t=1}^{m} \log \left(\sum_{i=1}^{c} h_i(\mathbf{x}_t, y_t; \mathbf{\Phi}^{(c)}) \right). \tag{2}$$

Thus, the maximum likelihood estimation problem is finding the $\mathbf{\Phi}^{(c)}$ that maximizes $L(\mathbf{\Phi}^{(c)})$.

2.3. Main algorithm

The basic idea of the proposed method is that after training the HME$^{(c)}$, the EN_b with the largest error is selected among the EN_i, $i = 1, \cdots, c$; then, by replacing EN_b with (GN_c, EN_b, EN_{c+1}), an extended HME$^{(c+1)}$ is constructed. This is based on the "divide-and-conquer" approach employed by existing methods such as CART [1]. The main algorithm is described as follows:

step 1: Initialize $\mathbf{\Phi}^{(2)}$ ($\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{v}_1$), and set $\mathbf{R}^{(2)} = (1, -1)^T$, $c = 2$;
step 2: Train all weights included in the HME$^{(c)}$;
step 3: Terminate the iteration if a stopping criterion is satisfied;
step 4: Select an EN_b to be extended, calculate $\mathbf{R}^{(c+1)}$, and initialize $\mathbf{w}_{c+1}$ and $\mathbf{v}_c$;
step 5: Set $c = c + 1$, and return to **Step 2**;

2.4. Initialization method (Step 1)

In our preliminary experiments using parity problems, when the initial values for $\mathbf{\Phi}^{(2)}$ were set to random values near 0, most trials converged to $\hat{\mathbf{\Phi}}^{(2)}$ ($\mathbf{w}_1 = \mathbf{w}_2 = \hat{\mathbf{w}}$, $\mathbf{v}_1 = \mathbf{0}$), where $\hat{\mathbf{w}}$ is the solution of a regression problem: find $\hat{\mathbf{w}}$ that minimizes $\sum_{t=1}^{m}(y_t - \mathbf{w}^T\mathbf{x}_t)^2$. Moreover, even though a small vector $\Delta\mathbf{\Phi}$ was added to $\hat{\mathbf{\Phi}}^{(2)}$, it usually held that $L(\hat{\mathbf{\Phi}} + \Delta\mathbf{\Phi}) < L(\hat{\mathbf{\Phi}})$, and in most trials $\mathbf{\Phi}^{(2)}$ came back to the same point $\hat{\mathbf{\Phi}}^{(2)}$.

Here, we analyze how this occurs by investigating the Hessian matrix $\nabla^2 L(\mathbf{\Phi}_0^{(2)})$. In general, we can assume that $\sum_{t=1}^{m} \mathbf{x}_t\mathbf{x}_t^T$ is positive definite. Also note that $\sum_{t=1}^{m}(y_t - \hat{\mathbf{w}}^T\mathbf{x}_t)\mathbf{x}_t = 0$. When $\hat{\mathbf{\Phi}}^{(2)}$ is set as $\mathbf{w}_1 = \mathbf{w}_2 = \hat{\mathbf{w}}$ and $\mathbf{v}_1 = \mathbf{0}$, we can see that $\nabla L(\hat{\mathbf{\Phi}}^{(2)}) = \mathbf{0}$, because

$$\begin{aligned}
\nabla_{\mathbf{w}_i} L(\hat{\mathbf{\Phi}}^{(2)}) &= \sum_{t=1}^{m} \frac{h_i(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})(y_t - \mathbf{w}_i^T\mathbf{x}_t)}{\sum_{k=1}^{2} h_k(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})}\mathbf{x}_t \;=\; \frac{1}{2}\sum_{t=1}^{m}(y_t - \hat{\mathbf{w}}^T\mathbf{x}_t)\mathbf{x}_t \;=\; \mathbf{0}, \\
\nabla_{\mathbf{v}_1} L(\hat{\mathbf{\Phi}}^{(2)}) &= \sum_{t=1}^{m} \frac{h_1(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})(1 - g_1(\mathbf{x}_t; \mathbf{v}_1)) - h_2(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})g_1(\mathbf{x}_t; \mathbf{v}_1)}{\sum_{k=1}^{2} h_k(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})}\mathbf{x}_t \;=\; \mathbf{0},
\end{aligned}$$

where $h_1(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)}) = h_2(\mathbf{x}_t, y_t; \hat{\mathbf{\Phi}}^{(2)})$ and $g_1(\mathbf{x}_t; \mathbf{v}_1) = 0.5$ for $\hat{\mathbf{\Phi}}^{(2)}$. On the other hand, by calculating second-order derivatives, we can see that $\nabla^2 L(\mathbf{\Phi}^{(2)})$ usually becomes semi-negative definite, when the target values are normalized in the range of $[0, 1]$. Thus, $\mathbf{\Phi}^{(2)}$ is a weak local maximum.

Fortunately, we can easily escape from this weak local maximum by using $\Delta\mathbf{\Phi}$ such that $\Delta\mathbf{w}_1 = \Delta\mathbf{w}_2 = \mathbf{0}$ and $\Delta\mathbf{v}_1$ is set to a random vector. Here, in order to randomly scatter the value of $g_1(\mathbf{x}_t; \mathbf{v}_1)$ around 0.5, each value of $\{v_{11}, \cdots, v_{1n}\}$ is set to a random value in the range of $[-1, 1]$, and v_{10} is set to $-\sum_{i=1}^{n} v_{1i} \left(\sum_{t=1}^{m} x_{ti}\right)$. That is, $\mathbf{v}_1$ becomes a random hyper-plane that includes the gravity of input vectors; training examples will be divided into two classes each of which includes much the same number of examples. Then, $L(\hat{\mathbf{\Phi}}) = L(\hat{\mathbf{\Phi}} + \Delta\mathbf{\Phi})$, $\nabla_{\mathbf{w}_1} L(\hat{\mathbf{\Phi}} + \Delta\mathbf{\Phi}) = \sum_{t=1}^{m} g_1(\mathbf{x}_t; \mathbf{v}_1)(y - \hat{\mathbf{w}}^T \mathbf{x}_t)\mathbf{x}_t \neq 0$, and the value of the objective function can be steadily increased in the next iteration.

2.5. Training method (Step 2)

2.5.1. Second-order learning algorithm

The conventional learning algorithm for an HME [6, 10] is based on the Expectation Maximization (EM) algorithm [2]. However, since this algorithm requires Iterative Reweighted Least Squares (IRLS) in the M-step, numerical instabilities are likely [10]. Since our preliminary experiments also suffered from this problem, we employed a second-order learning algorithm [9] based on a quasi-Newton method [4].

In this algorithm, the following are repeated until convergence: after calculating the gradient vector, the search direction ($\Delta\mathbf{\Phi}$) is calculated on the basis of the partial BFGS (Broydon-Fletcher-Goldfarb-Shanno) update; the optimal step-length λ that maximizes $L(\mathbf{\Phi} + \lambda\Delta\mathbf{\Phi})$ is calculated as the maximal point of a second-order approximation. Below, we show that the optimal step-length can be efficiently calculated for the HME$^{(c)}$ networks defined in Section 2.2.

Since λ is the only variable in $L(\cdot)$, we can express $L(\mathbf{\Phi} + \lambda\Delta\mathbf{\Phi})$ simply as $\zeta(\lambda)$; then the maximal point of a second-order approximation is given when $\lambda = -\zeta'(0)/\zeta''(0)$. By differentiating $\zeta(\lambda)$ and substituting 0 for λ, we obtain

$$\zeta'(0) = \sum_{t=1}^{m} \frac{\sum_{i=1}^{c} h_i'(\mathbf{x}_t, y_t; \mathbf{\Phi})}{\sum_{i=1}^{c} h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})}, \qquad \zeta''(0) = \sum_{t=1}^{m} \left\{ -\left(\frac{\sum_{i=1}^{c} h_i'(\mathbf{x}_t, y_t; \mathbf{\Phi})}{\sum_{i=1}^{c} h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})} \right)^2 + \frac{\sum_{i=1}^{c} h_i''(\mathbf{x}_t, y_t; \mathbf{\Phi})}{\sum_{i=1}^{c} h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})} \right\}.$$

Now that the derivative of $h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})$ is defined as $\frac{d}{d\lambda} h_i(\mathbf{x}_t, y_t; \mathbf{\Phi} + \lambda\Delta\mathbf{\Phi})_{|\lambda=0}$, we obtain

$$h_i'(\mathbf{x}_t, y_t; \mathbf{\Phi}) = h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})\alpha_i, \qquad h_i''(\mathbf{x}_t, y_t; \mathbf{\Phi}) = h_i'(\mathbf{x}_t, y_t; \mathbf{\Phi})\alpha_i + h_i(\mathbf{x}_t, y_t; \mathbf{\Phi})\beta_i.$$

where

$$\begin{aligned} \alpha_i &= (y - \mathbf{w}_i^T \mathbf{x}_t)\Delta\mathbf{w}_i^T \mathbf{x} + \sum_{\{j|r_{ij}=1\}} \Delta\mathbf{v}_j^T \mathbf{x} - \sum_{\{j|r_{ij}\in\{1,-1\}\}} g_j(\mathbf{x}_t; \mathbf{v}_j)\Delta\mathbf{v}_j^T \mathbf{x} \\ \beta_i &= -(\Delta\mathbf{w}_i^T \mathbf{x})^2 - \sum_{\{j|r_{ij}\in\{1,-1\}\}} g_i(\mathbf{x}_t; \mathbf{v}_j)(1 - g_j(\mathbf{x}_t; \mathbf{v}_j))(\Delta\mathbf{v}_j^T \mathbf{x})^2. \end{aligned}$$

Here, recall that $\Delta\mathbf{w}_i$ and $\Delta\mathbf{v}_i$ mean the search directions with respect to $\mathbf{w}_i$ and $\mathbf{v}_i$, respectively.

2.5.2. Weight reduction

As construction of the HME proceeds, the magnitude of weight vectors in gating networks generally becomes very large; since these output values approach 0 or 1, their derivatives almost become 0. Since these weights can hardly be modified, the magnitude of such weights should be reduced; this reduction is equivalent to decreasing the gain (or slope) of the sigmoid nonlinearity. In our preliminary experiments, when the magnitude of all gating weights was simultaneously reduced, a small reduction did not have much effect, but a large reduction was likely to destroy what the HME had learned so far. Since it is difficult to know an adequate reduction in advance, we employed a method where the weight reduction was performed separately. After training the HME$^{(c)}$, the HME$^{(c)}$ is trained again by $\mathbf{v}_k^{(new)} = \gamma\mathbf{v}_k^{(old)}$. This process is iterated from $k = 1$ to $k = c - 1$. In our experiments, γ was set to 0.1.

Table: 1: Constructive HME for parity problems

parity bits	4	5	6	7	8
converged trials (out of 100)	100	100	100	100	99
average number of ENs	3.07	3.85	4.01	4.20	5.23
standard deviation number of ENs	0.26	0.43	0.10	0.47	0.75
minimal number of ENs	3	3	4	4	5

2.6. Extension method (Step 4)

In order to extend the HME$^{(c)}$, the EN_i that maximizes the following formula is selected:

$$\mathrm{Err}(EN_i) = \sum_{t=1}^{m}(y_t - \mathbf{w}_i^T\mathbf{x}_t)^2 \times \prod_{\{j|r_{ij}^{(c)}=1\}} g_j(\mathbf{x}_t;\mathbf{v}_j) \times \prod_{\{j|r_{ij}^{(c)}=-1\}} (1 - g_j(\mathbf{x}_t;\mathbf{v}_j)). \quad (3)$$

In this formula, the first squared value represents the error of each example, and the remaining product term represents the probability that the example is assigned to EN_i; thus, the total summation can be regarded as the expected error of EN_i. The EN_b with the largest error is selected.

The values of $\mathbf{R}^{(c+1)}$ are determined as follows. First, since EN_{c+1} is located under the former location of EN_b, for $1 \leq j \leq c-1$, each element $r_{c+1,j}$ is set equal to r_{bj}. Then, since GN_c takes effect only on EN_b and EN_{c+1}, $r_{bc} = 1$, $r_{c+1,c} = -1$, and for $i \neq b$, each element r_{ic} is set equal to 0.

The values of $\mathbf{w}_{c+1}$ and $\mathbf{v}_c$ are initialized using the weight initialization method described above; namely, $\mathbf{w}_{c+1} = \mathbf{w}_b$, each value of $\{v_{c1}, \cdots, v_{cn}\}$ is set to a random value in the range of $[-1, 1]$, and v_{c0} is set to

$$v_{c0} = -\sum_{i=1}^{n} v_{ci} \left(\sum_{t=1}^{m} x_{ti} \times \prod_{\{j|r_{bj}^{(c)}=1\}} g_j(\mathbf{x}_t;\mathbf{v}_j) \times \prod_{\{j|r_{bj}^{(c)}=-1\}} (1 - g_j(\mathbf{x}_t;\mathbf{v}_j)) \right).$$

Here, $\mathbf{v}_c$ becomes a random hyper-plane that includes the gravity of weighted input vectors with respect to EN_b.

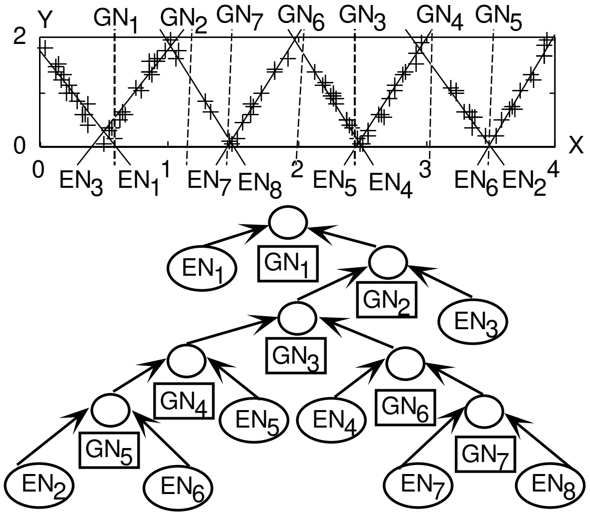
3. Evaluation by experiments

3.1. Parity problems

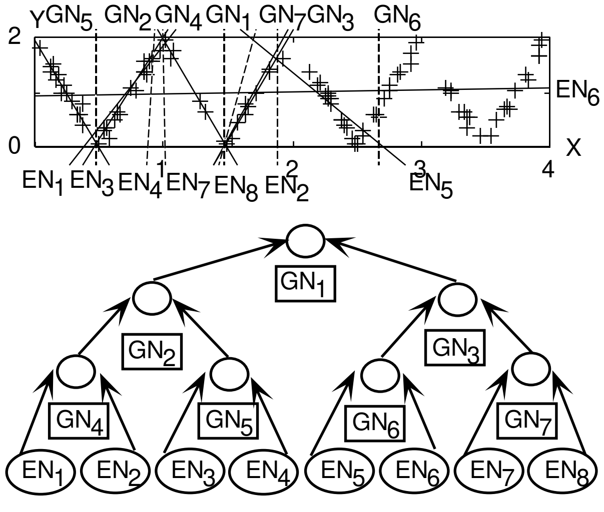
By using 4- to 8-bit parity problems, the proposed method was evaluated. In this experiment, all possible input patterns were used as training examples, the target values were set to 0 or 1, and the maximum number of ENs was set to 8. The iteration of **Step 2** was terminated when $\|\nabla L(\mathbf{\Phi}^{(c)})\|/N^{(c)} < 10^{-8}$, while the iteration of the main loop was terminated when each example satisfied $\sum_{i=1}^{c} \mathrm{Err}(EN_i) < 0.1$. Table 1 shows the learning results; trials were performed 100 times for each number of bits. The minimum number of ENs to solve an n-bit parity problem is given by $\lfloor n/2 \rfloor + 1$. Table 1 shows that most trials of the proposed method converged by using the minimum number of ENs. The conventional method [10] required many more ENs to solve these parity problems. For example, in the case of the 8-bit parity problem, it required 64 ENs in order to achieve a success rate of 67%.

3.2. Function approximation problem

Consider a piecewise linear function that consists of several lines whose slope is -4 or 4; these lines are alternately connected to each other from $(x, y) = (0, 2)$ to $(4, 2)$, x is an input value and y is a target output value. In this experiment, the value of x was randomly generated in the range of $[0, 4]$, and the corresponding value of y was calculated from x, where each value of y was corrupted by adding noise generated according



(a) Result of constructive algorithm



(b) Results of 3-level balanced binary tree

Fig. 1: HMEs for function approximation

to a normal distribution with a mean of 0 and a variance of 0.1. The total number of training examples was set to 100, and the same stopping criteria as in the previous experiment was employed. In all 10 trials, our constructive algorithm found desirable results. An example of the learning results is shown in Figure 1(a). Its structure is expressed by

$$(GN_1, EN_1, (GN_2, (GN_3, (GN_4, (GN_5, EN_2, EN_6), EN_5), (GN_6, EN_4, (GN_7, EN_7, EN_8)), EN_3))).$$

On the other hand, when a 3-level balanced binary tree expressed by

$$(GN_1, (GN_2, (GN_4, EN_1, EN_2), (GN_5, EN_3, EN_4)), (GN_3, (GN_6, EN_5, EN_6), (GN_7, EN_7, EN_8))),$$

was given in advance, we were not able to obtain a reasonable result during 10 trials. Figure 1(b) shows an example of the learning results. In order to obtain a desirable result using the 3-level balanced binary tree, the decision boundary of GN_1 must be $x = 2$. Actually, in the case of Figure 1(b), the decision boundary of GN_1 was $x \approx 1.5$; thus, redundant ENs appeared when $x < 1.5$. Conversely, more ENs were required when $x > 1.5$. When the structure of the HME is fixed in advance, learning will be more difficult because the decision boundaries of several GNs are predetermined. Figure 2 shows the learning process of our constructive algorithm; although the decision boundary of GN_1 was $x \approx 0.5$, our constructive algorithm was able to obtain a desirable result because new GNs with adequate boundaries were successively generated.

4. Conclusion

We have proposed a constructive learning algorithm for HMEs that includes an initialization method, a training method and an extension method. In experiments using parity problems and a function approximation problem, the proposed algorithm worked better than the conventional algorithm in respect that our algorithm obtained the desirable results with the minimum size of HME. In the future, we plan to do further comparisons using a wider variety of problems.

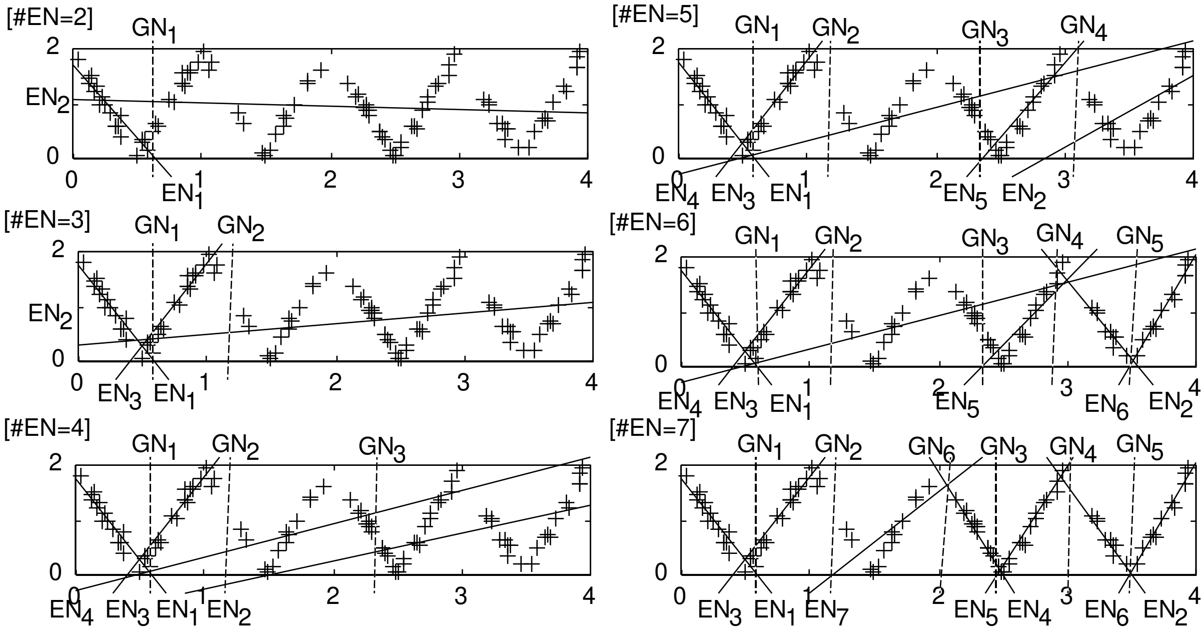


Fig. 2: Learning process of constructive algorithm

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Montrey, CA: Wadsworth International Group, 1984.

[2] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via EM algorithm," *J. Royal Statist. Soc. Ser. B (methodology)*, vol. 39, pp. 1–38, 1977.

[3] S. Fahlman and C. Libiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2* (D. Touretzky, ed.), (Los Altos, CA), pp. 524–532, Morgan Kaufmann, 1990.

[4] P. Gill, W. Murray, and M. Wright, *Practical optimization*. London: Academic Press, 1981.

[5] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[6] M. Jordan and R. Jacobs, "Hierachical mixtures of experts and EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[7] T. Kwok and D. Teung, "Constructive feedforward neural networks for regression problems: a survey," tech. rep., HKUST-CS95-43, 1995.

[8] R. Reed, "Pruning algorithms - a survey," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[9] K. Saito and R. Nakano, "A connectionist approach to numeric law discovery," in *Machine Intelligence*, vol. 15, 1995 (to appear).

[10] S. Waterhouse and A. Robinson, "Classification using hierarchical mixtures of experts," in *Proc. of NNSP*, pp. 177–186, 1994.