Bidirectional Clustering of Weights for Finding Succinct Multivariate Polynomials

Yusuke Tanahashi and Ryohei Nakano

Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466–8555 Japan

Summary
We present a weight sharing method called BCW and evaluate its effectiveness by applying it to find succinct multivariate polynomials. In our framework, polynomials are obtained by learning a three-layer perceptron for data having only numerical variables. If data have both numerical and nominal variables, we consider a set of nominally conditioned multivariate polynomials. To obtain succinct polynomials, we focus on weight sharing. Weight sharing constrains the freedom of weight values such that weights are allowed to have one of common weights. The BCW employs merge and split operations based on 2nd-order optimal criteria, and can escape local optima through bidirectional clustering. Moreover, the BCW selects the optimal model based on the Bayesian Information Criterion (BIC). Our experiments showed that connectionist polynomial regressions with the proposed BCW can restore succinct polynomials almost equivalent to the original for artificial data sets, and obtained readable results and satisfactory generalization performance better than other methods for many real data sets.

Key words:
polynomial regression, multi-layer perceptron, weight sharing, bidirectional clustering

1. Introduction

Polynomials are quite suitable for expressing underlying regularities among multiple variates. Previously, BACON system [1] was proposed by Langley, which finds polynomials by a combinatorial searching approach through a trial and error process. However, it suffers from a combinatorial explosion in the case that data have a lot of variables. As an alternative, a connectionist numerical approach, such as *RF5* [2], has been investigated. The RF5 employs a three-layer perceptron to discover the optimal polynomial which fits multivariate data having only numerical variables, and it worked well. For data having both numerical and nominal variables, a set of nominally conditioned polynomials is considered as a fitting model. In this model, each polynomial is accompanied with the corresponding nominal condition stating a subspace where the polynomial is applied. A method called *RF6.4* [3] was proposed to find such a set of nominally conditioned polynomials. The RF6.4 proceeds in two steps: learning of a multi-layer perceptron, and rule restoration from the learned perceptron.

These polynomial models have enough power to fit nonlinear data well, but the readability of the final output is not so good when data have many variables. For the purpose of obtaining succinct polynomials, we focus on *weight sharing* [4][5]. Weight sharing means constraining the freedom of weight values such that weights in a network are divided into several clusters and weights within the same cluster have the same value called a *common weight*. If a common weight value is very close to zero, then all the corresponding weights can be removed as irrelevant to result in disconnection, which is called *weight pruning*. If we employ weight sharing and pruning, we will obtain as simple polynomials as possible, which greatly improves the readability in knowledge discovery from data.

Weight sharing and pruning have been widely used to reduce the effective complexity of a network. Nowlan and Hinton proposed *soft weight sharing* [6][7], and soft weight sharing was applied to multi-layer perceptrons for efficient VLSI implementation [8]. LeCun el al. proposed *OBD* (*optimal brain damage*) [9] for removing unimportant weights from a network by using the Hessian matrix. Moreover, weight sharing was used to learn artificial neural network architectures for othello evaluation [10] focusing its symmetries.

Since we pursue crisp readability, we have investigated a hard weight sharing, and this paper proposes *BCW (bidirectional clustering of weights)*. The BCW employs both cluster-merge and cluster-split operations based on second-order optimal criteria, and can escape local optima to find global optima or semi-optima through bidirectional operations. When we apply the BCW to RF5 or RF6.4, we should determine the vital parameters: the numbers J, R of hidden units, the number I of rules, and the number G of clusters for the weight sharing. As a criterion to select the best model, the BCW employs the information criterion called BIC [11] for selecting optimal J^*, R^*, I^* and G^*. Since BIC doesn't need repetitive learning, the BCW can select the optimal model very fast.

Section 2 explains the basic framework of the connectionist polynomial regressions RF5 and RF6.4.

Manuscript received May 5, 2008
Manuscript revised May 20, 2008

Section 3 explains the proposed BCW in detail. Section 4 explains how to apply the BCW to connectionist polynomial regressions. Section 5 evaluates how the BCW worked for finding succinct multivariate polynomials through our experiments.

2. Connectionist Polynomial Regression

2.1 Multivariate Polynomial Regression

We explain the basic framework of a connectionist multivariate polynomial regression method RF5 [2]. We consider a regression problem to find the following polynomial made of multiple variables. Here $\boldsymbol{x}$ = $(x_1,\ldots,x_k,\ldots,x_K)$ is a vector of numerical explanatory variables.

$$f(\boldsymbol{x};\boldsymbol{w}) = w_0 + \sum_{j=1}^{J} w_j \prod_{k=1}^{K} x_k^{w_{jk}} \tag{1}$$

By assuming $x_k > 0$, we rewrite it as follows.

$$f(\boldsymbol{x};\boldsymbol{w}) = w_0 + \sum_{j=1}^{J} w_j \exp\left(\sum_{k=1}^{K} w_{jk} \ln x_k\right) \tag{2}$$

Here, $\boldsymbol{w}$ is composed of w_0, w_j and w_{jk}. The right hand side can be regarded as a feedforward computation of a three-layer perceptron having J hidden units with w_0 as a bias term, which is shown in Fig. 1. Note that an activation function of a hidden unit is exponential. A regression problem requires us to estimate $f(\boldsymbol{x};\ \boldsymbol{w})$ from training data $\{(\boldsymbol{x}^{\mu}, y^{\mu}) : \mu = 1,\ldots,N\}$, where y denotes a numerical target variable. The following mean squared error (*MSE*) is employed as an error function.

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{\mu=1}^{N}\left(f(\boldsymbol{x}^{\mu};\boldsymbol{w}) - y^{\mu}\right)^2 \tag{3}$$

To obtain good results efficiently in our learning, we use the BPQ method [12], which has the quasi-Newton framework with the BFGS update and calculates the step-length by using the second-order approximation.

2.2 Nominally Conditioned Polynomial Regression

If data have both numerical and nominal variables, we can consider a set of nominally conditioned multivariate polynomials. RF6.4 [3] can find such a set of polynomials. Let $(q_1,\ldots,q_{K_1},x_1,\ldots,x_{K_2},y)$ or ($\boldsymbol{q}$, $\boldsymbol{x}$, y) be a vector of variables, where q_k and x_k are nominal and numerical explanatory variables respectively, and y is a numerical target variable. For each q_k we introduce a *dummy variable* q_{kl} defined as follows: $q_{kl} = 1$ if q_{kl} matches the l-th category of q_k, and $q_{kl} = 0$ otherwise. Here, $l = 1,\ldots,L_k$, and L_k is the number of distinct categories appearing in q_k.

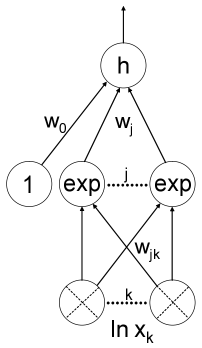

Figure 1:Three-layer perceptron for RF5

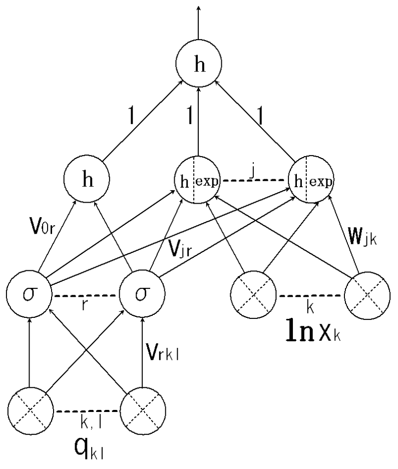

Figure 2: Four-layer perceptron for RF6.4

As a true model governing data, we consider the following set of *I rules*.

(4)

where Q_k^i and $\boldsymbol{w}^i$ denote a set of q_{kl} and a parameter vector of $\phi(\boldsymbol{x},\boldsymbol{w}^i)$ respectively used in the i-th rule. As a class of regression function $\phi(\boldsymbol{x},\boldsymbol{w}^i)$, we consider the following multivariate polynomial. Thus, a rule is nothing but a nominally conditioned polynomial.

$$\phi(\boldsymbol{x};\boldsymbol{w}^i) = w_0^i + \sum_{j=1}^{J} w_j^i \exp\left(\sum_{k=1}^{K_2} w_{jk}^i \ln x_k\right) \tag{5}$$

Equation (5) can be represented by the following single numerical function, which can be learned by using a single four-layer perceptron as shown in Fig. 2.

$$\begin{aligned} f(\boldsymbol{q},\boldsymbol{x};\boldsymbol{\theta}) &= c_0 + \sum_{j=1}^{J} c_j \exp\left(\sum_{k=1}^{K_2} w_{jk} \ln x_k\right) \\ c_0 &= \sum_{r=1}^{R} v_{0r}\sigma_r, \quad c_j = \sum_{r=1}^{R} v_{jr}\sigma_r \\ \sigma_r &= \sigma\left(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} v_{rkl}\, q_{kl}\right) \end{aligned} \tag{6}$$

Here, $\boldsymbol{\theta}$ is composed of v_{0r}, v_{jr}, v_{rkl} and w_{jk}.

Now we assume we have finished learning the perceptron. Then, a set of rules is extracted from the learned perceptron. As the first step of rule restoration, coefficient vectors for all samples $\{\boldsymbol{c}^\mu = (c_0^\mu, c_1^\mu, ..., c_J^\mu) : \mu = 1, ..., N\}$ are quantized into I representatives $\{\boldsymbol{a}^i = (a_0^i, a_1^i, ..., a_J^i) : i = 1, \ldots, I\}$. For vector quantization we employ the K-means due to its simplicity, to obtain I disjoint subsets $\{G_i : i = 1,\ldots,I\}$ where the distortion d_{VQ} is minimized.

$$d_{VQ} = \sum_{i=1}^{I} \sum_{\mu \in G_i} \|\boldsymbol{c}^\mu - \boldsymbol{a}^i\|^2, \quad \boldsymbol{a}^i = \frac{1}{N_i} \sum_{\mu \in G_i} \boldsymbol{c}^\mu \tag{7}$$

As the second step, the final rules are obtained by solving a simple classification problem whose training samples are $\{(\boldsymbol{q}^\mu, i(\boldsymbol{q}^\mu)) : \mu = 1, \ldots, N\}$, where $i(\boldsymbol{q}^\mu)$ indicates the representative label of the μ-th sample. Here we employ the C4.5 decision tree generation program [13] due to its wide availability.

3 Proposed Method of Weight Sharing

3.1 Basic Definitions

Let $E(\boldsymbol{w})$ be an error function to minimize, where $\boldsymbol{w} = (w_1, \ldots, w_d, \ldots, w_D)$ denotes a vector of D weights in a multi-layer perceptron. Then, we define a set of G clusters $\Omega(G) = \{S_1,\ldots,S_g,\ldots,S_G\}$, where S_g denotes a set of weights such that $S_g \neq \phi$, $S_g \cap S_{g'} = \phi \ (g \neq g')$ and $S_1 \cup ... \cup S_G = \{1, ..., D\}$. Also, we define a vector of G common weights $\boldsymbol{u} = (u_1,\ldots, u_g,\ldots, u_G)$ associated with a cluster set $\Omega(G)$ such that $w_d = u_g$ if $d \in S_g$.

Now we consider a relation between $\boldsymbol{w}$ and $\boldsymbol{u}$. Let $\boldsymbol{e}_d^D$ be the D-dimensional unit vector whose elements are all zero except for the d-th element, which is equal to unity. Then the original weight vector $\boldsymbol{w}$ can be expressed by using a $D \times G$ transformational matrix $\boldsymbol{A}$ as follows.

$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{u}, \quad \boldsymbol{A} = \left[\sum_{d \in S_1} \boldsymbol{e}_d^D, ..., \sum_{d \in S_G} \boldsymbol{e}_d^D\right] \tag{8}$$

Note that we have a one-to-one mapping between the matrix $\boldsymbol{A}$ and the cluster set $\Omega(G)$. Therefore, our goal is to find $\Omega(G^*)$ which minimizes $E(\boldsymbol{A}\boldsymbol{u})$, where G^* denotes the optimal number of clusters.

In our method, we employ the BPQ method to learn the perceptron with clustered weights. So we need to calculate a gradient vector $\boldsymbol{g}(\tilde{\boldsymbol{w}})$ of clustered parameters $\tilde{\boldsymbol{w}}$. We can calculate $\boldsymbol{g}(\tilde{\boldsymbol{w}})$ as follows by using transformational matrix $\boldsymbol{A}$ and a gradient vector $\boldsymbol{g}(\boldsymbol{w})$ of non-clustered parameters $\boldsymbol{w}$.

$$\boldsymbol{g}(\tilde{\boldsymbol{w}}) = \boldsymbol{A}\boldsymbol{g}(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{g}(\boldsymbol{w}) \tag{9}$$

By using equation (9), we can easily calculate $\boldsymbol{g}(\tilde{\boldsymbol{w}})$.

Below we outline the BCW (bidirectional clustering of weights) method. Since a weight clustering problem will have many local optima, the BCW is implemented as an iterative method where cluster-merge and cluster-split operations are repeated in turn until convergence. In order to obtain good clustering results, the BCW must be equipped with a reasonable criterion for each operation. To this end, we derive the second-order optimal criteria with respect to the error function.

3.2 Bottom-up Clustering

One-step bottom-up clustering is to transform $\Omega(G)$ into $\Omega(G-1)$ by a cluster-merge operation; i.e., clusters S_g and $S_{g'}$ are merged into a single cluster $\tilde{S}_g = S_g \cup S_{g'}$. Clearly, we want to select a suitable pair of clusters so as to minimize the increase of the error function.

Given a pair of clusters S_g and $S_{g'}$, we consider the second-order optimal criterion for merging $DisSim(S_g, S_{g'})$, which approximates to the increase of $E(\boldsymbol{w})$ to the 2nd order. We select a pair which minimizes $DisSim(S_g, S_{g'})$ defined below.

$$\begin{aligned} DisSim(S_g, S_{g'}) &= \frac{\left(\boldsymbol{f}_A^T \boldsymbol{C}_A \boldsymbol{d} - \hat{\boldsymbol{u}}^T \boldsymbol{d}\right)^2}{\boldsymbol{d}^T \boldsymbol{C}_A \boldsymbol{d}} \\ \boldsymbol{C}_A &= \left(\boldsymbol{A}^T \boldsymbol{H}(\tilde{\boldsymbol{w}}) \boldsymbol{A}\right)^{-1}, \quad \boldsymbol{f}_A = \boldsymbol{A}^T \boldsymbol{g}(\tilde{\boldsymbol{w}}) \\ \boldsymbol{d} &= \boldsymbol{e}_g^G - \boldsymbol{e}_{g'}^G \end{aligned} \tag{10}$$

Here, $\boldsymbol{g}(\boldsymbol{w})$ and $\boldsymbol{H}(\boldsymbol{w})$ denote the gradient and Hessian of $E(\boldsymbol{w})$ respectively, $\hat{\boldsymbol{u}}$ and $\tilde{\boldsymbol{w}}$ denote trained weight vectors. Based on the above criterion, the *one-step bottom-up clustering with retraining* selects a pair of clusters which minimizes $DisSim(S_g, S_{g'})$, and merges these two clusters. After the merge, the network with $\Omega(G-1)$ is retrained.

3.3 Top-down Clustering

One-step top-down clustering is to transform $\Omega(G)$ into $\Omega(G+1)$ by a cluster-split operation; i.e., a cluster S_g is split into two clusters S'_g and S'_{G+1} where $S_g = S'_g \cup S_{G+1}$. In this case, we want to select a suitable cluster and its partition so as to maximize the decrease of the error function.

Just after the splitting, we have a $(G+1)$-dimensional common weight vector $\tilde{\boldsymbol{v}} = (\hat{\boldsymbol{u}}^T, \hat{u}_g)^T$, and a new $D \times (G+1)$ transformational matrix $\boldsymbol{B}$ defined as below.

$$\boldsymbol{B} = \left[\sum_{d \in S_1} \boldsymbol{e}_d^D, ..., \sum_{d \in S'_g} \boldsymbol{e}_d^D, ..., \sum_{d \in S_G} \boldsymbol{e}_d^D, \sum_{d \in S_{G+1}} \boldsymbol{e}_d^D\right] \tag{11}$$

Then, we define the second-order optimal criterion for splitting *GenUtil*(S_g, S_{G+1}), which approximates to the decrease of $E(\boldsymbol{w})$ to the 2nd order. The values are positive, and the larger the better.

$$\begin{aligned} &GenUtil(S_g, S_{G+1}) = \boldsymbol{f}_B^T \boldsymbol{C}_B \boldsymbol{f}_B \\ &\boldsymbol{C}_B = \left(\boldsymbol{B}^T \boldsymbol{H}(\boldsymbol{B}\tilde{\boldsymbol{v}})\boldsymbol{B}\right)^{-1}, \quad \boldsymbol{f}_B = \boldsymbol{B}^T \boldsymbol{g}(\boldsymbol{B}\tilde{\boldsymbol{v}}) \end{aligned} \tag{12}$$

When a cluster has *m* elements, the number of different splitting amounts to $(2^m - 2)/2 = 2^{m-1} - 1$. This means an exhaustive search suffers from combinatorial explosion. In order to avoid this problem, we consider three kinds of splitting methods as shown next;

split1: Split a cluster into only one element and the others.

split2: Sort the gradients of members of a cluster in ascending order and split the cluster into two groups having smaller gradients or larger gradients.

split3: Perform the K-means to quantize the parameters obtained by the learning of non-clustered network and split a cluster depend on the result of the quantization.

Since the computational cost of performing these three methods is much smaller than that of learning of the perceptron, the BCW employs all of these three methods. Examining all the clusters, the BCW selects a suitable cluster to split and its splitting based on the criterion (12). After the splitting, the network with $\Omega(G+1)$ is retrained.

3.4 Bidirectional Clustering of Weights (BCW)

In general there may exist many local optima for a clustering problem. The single usage of either the bottom-up or top-down clustering will get stuck at a local optimum. Thus, we consider an iterative usage of both clusterings to obtain bidirectional clustering of weights (BCW).

The procedure of BCW is shown below. The BCW always converges since the number of different *A* is finite. Here *h* is the width of bidirectional clustering.

step1: Get the initial set $\Omega(D)$ through learning of a multi-layer perceptron. Perform scalar quantization for $\Omega(D)$ to get $\Omega_1(2)$. Keep the matrix $\mathbf{A}^{(0)}$ at $\Omega_1(2)$. $t \leftarrow 1$.

step2: Perform repeatedly the one-step top-down clustering with retraining from $\Omega_1(2)$ to $\Omega(2+h)$. Update the best performance for each *G* (= 2,3,…,2+h) if necessary.

step3: Perform repeatedly the one-step bottom-up clustering with retraining from $\Omega(2+h)$ to $\Omega_2(2)$. Update the best performance for each *G* if necessary. Keep $\mathbf{A}^{(t)}$ at $\Omega_2(2)$.

step4: If $\mathbf{A}^{(t)}$ is equal to one of the previous ones $A^{(t-1)}, \dots, A^{(0)}$, then stop the iteration and output the best performance of $\Omega(G)$ for each *G* as the final result. Otherwise, $t \leftarrow t+1$, $\Omega_1(G) \leftarrow \Omega_2(G)$ and go to **step 2**.

4 Applying BCW to Connectionist Polynomial Regression

The connectionist polynomial regression RF5 or RF6.4 can fit multivariate non-linear data with satisfiable precision. However, if data have many input variables, the readability of a polynomial gets worse. So it is important to get as succinct a result as possible for crisp readability. To this end, we consider applying the BCW to RF5 or RF6.4. In this case, weight sharing is applied only to weights w_{jk} in Eq. (2) or Eq. (6) for simplicity. By applying the BCW to RF5 or RF6.4, we expect not only the readability of the results but also good generalization performance avoiding the over-fitting.

Given data, we don't know in advance the optimal numbers J^* and R^* of hidden units, the optimal number I^* of rules, or the optimal number G^* of clusters for the BCW. Therefore, in order to find a model having excellent performance, we need a criterion suitable for selecting J^*, R^*, I^* and G^*. As the criterion for selecting the best model, the BCW employs the Bayesian Information Criterion (BIC) [11]. Compared to other means such as cross-validation [14] or bootstrap method [15] which need repetitive learning, BIC can select the optimal model in much less time because we need only one learning for each *J*, *R*, *I* and *G*.

The whole procedures of RF5 or RF6.4 with the BCW, including the procedure of model selection, are summarized below:

RF5+BCW:

step1: Learn the three-layer perceptron without weight sharing for each *J* = 1,2,…, and select the *J* which minimizes Eq. (13) as J^*.

$$BIC(J) = \frac{N}{2}\log\left(\frac{1}{N}\sum_{\mu=1}^{N}\left(f(\boldsymbol{q}^{\mu}, \boldsymbol{x}^{\mu}; \hat{\boldsymbol{\theta}}_J) - y^{\mu}\right)^2\right) + \frac{J(K+1)}{2}\log N \tag{13}$$

step2: Learn the perceptron under the condition of J^* with weight sharing, and select the *G* which minimizes Eq. (14) as G^*.

$$BIC(G) = \frac{N}{2}\log\left(\frac{1}{N}\sum_{\mu=1}^{N}\left(f(\boldsymbol{q}^{\mu}, \boldsymbol{x}^{\mu}; \hat{\boldsymbol{\theta}}_G) - y^{\mu}\right)^2\right) + \frac{G}{2}\log N \tag{14}$$

step3: Learn the perceptron under the condition of J^* and G^* with pruning of a near-zero common weight, and get the final result.

RF6.4+BCW:

step1: Learn the four-layer perceptron without weight sharing for each $J = 1,2,\ldots$, $R = 1,2,\ldots$, and select the J and R which minimize Eq. (15) as J^* and R^*.

$$BIC(J,R) = \frac{N}{2}\log\left(\frac{1}{N}\sum_{\mu=1}^{N}\left(f(q^\mu, x^\mu; \hat{\theta}_{J,R}) - y^\mu\right)^2\right) + \frac{JK_2 + R(J+K_1+1)}{2}\log N \quad (15)$$

step2: Learn the perceptron under the condition of J^* and R^* with weight sharing, and select the G which minimizes Eq. (14) as G^*.

step3: Learn the perceptron under the condition of J^*, R^* and G^* with pruning of a near-zero common weight.

step4: Quantize the coefficient vectors by the K-means for each $I = 1,2,\ldots$, and select the I which minimizes Eq. (16) as I^*. As for $BIC(I)$, we adopt Pelleg's X-means approach [16]. Here, $\hat{\Sigma}$ is a covariance matrix calculated by Eq. (17).

$$BIC(I) = -\sum_{i=1}^{I} N_i \log N_i + \frac{N}{2}\log|\hat{\Sigma}| + \frac{1}{2}\sum_{\mu=1}^{N}\left(c^\mu - a^{(\mu)}\right)^T \hat{\Sigma}^{-1}\left(c^\mu - a^{(\mu)}\right) + \frac{I(J^*+2)}{2}\log N \quad (16)$$

$$\hat{\Sigma} = \frac{1}{N-I}\sum_{i=1}^{I}\sum_{\mu\in G_i}\left(c^\mu - a^{(\mu)}\right)\left(c^\mu - a^{(\mu)}\right)^T \quad (17)$$

step5: Solve the classification problem by using the C4.5 program [13] and get the final result.

5 Experiments for Evaluation

5.1 Experiments for applying BCW to RF5 using an Artificial Data Set

We consider the following multivariate polynomial:

$$y = 2 - 3x_1^{1.5}x_2^{2}x_3^{1.5} + x_2^{0.5}x_3^{0.5}x_4^{1.5}x_5^{1.5} \quad (18)$$

Here we introduce ten irrelevant explanatory variables $x_6,\ldots,x_{15}$. For each sample, each x_k value is randomly generated in the range of (1,2), while the corresponding value of y is calculated by Eq. (18) with small Gaussian noise $N(0, 0.3)$ added. The size of training data is 300 (N = 300).

The initial values for weights w_{jk} are independently generated according to a uniform distribution with a range of (-1, 1); weights w_j are initially set equal to zero, but the bias w_0 is initially set to the average output over all training samples. The iteration is terminated when the gradient vector is sufficiently small, i.e., each element of the vector is less than 10^5. Weight sharing was applied only to weights w_{jk} in a three-layer perceptron. Here we set the width of bidirectional clustering as h=10.

Table 1: Frequency with which *BIC*(J) is minimized (RF5+BCW, Artificial Data 1)

models	J = 1	J = 2	J = 3	J = 4	J = 5	total
Freq.	0	**93**	7	0	0	100

Table 2: Frequency with which BIC(G) is minimized (RF5+BCW, Artificial Data 1)

models	G = 2	G = 3	G = 4	G = 5	G = 6	G = 7
Freq.	0	0	**89**	11	0	0
	G = 8	G = 9	G = 10	G = 11	G = 12	total
	0	0	0	0	0	100

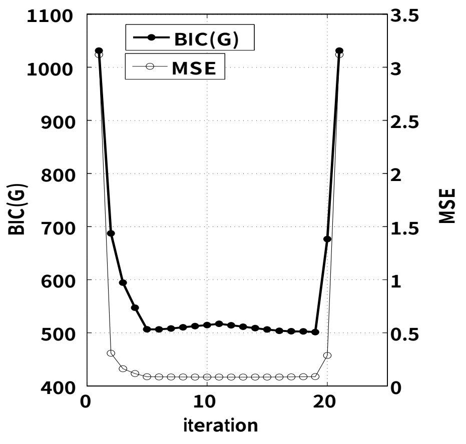

Fig. 3: Bidirectional clustering for Artificial Data 1

The number of hidden units was changed from one (J=1) to five (J=5). We performed 100 times of learning of a perceptron with different initial values of parameters. Note that J^*=2 and G^*=4 for our artificial data. Table 1 compares the frequency with which $BIC(J)$ was minimized. We can see that $BIC(J)$ was minimized at J=2 for most trials. When $BIC(J)$ was minimized at J=3, learning with J=2 converged to local optima. So we can say the optimal number J^* of hidden units is 2, which is correct.

Now that we have the optimal number J^*=2, Table 2 compares the frequency with which $BIC(G)$ was minimized under J=2. $BIC(G)$ was minimized most frequently at G=4, so we can say the optimal number G^* of clusters is 4, which is correct again. When $BIC(G)$ was minimized at G=5, the cluster having a near-zero value was split and G was incremented from the optimal value. We can obtain the same optimal model by applying the weight pruning.

Figure 3 shows how training error MSE and $BIC(G)$ changed through BCW learning in a certain trial with J=2. Training error MSE got smaller as the number G of clusters increased. Although we couldn't get a good performance of $BIC(4)$ at the beginning of the bidirectional clustering, the second $BIC(4)$ (*iteration* = 19 in the figure) was the best.

Then we pruned the near-zero common weight, and retrained the network again. The *pruning* means changing a very small common weight ($|u|$ < 0.01) to zero and performing retraining after that. Then we have the following function, almost equivalent to the original Eq. (18).

$$y = 1.9632 - 3.0116 x_1^{1.4990} x_2^{1.9951} x_3^{1.4990} + 1.0183 x_2^{0.4879} x_3^{0.4879} x_4^{1.4990} x_5^{1.4990} \quad (19)$$

Note that we have exactly the same result as the above for a model of J=2 and G=5 if the pruning is applied. This means our method has some robustness for model selection.

The whole elapsed cpu time per trial required for applying the BCW to RF5 is about 3 minutes in our experiment. We can select the best model and find almost equivalent to the original multivariate polynomial in reasonable cpu time.

5.2 Experiments for applying BCW to RF6.4 using an Artificial Data Set

We consider the following set of nominally conditioned multivariate polynomials:

$$\begin{cases} \textbf{if } q_{21} \wedge q_{31} \textbf{ then} \\ \qquad y = 1 + 2x_1^2 x_2 x_3^{0.5} + 3x_2 x_3^{0.5} x_4^2 x_5 \\ \textbf{if } q_{22} \wedge q_{31} \textbf{ then} \\ \qquad y = 2 - x_1^2 x_2 x_3^{0.5} + x_2 x_3^{0.5} x_4^2 x_5 \\ \textbf{if } (q_{12} \vee q_{13}) \wedge q_{32} \textbf{ then} \\ \qquad y = -3 + x_1^2 x_2 x_3^{0.5} - x_2 x_3^{0.5} x_4^2 x_5 \\ \textbf{if } q_{11} \wedge q_{32} \textbf{ then} \\ \qquad y = 2 + 3x_1^2 x_2 x_3^{0.5} + 4x_2 x_3^{0.5} x_4^2 x_5 \end{cases} \quad (20)$$

Here we have 15 numerical and 4 nominal explanatory variables with L_1 = 3, L_2 = L_3 = 2 and L_4 = 5. Obviously, variables q_4, x_6, …, x_{15} are irrelevant. For each sample, values of x_k and q_k are randomly taken from the interval (1,2) and from its categories respectively, while the corresponding value of y is calculated by Eq. (20) with small Gaussian noise $N(0, 0.3)$ added. The size of training data is 500 (N = 500).

The initial values for weights w_{jk}, v_{0r}, v_{jr} and v_{rkl} are independently generated according to a uniform distribution with a range of (-1, 1); The iteration is terminated when each element of the gradient vector gets smaller than 10^5. We set the width of bidirectional clustering as h=10. The numbers of hidden units was changed from one (J=1, R=1) to five (J=5, R=5), and the number of rules was changed from one (I=1) to ten (I=10). Note that J^*=2, G^*=4 and I^*=4 for our artificial data.

Table 3: Frequency with which BIC(J, R) is minimized (RF6.4+BCW, Artificial Data 2)

models	J = 1	J = 2	J = 3	J = 4	J = 5
R = 1	0	0	0	0	0
R = 2	0	0	0	0	0
R = 3	0	**86**	0	0	0
R = 4	0	14	0	0	0
R = 5	0	0	0	0	0

Table 4: Frequency with which BIC(G) is minimized (RF6.4+BCW, Artificial Data 2)

models	G = 2	G = 3	G = 4	G = 5	G = 6	G = 7
Freq.	0	0	**83**	12	5	0
	G = 8	G = 9	G = 10	G = 11	G = 12	total
	0	0	0	0	0	100

Table 5: Frequency with which $BIC(I)$ is minimized (RF6.4+BCW, Artificial Data 2)

models	I = 1	I = 2	I = 3	I = 4	I = 5	
Freq.	0	0	0	**82**	10	
	I = 6	I = 7	I = 8	I = 9	I = 10	total
	5	0	0	0	3	100

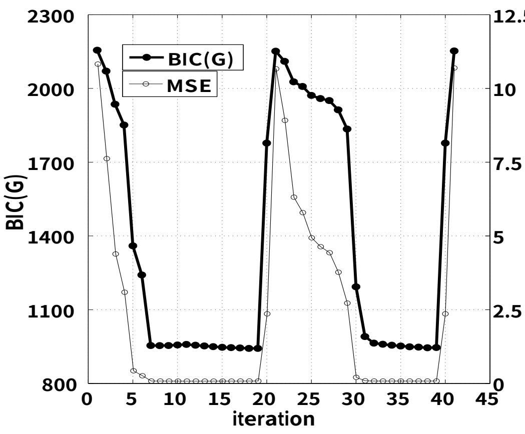

Fig. 4: Bidirectional clustering for Artificial Data 2

Table 3 compares the frequency with which $BIC(J, R)$ was minimized. $BIC(J, R)$ was minimized at J=2 for all trials, so the optimal number J^* is 2, which is correct. In this experiment, R=3 is selected most frequently. By previous experiments, we brought out the fact that original rules can be restored with satisfactory accuracy if R is set to the value greater than or equal to the *rank*($\boldsymbol{C}$), where $\boldsymbol{C}$ denotes the coefficient matrix composed of the coefficient vectors $\boldsymbol{c}_0$ and $\boldsymbol{c}_j$. So R=3 is enough to restore the original rules written in Eq. (21). However, we can restore the original rules if R is 4. Thus, it is no problem that R=4 is selected as the best model for a few trials.

Now that we have the optimal numbers J^*=2 and R^*=3, Table 4 compares the frequency with which *BIC*(G) was minimized under J=2 and R=3. We can see that *BIC*(G) was minimized at G=4, so we can select the optimal number of common weights, G^*=4.

Figure 4 shows how training error *MSE* and *BIC*(G) changed through BCW learning in a certain trial with J=2 and R=3. In this trial, bidirectional clustering was repeated two times until convergence.

We have learned the perceptron with J=2, R=3 and G=4. Then, we pruned the near-zero common weight, and retrained the network again. Then, we obtained the coefficient vectors $\boldsymbol{c}_0$ and $\boldsymbol{c}_j$, and quantized them for rule restoring. Table 5 compares the frequency with which *BIC*(I) was minimized through the vector quantization using the K-means. *BIC*(I) was minimized most frequently at I=4, so the optimal number I^* of rules is 4. However, I larger than 4 was selected as the best model in some trials. This is because we added noise to the data.

By applying the C4.5 program, we have the following rule set, almost equivalent to the original Eq. (20).

$$\begin{cases} \text{if } q_{21} \wedge q_{31} \text{ then} \\ \quad y = 1.0076 + 1.9823 x_1^{2.0109} x_2^{1.0520} x_3^{0.4981} \\ \qquad +2.9832 x_2^{1.0520} x_3^{0.4981} x_4^{2.0109} x_5^{1.0520} \\ \text{if } q_{22} \wedge q_{31} \text{ then} \\ \quad y = 2.0123 - 1.0001 x_1^{2.0109} x_2^{1.0520} x_3^{0.4981} \\ \qquad +0.9699 x_2^{1.0520} x_3^{0.4981} x_4^{2.0109} x_5^{1.0520} \\ \text{if } (q_{12} \vee q_{13}) \wedge q_{32} \text{ then} \\ \quad y = -2.9800 + 1.1275 x_1^{2.0109} x_2^{1.0520} x_3^{0.4981} \\ \qquad -0.9910 x_2^{1.0520} x_3^{0.4981} x_4^{2.0109} x_5^{1.0520} \\ \text{if } q_{11} \wedge q_{32} \text{ then} \\ \quad y = 1.9752 + 3.0646 x_1^{2.0109} x_2^{1.0520} x_3^{0.4981} \\ \qquad +4.0423 x_2^{1.0520} x_3^{0.4981} x_4^{2.0109} x_5^{1.0520} \end{cases} \tag{21}$$

The whole elapsed cpu time required for applying the BCW to RF6.4 is about 11 minutes in our experiment.

5.3 Experiments using Real Data Sets

We evaluated the performance of applying the BCW to RF5 or RF6.4 by using 10 real data sets[1]. Table 6 describes these real data sets. Here, K_2 and K_{all} denote the total numbers of numerical explanatory variables and dummy variables respectively. The initial values of parameters and termination condition are set in the same way as before. All numerical variables are normalized as follows.

$$\tilde{x}_j = \frac{x_j}{\max(x_j)}, \quad \tilde{y} = \frac{y - \text{mean}(y)}{\text{std}(y)} \tag{22}$$

First, we performed 100 times of learning of the perceptron of RF5 or RF6.4 without weight sharing. We varied J or R from 1 to 5. Table 7 shows the best J and R which minimized *BIC*(J) of RF5 or *BIC*(J,R) of RF6.4 most frequently.

Next, we performed 100 times of learning of the

[1]The cpu and boston data sets were taken from the UCI Repository of Machine Learning Databases. The college, cholesterol, b-carotene, mpg, lung-cancer and wage data sets were taken from the StatLib. The yokohama data set was taken from the official web page of Kanagawa prefecture, Japan. The baseball data set was taken from the directory of Japanese professional baseball player in 2006.

Table 6: Real data sets used in our experiment

data set	contents of criterion variable	N	K_2	K_{all}
cpu	performance of CPU	205	6	26
boston	housing price in Boston	486	12	76
college	instructional expenditure of colleges	1129	11	51
cholesterol	amount of cholesterol	297	5	13
b-carotene	amount of beta-carotene	315	10	6
mpg	fuel cost of cars	388	5	28
lung-cancer	survival time of lung cancer patients	126	3	6
wage	wage of workers per hour	534	2	14
yokohama	housing price in Yokohama	558	4	23
baseball	annual salary of baseball players	219	10	14

Table 7: J^* of RF5 and J^*, R^* of RF6.4 (Real Data)

	RF5	**RF6.4**	
	J*	***J****	***R****
cpu	3	3	3
boston	1	2	1
college	3	2	2
cholesterol	2	2	3
b-carotene	1	1	4
mpg	2	2	3
lung-cancer	4	3	5
wage	5	4	3
yokohama	3	5	2
baseball	3	3	4

Table 8: Frequency with which *BIC*(G) is minimized (RF5+BCW, Real Data)

G	2	3	4	5	6	7	8	9	10	11	12	total
cpu	0	0	0	**83**	15	2	0	0	0	0	0	100
boston	0	0	**65**	21	13	0	1	0	0	0	0	100
college	0	0	0	0	0	0	1	**57**	21	13	8	100
cholesterol	0	0	0	12	**52**	11	8	7	10	0	0	100
b-carotene	0	0	0	0	**61**	21	8	3	7	0	0	100
mpg	0	0	**52**	10	22	10	0	0	6	0	0	100
lung-cancer	0	4	**31**	9	13	11	7	6	6	3	10	100
wage	0	0	0	**37**	14	14	8	2	25	0	0	100
yokohama	0	0	0	0	0	4	8	**47**	21	2	18	100
baseball	0	0	0	**70**	3	2	6	5	4	5	5	100

bidirectional clustering with different initial values of parameters. The numbers *J* and *R* are set to the values showed in Table 7. The optimal number J^* of hidden units in RF5 was the same as J^* in RF6.4 for five data sets out of ten. Overall, two J^* are very similar. Tables 8 and 9 compare the frequency with which *BIC*(*G*) was minimized in RF5 and RF6.4 respectively. The optimal number G^* of common weights in RF5 was the same as G^* in RF6.4 for only three data sets out of ten. In most data sets, two G^* are similar, but there are a few cases where two G^* are rather different.

Table 9: Frequency with which BIC(G) is minimized (RF6.4+BCW, Real Data)

G	2	3	4	5	6	7	8	9	10	11	12	total
cpu	0	0	0	0	**83**	10	4	0	0	0	3	100
boston	0	0	0	0	0	0	0	6	20	21	**53**	100
college	0	0	0	0	0	0	0	17	**63**	15	5	100
cholesterol	0	0	0	0	**41**	16	5	17	21	0	0	100
b-carotene	0	0	0	0	0	7	**67**	21	5	0	0	100
mpg	0	0	0	6	**72**	7	7	5	3	0	0	100
lung-cancer	0	4	**52**	4	8	14	8	10	0	0	0	100
wage	0	0	0	**62**	21	2	15	0	0	0	0	100
yokohama	0	0	0	0	**50**	3	17	12	12	0	6	100
baseball	0	0	0	0	0	6	**61**	31	0	2	0	100

Table 10: Frequency with which *BIC*(*I*) is minimized (RF6.4+BCW, Real Data)

I	1	2	3	4	5	6	7	8	9	10	total
cpu	0	0	0	0	26	**62**	4	0	0	8	100
boston	0	0	0	0	0	0	0	0	29	**71**	100
college	0	0	0	0	0	0	2	10	**68**	20	100
cholesterol	0	0	0	0	0	0	8	**81**	4	7	100
b-carotene	0	0	0	0	4	**59**	18	4	1	14	100
mpg	0	0	11	15	**46**	21	7	0	0	0	100
lung-cancer	0	0	25	**62**	13	0	0	0	0	0	100
wage	0	0	0	0	0	31	**51**	10	2	6	100
yokohama	0	0	0	20	**52**	9	7	2	1	9	100
baseball	0	0	0	0	0	**71**	29	0	0	0	100

Table 11: Final polynomials obtained by RF5+BCW (Real Data)

data set	polynomial
cpu	$y = -0.5787 + 0.0984(x_1x_2x_3x_5)^{0.6465} + 0.0847(x_1x_2x_3x_5)^{0.6465} + 0.0165x_1^{-1.9946}(x_4x_6)^{1.0719}$
boston	$y = -1.2856 + 1.0688(x_4x_7x_{12})^{-0.2939}(x_5x_8x_{11})^{0.1687}(x_9x_{10})^{-0.1267}$
college	$y = -0.8470 - 0.1943x_1^{-0.4855}(x_2x_6x_7x_9)^{0.1133}x_3^{0.6548}x_4^{-0.5564}x_5^{0.3743}x_8^{-0.0933}(x_{11}x_{12})^{0.3743}$ $+0.3098(x_1x_4)^{-0.3280}(x_2x_3x_6x_8)^{0.1133}x_5^{-0.0933}x_7^{0.6584}(x_{11}x_{12})^{0.0675} + 0.4062x_1^{0.6584}x_2 - 0.4855(x_3x_5x_6)^{0.1133}(x_{10}x_{11})^{0.3743}x_{12}^{-0.3280}$
cholesterol	$y = -0.8266 + 0.3531x_1^{-0.2855}(x_2x_5)^{0.2608}x_3^{-0.2075} + 0.3286x_1^{0.8757}(x_2x_5)^{-0.2075}x_3^{0.6005}$
b-carotene	$y = 0.7130 - 0.6667x_1^{-0.0601}x_2^{0.1955}(x_4x_7)^{0.0816}x_5^{-0.1570}(x_8x_{10})^{-0.1206}$
mpg	$y = -0.8155 + 1.2614(x_1x_2x_5)^{-0.1024}x_3^{-0.4020} - 0.5437x_1^{-0.1024}x_4^{0.4709}$
lung-cancer	$y = -0.5464 - 0.0957x_1^{1.6679}x_2^{0.2851}x_3^{1.2930} - 0.0743(x_1x_3)^{0.2077}x_2^{0.2851} + 0.0466(x_1x_2)^{0.2851}x_3^{0.2077} + 0.5243x_1^{1.2930}(x_2x_3)^{0.2077}$
wage	$y = -1.9990 + 1.0948x_2^{0.1345} - 0.2850x_1^{1.5548}x_2^{0.1345} + 0.2477x_1^{0.9810} + 0.5706x_1^{0.9810}x_2^{0.3538} + 0.2485x_1^{0.9810}$
yokohama	$y = -2.7450 + 1.4982x_1^{-0.1970}x_2^{-0.1507}x_3^{0.1734}x_4^{-0.2963} + 1.3147x_1^{0.2369}x_2^{-0.1970}x_3^{-0.2963}x_4^{0.3541} - 0.0999x_1^{-0.8529}x_2^{0.1734}x_4^{-1.6902}$
baseball	$y = -0.5524 - 0.1318(x_1x_5x_6x_7x_{10})^{0.2851}x_2^{-0.8823}(x_3x_8x_9)^{0.5085} + 0.0880x_2^{-0.8823}(x_3x_4)^{0.5085}(x_5x_7x_9x_{10})^{0.2851}x_8^{1.0288} + 0.1845x_3^{1.0288}x_7^{0.2051}$

Table 12: Final rules obtained by RF6.4+BCW (CPU Data)

rule	nominal variable q_1	Polynomial
rule1	26	$y = -0.5114 + 0.1182(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} - 0.0205(x_1x_5)^{-0.2545}x_4^{1.9526} + 0.1002(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$
rule2	7	$y = -0.8296 + 0.0056(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} + 0.0684(x_1x_5)^{-0.2545}x_4^{1.9526} + 0.1002(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$
rule3	2, 3, 12, 13, 14, 17, 19, 21, 22, 23	$y = -0.4385 + 0.3313(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} - 0.0370(x_1x_5)^{-0.2545}x_4^{1.9526} + 0.0953(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$
rule4	1, 4, 5, 11, 15, 16, 18, 20, 24, 25	$y = -0.5323 + 0.0090(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} + 0.0044(x_1x_5)^{-0.2545}x_4^{1.9526} + 0.1625(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$
rule5	8,9,10	$y = -0.5743 + 0.0047(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} + 0.0403(x_1x_5)^{-0.2545}x_4^{1.9526} + 0.0843(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$
rule6	6	$y = -0.5545 + 0.6808(x_1x_2x_6)^{0.4619}x_3^{0.7466}x_4^{1.9526}x_5^{-0.6531} + 0.0086(x_1x_5)^{-0.2545}x_4^{1.9526} - 0.5963(x_1x_4)^{-0.2545}(x_2x_5)^{0.4619}x_3^{8.7466}$

In RF6.4, we quantized the coefficient vectors $\boldsymbol{c}_0$, $\boldsymbol{c}_j$ for rule restoring. We performed 100 times learning of the perceptron with weight sharing under the condition of J^*, R^* and G^* showed in Tables 7 and 9. We varied the number *I* of rules from 1 to 10. Table 10 compares the frequency with which *BIC*(*I*) was minimized.

Table 11 shows the final polynomials obtained by RF5+BCW after the weight pruning. We found very simple polynomials across the board.

Tables 12 and 13 show the final sets of rules obtained

Table 13: Final rules obtained by RF6.4+BCW (Lung-cancer Data)

rule	nominal variables			Polynomial
	q_1	q_2	q_3	
rule1	2	1	2	$y = -3.5677 + 4.0499 x_1^{0.8096} - 0.0668 (x_1 x_2)^{0.8096} x_3^{-2.3223} + 1.9200 (x_1 x_2 x_3)^{-1.1109}$
rule2	1	1	2	$y = -1.0920 + 1.6133 x_1^{0.8096} - 0.1108 (x_1 x_2)^{0.8096} x_3^{-2.3223} + 0.0320 (x_1 x_2 x_3)^{-1.1109}$
	2	4	2	
rule3	1	3,4	2	$y = -0.6002 + 0.3157 x_1^{0.8096} - 0.0021 (x_1 x_2)^{0.8096} x_3^{-2.3223} + 0.1274 (x_1 x_2 x_3)^{-1.1109}$
	2	1,2,3,4	1	
	2	2,3	2	
rule4	1	1, 2, 3, 4	1	$y = -0.1057 + 0.0087 x_1^{0.8096} + 0.0062 (x_1 x_2)^{0.8096} x_3^{-2.3223} + 0.0577 (x_1 x_2 x_3)^{-1.1109}$
	1	2	2	

Table 14: Comparison of 10-fold cross validation errors (Real Data having only numerical variables)

data set	MR	HMEx	NNx	NNx+BCW	RF5	RF5+BCW
cpu	4.609E+03	**1.569E+03**	1.625E+03	1.595E+03	2.206E+03	1.627E+03
boston	2.355E+01	1.060E+01	9.740E+00	1.054E+01	9.500E+00	**8.206E+00**
college	9.570E+06	9.495E+06	9.230E+06	**9.090E+06**	9.659E+06	9.342E+06
cholesterol	2.641E+03	2.476E+03	2.579E+03	2.606E+03	2.511E+03	**2.369E+03**
b-carotene	2.989E+04	2.887E+04	2.923E+04	**2.850E+04**	2.886E+04	2.923E+04
mpg	1.809E+01	1.588E+01	**1.541E+01**	1.676E+01	1.711E+01	1.696E+01
lung-cancer	2.284E+04	2.183E+04	2.090E+04	1.988E+04	2.186E+04	**1.954E+04**
wage	2.124E+01	1.924E+01	1.925E+01	**1.897E+01**	2.029E+01	1.976E+01
yokohama	1.232E+09	4.014E+08	4.250E+08	4.410E+08	4.493E+08	**3.887E+08**
baseball-2006	1.755E+07	1.602E+07	1.656E+07	1.626E+07	1.668E+07	**1.521E+07**

Table 15: Comparison of 10-fold cross validation errors (Real Data having both numerical and nominal variables)

data set	QT	HMEq	NNqx	NNqx+BCW	RF6.4	RF6.4+BCW
cpu	4.301E+03	1.453E+03	1.566E+03	1.498E+03	2.021E+03	**1.363E+03**
boston	1.350E+01	1.209E+01	9.450E+00	1.025E+01	9.262E+00	**7.533E+00**
college	9.703E+06	8.661E+06	6.910E+06	7.331E+06	**6.330E+06**	8.283E+06
cholesterol	2.584E+03	2.502E+03	2.577E+03	2.517E+03	2.446E+03	**2.418E+03**
b-carotene	2.928E+04	2.895E+04	2.893E+04	2.786E+04	2.958E+04	**2.646E+04**
mpg	1.622E+01	1.392E+01	1.373E+01	1.569E+01	**1.283E+01**	1.495E+01
lung-cancer	2.136E+04	2.154E+04	2.156E+04	**1.801E+04**	2.172E+04	1.857E+04
wage	1.884E+01	1.925E+01	1.864E+01	**1.818E+01**	1.849E+01	1.830E+01
yokohama	4.151E+08	3.790E+08	**3.462E+08**	4.064E+08	3.911E+08	4.667E+08
baseball-2006	1.849E+07	**1.406E+07**	1.670E+07	1.680E+07	1.732E+07	1.620E+07

by RF6.4+BCW for two data sets. Since there is not enough space for showing all the rule sets, we show only the results for cpu and lung-cancer data sets. The tables show we found very simple rule sets.

We compared the generalization performance of several regression methods by 10-fold cross validation. We considered four kinds of regression methods (linear regression, HME, three-layer perceptron, and polynomial regression). As linear regression models, we use multiple regression (MR) for the data having only numerical variables, and quantification theory type I (QT) for the data having both numerical and nominal variables. The simplest regression form of the HME (Hierarchical Mixtures of Experts) [17][18] is piecewise linear regression, which constructs subspaces by using gating variables and applies linear regression in each subspace. We use the HME in two ways: using numerical variables as gating variables (HMEx), or using nominal variables as gating (HMEq). We varied the number I of experts from 1 to 8. For a three-layer perceptron, we learn the perceptron in four ways: using only numerical variables (NNx), using both numerical and nominal variables (NNqx), or applying the BCW to NNx or NNqx (NNx+BCW, NNqx+BCW). We varied the number J of hidden units of the perceptron from 1 to 5, and the number G of clusters from 2 to 12. As connectionist polynomial regression models, we compare four models (RF5, RF6.4, RF5+BCW, RF6.4+BCW).

Tables 14 and 15 compare the generalization performances by 10-fold cross validation errors. Table 14 shows the results using only numerical variables, and

Table 15 shows the results using both numerical and nominal variables. We can see that the performance of RF5+BCW or RF6.4+BCW was better than plain RF5 or RF6.4 in most data sets, and showed the best performance in many data sets. So we can say that applying the BCW to connectionist polynomial regressions is valuable for finding succinct polynomials and fitting better the data. Although NN or NN+BCW showed the best performance in some data sets, they are rather poor in terms of the readability.

6 Conclusion

This paper proposed a weight sharing method called BCW and applied it to connectionist polynomial regression methods called RF5 and RF6.4 for finding succinct multivariate polynomials. In our experiments using artificial data, our method selected the original model by BIC and found polynomials almost equivalent to the original. In our experiments using 10 real data sets, our method found polynomials having crisp readability with satisfactory generalization performance. In the future we plan to apply the BCW to other models and evaluate its usefulness.

References

[1] P. Langley, "Bacon.1: a general discovery system," Proc. 2nd National Conf. of the Canadian Society for Computational Studies of Intelligence, pp.173-180, 1978.

[2] K. Saito and R. Nakano, "Law discovery using neural networks," Proc. 15th Int. Joint Conf. on Artificial Intelligence, pp.1078–1083, 1997.

[3] Y. Tanahashi, K. Saito and R. Nakano, "Piecewise multivariate polynomials using a four-layer perceptron," Proc. 8th Int. Conf. on Knowledge-based Intelligent Information & Engineering Systems, pp.602-608, 2004.

[4] C. M. Bishop, "Neural networks for pattern recognition," Clarendon Press, 1995.

[5] S. Haykin, "Neural networks - a comprehensive foundation, 2nd edition," Prentice-Hall, 1999.

[6] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight sharing," Neural Computation, vol.4, no.4, pp.473-493, 1992.

[7] C. M. Bishop, "Pattern recognition and machine learning," Springer, 2006.

[8] F. Koksal, E. Alpaydin and G. Dundar, "Weight quantization for multi-layer perceptrons using soft weight sharing," Proc. of Int. Conf. on Artificial Neural Networks, pp.211-216, 2001.

[9] Y. LeCun, J. S. Denker and S. A. Solla, "Optimal brain damage," Advances in Neural Information Processing Systems, vol.2, ppl598-605, 1990.

[10] K. J. Binkley, K. Seehart and M. Hagiwara, "A study of artificial neural network architectures for othello evaluation functions," Trans. of the Japanese Society for Artificial Intelligence 22(5), pp.461-471, 2007.

[11] G. Schwarz, "Estimating the dimension of a model," Annals of Statistics 6, pp.461-464, 1987.

[12] K. Saito, R. Nakano, "Partial BFGS update and efficient step-length calculation for three-layer neural networks," Neural Computation 9(1), pp.239-257, 1997.

[13] J. R. Quinlan, "C4.5: programs for machine learning," Morgan Kaufmann, 1993.

[14] M. Stone, "Cross-validatory choice and assessment of statistical predictions," Journal of the Royal Statistical Society B, vol.64, pp.111–147, 1974.

[15] B. Efron, "Bootstrap methods: Another look at the jackknife," Ann. Statist, vol.7, pp.1-26, 1979.

[16] D. Pelleg and A. Moore, "X-means: Extracting K-means with efficient estimation of the number of clusters," Proc. 17th Int. Conf. on Machine Learning, pp.727-734, 2000.

[17] R.A. Jacobs, M.I. Jordan, S.J. Nowlan and G.E. Hinton, "Adaptive mixtures of local experts," Neural Computation, vol.3, no.1, pp.79-87, 1991.

[18] M.I. Jordan and R.A. Jacobs, "Hierarchical mixtures of experts and EM algorithm," Neural Computation, vol.6, no.2, pp.181-214, 1994.