Evolving Hypernetworks for Pattern Classification

Joo-Kyung Kim and Byoung-Tak Zhang

Abstract— **Hypernetworks consist of a large number of hyperedges that represent higher-order features sampled from training patterns. Evolutionary algorithms have been used as a method for evolving hypernetworks. The order of a hyperedge is defined as the number of feature variables in the hyperedge and it is an important parameter of the hypernetwork model. Previous studies used fixed-order hyperedges which limit model spaces and, thus, the best performance achievable by hypernetworks. Here, we present a method for evolving variable-order hypernetwork models. To find the proper orders automatically, the fitness values are calculated for each hyperedge and the hyperedges with low fitness values are substituted by new hyperedges. The method was tested on three data sets from UCI machine learning repository. The results show that the evolutionary hypernetworks show classification accuracies comparable to those of other conventional algorithms, find appropriate orders of hyperedges automatically, and extract important rules in the hyperedges for the given pattern classification problems.**

I. Introduction

Capturing higher-order correlation can be very important for complex pattern classification problems [1]. Hypernetworks have been presented as a probabilistic model of learning higher-order correlations using hypergraph structure [2], [3]. The hypernetwork model has a random graph structure consisting of a large number of hyperedges. The hyperedges represent feature combinations and are sampled from the training examples. For pattern classification problems, the model makes decisions by voting, i.e., counts the labels on the hyperedges matched to a given input pattern and reporting the winning class label. Each hyperedge which is a feature subset of training patterns reflects a local property of training patterns. Because each hyperedge can contain many feature variables, it represents a higher-order correlation structure of the data.

Hypernetworks can be represented as hypergraphs [4]. Possible feature variables of hyperedges correspond to vertices, and each hyperedge in hypernetworks corresponds to a hyperedge in hypergraphs. The weight of a hyperedge in hypergraphs corresponds to the number of identical hyperedges in hypernetworks. Hyperedges can be converted to association rules [5] that may explain highly related features observed from items in data mining. There are also studies representing association rules as hypergraphs [6].

Previous studies have shown that hypernetworks can be evolved to solve various machine learning problems. Zhang and Jang [7] used a hypernetwork model for learning to diagnose microarray data for ALL/AML leukemia. It is also suggested that the hypernetwork model can be realized in DNA molecules and be evolved by molecular operations[3]. Kim and Zhang [8] performed text classification using hypernetworks. Zhang and Kim [2] showed that hypernetworks can perform pattern completion tasks and thus may be used as associative memories. The classification processes of the hypernetwork models are very simple. It is based on a majority voting of hyperedges that matched to the given input pattern. However, constructing hypernetworks with improved performance are limited since they used only hyperedges generated during the initialization processes and the orders of hyperedges are fixed. We can miss hyperedges which are critical for classification at the initialization processes, and the best orders of hyperedges can be different for different problems. Therefore, these drawbacks can result in low performance.

In this paper, we propose a method to evolve the hypernetworks that searches effectively large problem spaces by resampling of hyperedges and finds appropriate orders of hyperedges during the evolutionary learning process. We also show simulation results on three data sets, i.e., Optical Recognition of Handwritten Digits data, SPECT heart data, and 1984 United States Congressional Voting Records Database data from UCI machine learning repository.

The rest of this paper is organized as follows. Section 2 describes the hypernetwork model. Section 3 presents the evolutionary algorithm for learning hypernetworks of variable-order hyperedges for pattern classification. Section 4 shows the simulation results and their analysis compared to other state-of-the-art machine learning algorithms. Section 5 draws conclusions.

II. The Hypernetwork Model

A hypernetwork is described as a classifier. Let $X = \{x_1, x_2, ..., x_n\}$ be the training set. For k dimensionality of an input space, each training pattern $x_i = \{x_{i_1}, x_{i_2}, ..., x_{i_k}, y_i\}$ consists of k components where $x_{i_j} \in \{0, 1\}$ and the class label $y \in Y$. Then a classifier f outputs the label of the given input z where $z = \{z_1, z_2, ..., z_k\}$. Therefore we can regard a classifier as a function $y = f(z)$.

A hypernetwork can be described similar to a hypergraph. A hypergraph is an undirected graph $G = (V, E)$ whose edges connect a non-null number of vertices, where $V = \{v_1, v_2, ..., v_n\}$ is a set of vertices, $E = \{e_1, e_2, ..., e_m\}$ is a set of hyperedges, and $e_i = \{v_{i_1}, v_{i_2}, ..., v_{i_j}\}$ is a hyperedge that contains one or more vertices and has a weight value.

Figure 1 is a sample of hypergraph[1], where $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and $E = \{e_1 = \{v_1, v_2, v_3\}, e_2 = \{v_2, v_3\}, e_3 = \{v_3, v_5, v_6\}, e_4 = \{v_4\}\}$.

The authors are with the School of Computer Science and Engineering, Seoul National University, Seoul 151-744, Korea (email: jkkim@bi.snu.ac.kr; btzhang@bi.snu.ac.kr).

[1] http://upload.wikimedia.org/wikipedia/commons/0/09/Hypergraph.gif

1-4244-1340-0/07/$25.00 ©2007 IEEE

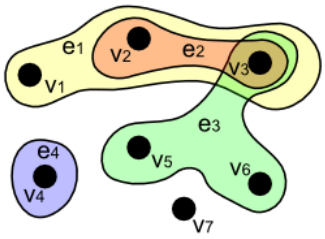

Fig. 1. A sample of hypergraph

A hypernetwork is $H = (X, L)$, where X is a set of feature variables, $L = \{l_1, l_2, ..., l_m\}$ is a library or population that is a multiset of hyperedges, and $l_i = \{l_{i_{f_1}}, l_{i_{f_2}}, ..., l_{i_{f_c}}, y_i\}$ is a hyperedge that contains c feature variables and a class label. f values of hyperedges are feature indices of a problem space. Therefore $1 \leq f \leq k$, where k is the dimensionality of the problem space. While each hyperedge in hypergraphs is unique and has a weight value, the weight of a hyperedge l_i in hypernetworks is calculated as the number of identical elements l_i in L. The number of feature variables that a hyperedge contains is called an order.

Once a hypernetwork is properly evolved, we can use it as a classifier. For a input pattern, a hypernetwork H outputs the estimated class label of the input. i.e., $y^* = H(x)$, where x is an input pattern, H is a hypernetwork, and y^* is the estimated class label. The classification procedure is summarized as Figure 2.

- 1. Presents an input pattern z.
- 2. Extract all hyperedges that are matched to the input pattern x.
- 3. Count the number of each class in the extracted hyperedges.
- 4. Classify the input pattern x as the class that counted most frequently.

Fig. 2. Classification procedure

We say that a hyperedge is matched to a pattern when values of all feature variables in the hyperedge are equal to corresponding components of the pattern. For example, let a pattern whose dimensionality is 5 be $x_i = \{x_{i_1} = 1, x_{i_2} = 0, x_{i_3} = 0, x_{i_4} = 1, x_{i_5} = 1, x_{i_y} = 3\}$ and a hyperedge be $l_j = \{l_{j_1} = 1, l_{j_3} = 0, l_{j_4} = 1, l_{j_y} = 3\}$. In this case, there are 3 feature variables in the hyperedges that mean the order of the hyperedge is 3. Because $l_{j_1} = x_{i_1}$, $l_{j_3} = x_{i_3}$, and $l_{j_4} = x_{j_4}$, all feature variables in the library element are equal to the corresponding components of the pattern. Therefore we say that a hyperedge l_j and a pattern x_i are matched. Furthermore, if a hyperedge l_j is matched to a pattern x_i, we say that it is a correct match if $l_{j_y} = x_{i_y}$, and say that it is a wrong match if $l_{j_y} \neq x_{i_y}$.

III. Evolving the Hypernetworks

We propose a new evolving method that can search larger problem spaces compared to previous methods evolving hypernetworks, and can find appropriate orders of hyperedges automatically. The procedure is as in Figure 3. In the procedure, k is the dimensionality of input space, X is the training set, and w is a given weight to control the number of substituted hyperedges.

In Step 1, related variables are initialized. $l_i = RFS(x_i)$ is a random feature subset of x_i including x_{i_y} and the probability of the order $P(|l_i| = f) = orderProb[f - 1]$. fit_w is the fitness value for misclassified training patterns and fit_c is for correctly classified training patterns. In step 2, training patterns are classified by the current hypernetwork to differentiate misclassified patterns and correctly classified patterns. In step 3, fitness values are calculated for each hyperedge. In step 4, hyperedges are sorted to select library elements that will be remained. In step 5, the number of substituting hyperedges is calculated. In step 6, the new order probabilities are calculated. In Step 7 and 8, hyperedges not suitable for pattern classification are substituted.

A. Searching larger problem spaces

The number of possible hyperedges in a hypernetwork is $2^k \times C(n, k)$, where n is the dimensionality of the input space of the problem and k is the selected order. As the dimensionality of the input space increases, a hypernetwork cannot contain large number of possible hyperedges initially because the number of hyperedges that a hypernetwork can contain at a moment is limited. Evolving methods in previous studies only change the number of duplications of initially generated hyperedges. If the hypernetwork do not contain critical hyperedges initially, the performance improvement can be limited. In our method, we substitute hyperedges whose fitness is not so good to newly generated hyperedges. Therefore we can use more library elements, and the problem spaces we can search become larger. Indeed, because hyperedges that never match to any training patterns are needless, using hyperedges only generated from training patterns is enough to search the problem spaces.

From step 3 to step 5, we select hyperedges that will be substituted. In step 3, we calculate the classification ratio for training patterns. When we classify a training pattern, we do not use hyperedges sampled from the pattern because using them can result in survival of hyperedges with very high orders. Hyperedges with very high orders may be well matched only to source patterns. In that cases, even though the accuracies for training patterns can be very high, the accuracies for test patterns can still be low. This means that there may be some overfittings. In step 4, we decide the precedence of hyperedges by assigning fitness values. We first concentrate on the fitness that is calculated on misclassified training patterns because our aim is to substitute hyperedges that make the training pattern misclassified. The preference for hyperedges whose fitness values of correctly training patterns is next. The preference for hyperedges whose orders are low is the last. If there are two hyperedges and all the fitness values are equivalent with each other, we prefer the one whose order is low. In step 5, we decide the number of hyperedges that will be substituted. The number

- 1. Initialization.
  $L := \phi, maxSubstCnt := 0$
  for $i := 1$ to k do
    $orderProb[i] := 1/k$
  for each $x_i \in X$ do
    for $i := 1$ to 100 do
      $L := L \cup l_i$, where $l_i = RFS(x_i)$, and set $fit_w(l_i) := 0$, $fit_c(l_i) := 0$.
- 2. Classify training patterns according to Figure 2, except step 2. For step 2, we do not extract hyperedges sampled from the pattern x_i. Let the set of correctly classified patterns be X^c, the set of misclassified patterns be X^w, and misclassified ratio $r = \frac{|X^w|}{|X|}$.
- 3. Calculate fitness values.
  for each $L_i \in L$ do
    for each $x_i \in X^w$ do
      if l_i is matched to x_i then
        if $l_{i_y} = x_{i_y}$ then
          $fit_w(l_i) := fit_w(l_i) + |Y| - 1$
        else $fit_w(l_i) := fit_w(l_i) - 1$
    for each $x_i \in X^c$ do
      if l_i is matched to x_i then
        if $l_{i_y} = x_{i_y}$ then
          $fit_c(l_i) := fit_c(l_i) + |Y| - 1$
        else $fit_c(l_i) := fit_c(l_i) - 1$
- 4. Sort hyperedges as following criterion
  sort in descending order according to fit_w.
  if $fit_w(l_i) = fit_w(l_j)$, then
    sort in descending order according to fit_c.
    if $fit_c(l_i) = fit_c(l_j)$, then
      sort in ascending order according to orders.
- 5. Decide the number of substitution.
  $substCnt := w \times r \times |L|$
  if $substCnt > maxSubstCnt$, then
    $substCnt := maxSubstCnt$
    $maxSubstCnt := 0.8 \times substCnt$
- 6. Update order probabilities.
  For $|L| - substCnt$ hyperedges remained, the number of hyperedges for each order i is $libCnt[i]$.
  Then, $orderProb[i] := libCnt[i]/(|L| - substCnt)$.
- 7. Substitute $substCnt$ hyperedges from the bottom of sorted hyperedges. Each substituting hyperedge is $RFS(x_i)$, where $x_i \in X$ and x_{i_y} is the class label of the substituted hyperedge.
- 8. Substitute all hyperedges that are both never matched with misclassified patterns and matched with correctly classified patterns once or less. The substituting hyperedges are generated as step 7.
- 9. Go to step 4 unless $maxSubstCnt = 0$.

Fig. 3. Evolving procedure

is proportional to the classification ratio r calculated in step 2, the number of total hyperedges, and a given weight w. Therefore, if the classification ratio in step 2 was not good, we substitute more hyperedges, and if the ratio was good, we substitute fewer hyperedges. To reduce the maximum possible substituting number gradually, the number of substitution do not exceed nine tenth of the number of former epoch.

When the hyperedges are matched, we add $|Y| - 1$ to the fitness values, where Y is the number of classes. This is because as the number of classes in the problem increases, a tendency that is decreasing percentage of matched hyperedges whose class labels are correct emerges. As the number of classes increase, hyperedges that are rarely matched can get higher fitness values. In these cases, we need to give more incentive to hyperedges that match and correct for some patterns even though they are wrong for more patterns. Therefore we give more fitness values for correctly matched hyperedges when there are many class labels.

We generated equal number of hyperedges per pattern. However, in step 7 and 8, we use random training patterns with the same class labels as those of eliminated hyperedges. Keeping the distribution of class labels prevent hypernetwork classifying well just for patterns with specific class labels. On the other hand, we do not have to keep the number of hyperedges from one training pattern equal. Therefore, hyperedges from noisy training patterns can be weeded out because their fitness values may be bad in many cases. This can be thought to be a noise reduction process.

B. Finding appropriate orders

In the hypernetwork models, finding appropriate orders of hyperedges is very important for the performance of pattern classification. In general, there is a tendency that the lower the orders of hyperedges, the more training patterns are matched. That is because hyperedges with lower orders are matched even if just few components of the elements are equal to that of training patterns. For example, if $l_1 = \{l_{1_1} = 1, l_{1_3} = 0\}$ whose order is 2 and $l_2 = \{l_{2_2} = 0, l_{2_3} = 1, l_{2_5} = 0, l_{2_6} = 1\}$ whose order is 4 exist, l_1 is matched to a training pattern when just 2 components l_{1_1}, l_{1_3} are equal to that of a pattern while l_2 is matched when 4 components $l_{2_2}, l_{2_3}, l_{2_5}, l_{2_6}$ are equal. However, there is also a tendency that the lower the orders of hyperedges, the more training patterns whose labels are different from them are matched. Generally, patterns with the same class label are neighborhoods in terms of the distance between patterns. Patterns with the same class labels will have many common values of input variables whereas patterns that have different class labels will have a few common values of input variables. Therefore we should find appropriate orders of hyperedges that do not match to patterns whose class labels are different though they match to training patterns.

Evolving methods on previous studies simply increase the number of hyperedges that are matched to training patterns and optionally decrease them if not matched. If we use mixed orders of hyperedges with these methods, then hyperedges whose orders are relatively low are duplicated frequently

regardless of the labels of hyperedges are equal to that of training patterns as explained already.

To prevent this side effect, we introduce fitness values for each hyperedge. When a hyperedge is matched to a pattern, the fitness values are increased if the label of the element is equal to that of the pattern and the fitness values are decreased if not equal. The substitution of hyperedges is due to the fitness values. Now, hyperedges with lower orders can be matched to training patterns more frequently than hyperedges with higher orders, and we can penalize the matches that we do not want. However, with this method, patterns whose orders are very high and rarely matched to training patterns can have either 0 or 1 fitness. Because these patterns seldom participate to classification process, we do not need them, and substitute them also.

To find appropriate orders of hyperedges, we first initialize sampling order distribution uniformly in step 1. Then, we change the distribution due to hyperedges remained whose fitness is relatively better in step 6. By iterating this procedure we can approximately find proper orders for the given problems. Furthermore, rarely matched hyperedges are substituted in step 8.

IV. Simulation Results

We will demonstrate simulation results of hypernetwork models for 3 pattern classification problems. The accuracies of other probabilistic models are averages of 10 trials.

A. Optical Recognition of Handwritten Digits Data

We selected this data for test because of easy visualization of data. We can estimate the result easily. There are 3,823 training and 1,797 test data. The dimensionality of the input space of the original data is $32 \times 32 = 256$. It is reduced to 49 by first converting each 4×4 block into a pixel, and deleting side pixels that are always empty. The value of the converted pixel is decided by the number of non-zero pixels in original 4×4 pixels exceeds a threshold value. Figure 4 is the sample images of transformed handwritten digits per class. White pixels and black pixels mean that the corresponding feature values of the patterns are 0 and 1, respectively.

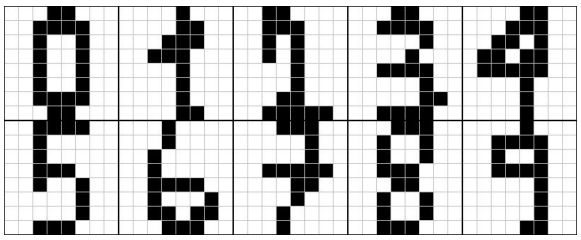

Fig. 4. Sample images of handwritten digits data

Figure 5 shows the accuracy change during the evolution. Five w values that control the number of hyperedges substituted at first epoch are tested. The accuracy converges best when $w = 2.5$. During the earlier epochs, there are tendencies of accuracy fluctuation. These are due to exploring large problem spaces that is changing many hyperedges. However, as the epochs go, the accuracies converge to high values.

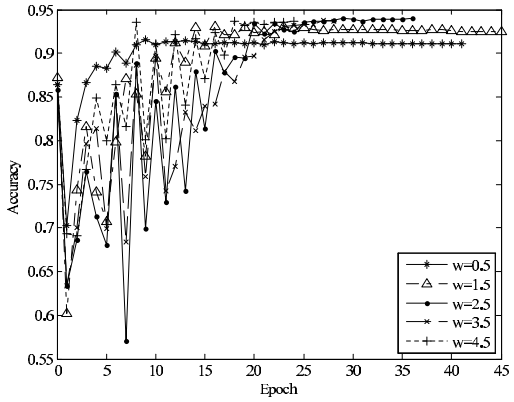

Fig. 5. Accuracy of handwritten digits data. The number of substituted hyperedges is proportional to the given weight parameter w.

Figure 6 shows the order distribution of hyperedges at the last epoch. Although the order distribution was uniform at the initialization, as the hypernetwork evolves, the proper order distribution is found. As the initial number of substituting hyperedges increases, the number of hyperedges that were generated at earlier epochs with relatively uniform sampling order distribution decreases. Therefore, the variance of orders of hyperedges also decreases in general.

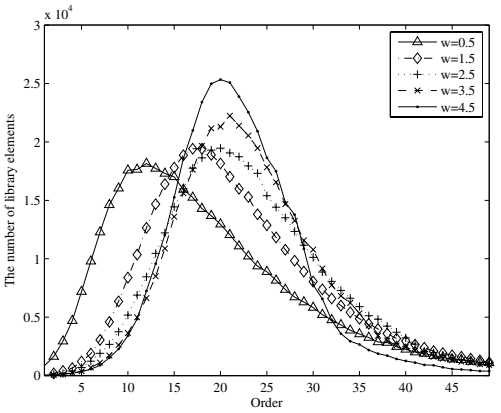

Fig. 6. Order distribution of hyperedges of handwritten digits data

Table I compares the accuracy to that of other algorithms. All other algorithms are tested using WEKA [2]. We included algorithms whose performance is known to be very good empirically such as Random Forests or Boosted Trees [9]. The result shows that after selecting proper w value, the

[2] http://www.cs.waikato.ac.nz/ml/weka/

accuracy of the hypernetwork model was very good for this problem.

TABLE I
COMPARISONS WITH OTHER ALGORITHMS ON HANDWRITTEN DIGITS

Algorithm	Accuracy
Random Forests (Number of trees=50)	0.941
Hypernetwork Model (w=2.5)	0.940
KNN (k=4)	0.935
AdaBoost (Weak Learner: C4.5, Number of iterations: 50)	0.933
SVM (Polynomial Kernel)	0.914
MLP	0.905
Naïve Bayes	0.874
C4.5	0.849

Figure 7 shows 5 visualized hyperedges whose classes are 0 and fitness values are very high. They are selected from the evolved hypernetwork with $w = 2.5$. The numbers of correct matches are 247, 243, 238, 225, and 222, respectively, and the numbers of wrong matches are 3, 26, 20, 0, and 17, respectively. Gray pixels mean that the hyperedges do not contain corresponding feature variable, white pixels correspond to feature variable whose values are 0, and black pixels correspond to those values are 1. In most cases, pixels near center are white, pixels rounding the center are black, and pixels out of the round are white. Comparing to a sample 0 image in Figure 4, we can easily imagine the 0 shape from the library elements shown in Figure 7. Therefore the hyperedges can be thought to be local rules for correct pattern classification.

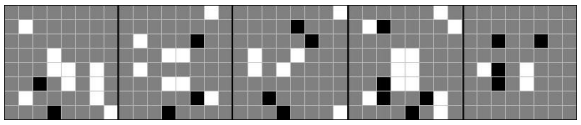

Fig. 7. Hyperedges whose classes are 0 and fitness values are very high

Figure 8 shows 5 training patterns that are source of many hyperedges remained (upper) and 5 that are rarely source of hyperedges remained (lower). The classes of first upper training patterns are 8, and four 9s, respectively. The classes of lower training patterns are 4, 1, and three 0s, respectively. Because many hyperedge whose sources are lower training patterns are substituted by hyperedges whose sources are different ones, we can regard them as noise patterns. Indeed, the shape of the lower training patterns looks noisy. Reducing the number of hyperedges whose sources are these noisy patterns can improve the performance of pattern classification.

B. SPECT heart data

There are 80 training and 187 test data. The dimensionality of the input space is 22.

Figure 9 shows the accuracy change during the evolution with 5 w values. The accuracy converges best when $w = 0.02$. Compared to Optical Recognition of Handwritten Digits, the accuracy is good when w value is very low. As

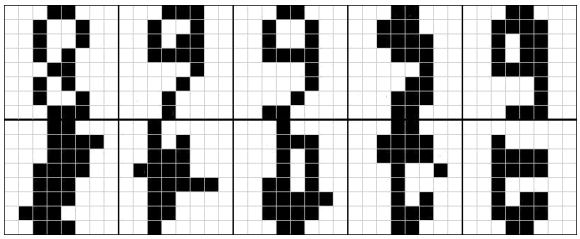

Fig. 8. Training patterns that are sources of many hyperedges remained (upper) and sources of hyperedges rarely remained (lower)

the w value is low, fluctuation tendency observed in digits domain is diminished.

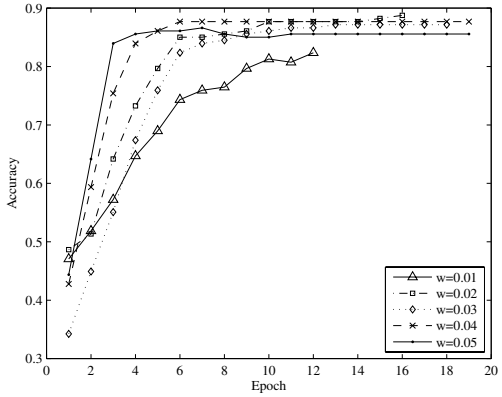

Fig. 9. Accuracy of SPECT heart data

Figure 10 shows the order distribution of hyperedges at the last epoch. Compared to digits domain, we can see that the mean of the orders is shifted down from the center, and that means the coherency of training patterns with the same class label is relatively low.

Table II compares the accuracy to that of other algorithms. For SPECT heart data, only hypernetwork model and the ensemble of CLIP4 [10] show noticeable accuracies.

TABLE II
COMPARISONS WITH OTHER ALGORITHMS ON SPECT HEART DATA

Algorithm	Accuracy
Hypernetwork Model (w=0.02)	0.888
Ensemble of CLIP4	0.887
KNN (k=1)	0.802
Random Forests (Number of trees=50)	0.777
MLP	0.759
AdaBoost (Weak Learner: C4.5, Number of iterations: 50)	0.759
C4.5	0.754
Naïve Bayes	0.749
SVM (Polynomial Kernel)	0.733

C. 1984 United States Congressional Voting Records Data

There are 435 data. The dimensionality of the input space is 16. Because there is no distinction of training and test data,

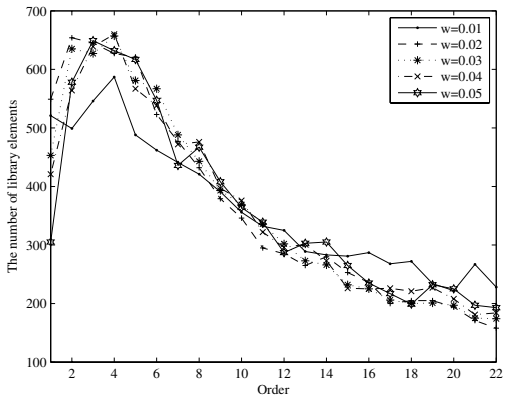

Fig. 10. Order distribution of hyperedges of SPECT heart data

we used 10 fold cross validation to measure the accuracy. The average accuracies were 0.948, 0.954, and 0.942 for $w = 0.5, 0.6$, and 0.7, respectively. Figure 11 shows the order distribution of hyperedges at the last epoch.

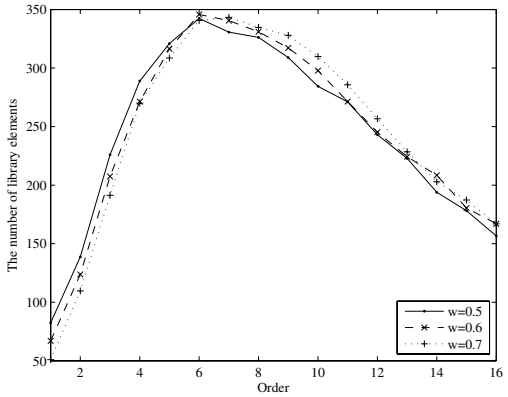

Fig. 11. Order distribution of hyperedges of Voting Records data

Table III compares the accuracy to that of other algorithms. This problem is relatively easy compared to other problems, and the accuracies of most algorithms including the hypernetwork are quite good.

V. Conclusions

This paper presented a method for evolving hypernetworks of variable-order hyperedges. The simulation results showed that the performance of the hypernetwork model for pattern classification is competitive to other conventional algorithms. In particular, the performance of the hypernetwork model on the SPECT heart data was remarkable since most conventional algorithms do not perform well on this data set. We could also observe that the hyperedges with the highest fitness values seem to be important rules for pattern classification problems.

TABLE III
Comparisons with other algorithms on Voting Records data

Algorithm	Accuracy
Random Forests (Number of trees=50)	0.964
C4.5	0.963
SVM (Polynomial Kernel)	0.961
Hypernetwork Model (w=0.6)	0.954
AdaBoost (Weak Learner: C4.5, Number of iterations: 50)	0.954
MLP	0.947
KNN (k=4)	0.931
Naïve Bayes	0.901

The proper order distribution of hyperedges for pattern classification can be found during the evolving process. Although there are some differences in the concrete shapes of order distributions in different problems, the hyperedges with very low or very high orders are weeded out. The hyperedges with very low orders often match with patterns with different class labels and the hyperedges with very high order match rarely with any pattern. It is interesting to compare the decision boundaries constructed by the hypernetworks and those by other pattern classification algorithms using subsets of the feature spaces [11], [12], [13].

The proper orders may depend on the between-class and within-class distances among patterns and the dimensionality of the input space. In terms of the between-class and within-class distances, the classification performance can be good when the between-class distances are long and within-class distances are short [14]. This can also be applied to the hypernetwork model. If the between-class distances are long, even though the hyperedges with low orders, they cannot be frequently matched to patterns with different class labels, and if the within-class distances are short, even though the hyperedges with high orders, they can be matched to patterns often.

Acknowledgement

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the National Research Lab. Program funded by the Ministry of Science and Technology (No. M10400000349-06J0000-34910), the Ministry of Education and Human Resources Development under the BK21-IT Program, and the Ministry of Industry and Commerce through the Molecular Evolutionary Computing (MEC) Project. The ICT at Seoul National University provided research facilities.

References

[1] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge Univeristy Press, 2003.
[2] B.-T. Zhang and J.-K. Kim, "DNA hypernetworks for information storage and retrieval," *Lecture Notes in Computer Science, DNA12*, vol. 4287, pp. 298–307, 2006.
[3] B.-T. Zhang and H.-Y. Jang, "A Bayesian algorithm for in vitro molecular evolution of pattern classifiers," *Lecture Notes in Computer Science, DNA10*, vol. 3384, pp. 458–467, 2005.
[4] C. Berge, *Graphs and Hypergraphs*, American Elsevier, 1976.
[5] R. Agrawal and R. Strikant, "Fast algorithms for mining association rules in large databases," *Proc. 20th Int. Conf. on Very Large Databases*, pp. 487–499, Santiago, Chile, 1994.

[6] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher, "Clustering based on association rule hypergraphs," *Research Issues on Data Mining and Knowledge Discovery*, 1997.
[7] B.-T. Zhang and H.-Y. Jang, "Molecular programming: evolving genetic programs in a test tube," *Proc. 2005 Genetic and Evolutionary Computation Conference*, vol. 2, pp. 1761–1768, 2005.
[8] S. Kim, M.-O. Heo, and B.-T. Zhang, "Text classifiers evolved on a simulated DNA computer," *Proc. 2006 IEEE Congress on Evolutionary Computation*, pp. 9196–9202, 2006.
[9] R. Caruana and A. Niculescu-Mizil, "An empirical comparision of supervised learning algorithms," *Proc. 23th International Conference on Machine Learning*, pp. 161–168, 2006.
[10] L. Kurgan and K. J. Cios, "Ensemble of Classifiers to Improve Accuracy of the CLIP4 Machine Learning Algorithm," *SPIEs International Symposium on Sensor Fusion: Architectures, Algorithms, and Applications VI*, Orlando, FL, U.S.A, 2002.
[11] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Patten Analysis and Machine Intelligence*, vol. 20, no. 8, Aug 1998.
[12] L. S. Oliveira, N. Benahmed, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Feature subset selection using genetic algorithms for handwritten digit recognition," *Proc. 14th Brazilian Symposium on Computer Graphics and Image Processing*, pp. 362–369, 2006.
[13] B.T. Zhang, P. Ohm, and H. Muhlenbein, "Evolutionary induction of sparse neural trees," *Evolutionary Computation*, vol. 5, no. 2, 213–236, 1997.
[14] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.