9-2007

Evolutionary combinatorial optimization for recursive supervised learning with clustering

Kiruthika RAMANATHAN
Singapore Management University, kiruthikar@smu.edu.sg

Sheng Uei GUAN

Citation

RAMANATHAN, Kiruthika and GUAN, Sheng Uei. Evolutionary combinatorial optimization for recursive supervised learning with clustering. (2007). *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, September 25-28*. 1168-1174.
Available at: https://ink.library.smu.edu.sg/sis_research/7395

Evolutionary combinatorial optimization for recursive supervised learning with clustering

Conference Paper · September 2007

DOI: 10.1109/CEC.2007.4424602 · Source: DBLP

2 authors:

Kiruthika Ramanathan
Agency for Science, Technology and Research (A*STAR)
81 PUBLICATIONS **589** CITATIONS
SEE PROFILE

Sheng-Uei Guan
La Trobe University; National University of Singapore; Brunel University; Xi'an Jiaoto…
257 PUBLICATIONS **2,420** CITATIONS
SEE PROFILE

Evolutionary Combinatorial Optimization for Recursive Supervised Learning with Clustering

Kiruthika Ramanathan, Sheng Uei Guan

***Abstract*— The idea of using a team of weak learners to learn a dataset is a successful one in literature. In this paper, we explore a recursive incremental approach to ensemble learning. In this paper, patterns are clustered according to the output space of the problem, i.e., natural clusters are formed based on patterns belonging to each class. A combinatorial optimization problem is therefore formed, which is solved using evolutionary algorithms. The evolutionary algorithms identify the "easy" and the "difficult" clusters in the system. The removal of the easy patterns then gives way to the focused learning of the more complicated patterns. Incrementally, neural networks are added to the ensemble to focus on solving successively difficult examples. The problem therefore becomes recursively simpler. Over fitting is overcome by using a set of validation patterns along with a pattern distributor. An algorithm is also proposed to use the pattern distributor to determine the optimal number of recursions and hence the optimal number of weak learners for the problem. In this paper, we show that the generalization accuracy of the proposed algorithm is always better than that of the underlying weak learner. Empirical studies show generally good performance when compared to other state-of-the-art methods.**

I. Introduction

RECURSIVE supervised learners are a combination of weak learners, data decomposition and integration. Instead of learning the whole dataset, neural networks are incrementally used to learn different subsets of the data, resulting in several sub solutions (or subnetworks). These subnetworks are then integrated together to form the final solution to the system. Figure 1 shows the general architecture of such learners.

In the design of recursive learners, several factors come into play in determining the generalization accuracy of the system. Factors include

1. The accuracy of the subnetworks
2. The accuracy of the integrator
3. The number of subnetworks

Kiruthika Ramanthan is with the Data Storage Institute, 5, Engineering Drive 1, Singapore 117608. (Phone: (65) 6874 8514; Fax: (65)6777 8517; Email: kiruthika_r@dsi.a-star.edu.sg).

Sheng Uei Guan is with the School of Engineering and Design, Brunel University, UK. (steven.guan@brunel.ac.uk).

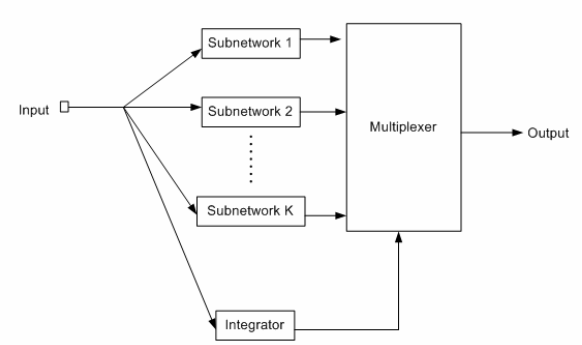

Figure 1. Architecture of the final solution of recursive learners

In RSL-CC [8], genetic algorithms are used to select subsets, which are in turn used to train subnetworks. The algorithm proposed aims to optimize the generalization accuracy of the system by finding (i) an optimal number of subnetworks (ii) Subnetworks with optimal accuracy. Here, genetic algorithms are not used to select the patterns for a subset, but to select clusters of patterns for a subset. By using this approach, we hope to group patterns into subsets and derive a more optimal partitioning of data. In this paper, we also prove that the generalization accuracy of RSL-CC is always better than, or equal to the underlying base learner.

The system proposed consists of a pre-trainer and a trainer. The pre-trainer is made up of a clusterer and a pattern distributor. The clusterer splits the data set into clusters of patterns. The pattern distributor assigns validation patterns to each of these clusterers. The trainer now solves a combinatorial optimization problem, choosing the clusters that can be learnt with best training and validation accuracy. These clusters now form the "easy" patterns which are then learnt using a gradient descent algorithm to create the first subnetwork. The remaining clusters form the "difficult" patterns. The trainer now focuses attention on the difficult patterns, thereby recursively isolating and learning increasingly "difficult" patterns and incrementally creating several corresponding subnetworks.

The use of genetic algorithms in selecting clusters is expected to be more efficient than their use in the selection of patterns for two reasons:

1. The number of combinations is now ${}^{n}C_{k}$ as opposed to ${}^{T}C_{L}$, where the number of available clusters n, is less

1-4244-1340-0/07/$25.00 ©2007 IEEE

than the number of training patterns T. Similarly, the number of clusters chosen k is is smaller than the number of training patterns chosen L. The solution space is now smaller, therefore increasing the probability of finding the better solutions.

2. The distribution of validation information is performed during pretraining, as opposed to during the training time. Validation pattern distribution is therefore a one-off process, thereby saving training time.

The rest of the paper is organized as follows. We begin with some preliminaries and related work in section 2. In section 3, we present a detailed description of the proposed Recursive supervised learning with clustering and combinatorial optimization (RSL-CC) algorithm. More details and specifics of the algorithm are then presented in section 4. Section 5 presents the results of the RSL-CC algorithm on some benchmark pattern recognition problems, comparing them with other recursive hybrid and non-hybrid techniques. In section 6, we summarize the important advantages and limitations of the algorithm and conclude with some general discussion and future work.

II. Some Preliminaries

A. Notation

m: input dimension
n: output dimension
K : number of recursions
I : input
O : output
Tr : Training
Val: Validation
P: ensemble of subsets
S: Neural network solution
E: Error
T: number of training patterns
r: Recursion index
N_{chrom}: Number of chromosomes
N_c: Number of clusters

B. Problem Formulation

Let $\mathbf{I}_{Tr} = \{\mathbf{I}_1, \mathbf{I}_2, ...\mathbf{I}_T\}$ be a representation of T training inputs. $\mathbf{I}_j$ is defined, for any $j \in T$ over an *m* dimensional feature space, i.e, $\mathbf{I}_j \in R^m$. Let $\mathbf{O}_{Tr} = \{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_T\}$ be a representation of the corresponding *T* training outputs. $\mathbf{O}_j$ is a binary string of length *n*. $\mathbf{Tr}$ is defined such that $\mathbf{Tr} = \{\mathbf{I_{tr}}, \mathbf{O_{tr}}\}$.

Similarly $\mathbf{I}_{Val} = \{\mathbf{I}_{v,1}, \mathbf{I}_{v,2}, ..., \mathbf{I}_{v,T_{val}}\}$ and $\mathbf{O}_{Val} = \{\mathbf{O}_{v,1}, \mathbf{O}_{v,2}, ..., \mathbf{O}_{v,T_{val}}\}$ represent the input and output patterns of a set of validation data, such that $\mathbf{Val} = \{\mathbf{I}_{Val}, \mathbf{O}_{Val}\}$. We wish to take $\mathbf{Tr}$ as training patterns to the system and $\mathbf{Val}$ as the validation data and come up with an ensemble of *K* subsets. Let $\mathbf{P}$ represent this ensemble of *K* subsets:

$\mathbf{P} = \{\mathbf{P^1}, \mathbf{P^2}, ..., \mathbf{P^K}\}$, where, for $i \in K$

$\mathbf{P^i} = \{\mathbf{Tr^i}, \mathbf{Val^i}\}$, $\mathbf{Tr}^i = \{\mathbf{I}_T{}^i, \mathbf{O}_{tr}{}^i\}$,

$\mathbf{Val^i} = \{\mathbf{I}_{Val}{}^i, \mathbf{O}_{Val}{}^i\}$

Here, $\mathbf{I}_{Tr}{}^i$ and $\mathbf{O}_{Tr}{}^i$ are $m \times T^i$ and $n \times T^i$ matrices respectively and $\mathbf{I}_{Val}{}^i$ and $\mathbf{O}_{Val}{}^i$ are $m \times T_{Val}{}^i$ and $n \times T_{Val}{}^i$ matrices respectively, such that $\sum_{i=1}^{K} T^i = T$ and $\sum_{i=1}^{K} T_{Val}{}^i = T_{Val}$.

A set of neural networks $S = \{S^1, S^2, ..., S^K\}$ is to be found where S^1 solves $\mathbf{P^1}$, S^2 solves $\mathbf{P^2}$ and so on.
We need to find $\mathbf{P}$ such that it fulfills two conditions:

Condition set 1

1. The individual subsets can be trained with a small mean square training error, i.e, $E_{tr}{}^i = \mathbf{O_{tr}}^i - S^i(\mathbf{I_{tr}}^i) \longrightarrow 0$,
2. None of the subsets $\mathbf{Tr^1}$ to $\mathbf{Tr^K}$ are overtrained, i.e, $\sum_{i=1}^{j} E_{val}{}^i < \sum_{i=1}^{j+1} E_{val}{}^i ; j, j+1 \in \boldsymbol{K}$

C. Related Work

As mentioned in the introduction, several methods are used to select a suitable partition $\mathbf{P} = \{\mathbf{P^1}, \mathbf{P^2}, ..., \mathbf{P^K}\}$ that fulfils the conditions 1 and 2 above. In this section we shall discuss some of them in greater detail and discuss their pros and cons:

1) Subset selection for supervised training

The process of finding the subsets $\mathbf{P} = \{\mathbf{P^1}, \mathbf{P^2}, ..., \mathbf{P^K}\}$ for training subnetworks plays an important part in determining the generalization accuracy of the ensemble learning methods. Several works, including topology based selection [12] and recursive pattern based training [7], use evolutionary algorithms to choose subsets of data.

Algorithms such as active data selection [1], boosting [2] and multi-sieving [3] implement this subset choice using neural networks trained with various training methods. Multiple weak learners have also been used and the best weak learner given the responsibility of training and solving a subset [4]. Other algorithms make use of more brute force methods such as the use of the Mahalanobhis distance [5]. Even other algorithms such as the output parallelism algorithm manually decompose their tasks [6-8]. Clustering has also been used to decompose datasets in some cases [9]. In the rest of this section, we will look at some of these approaches in greater detail.

2) Manual Decomposition

Output parallelism was developed by Guan and Li [6]. The idea involves splitting a n-class problem into n two-class problems. Each of the n sub problems consists of two outputs, $class_i$ and $\overline{class_i}$. Guan et al [8] later added an integrator in the form of a pattern distributor to the system. The output parallelism algorithm essentially develops a set of n subnetworks, each catering to a 2-class problem and integrates them using a pattern distributor.

While the algorithm is shown to be effective in terms of both training time and generalization accuracy, a major drawback of the algorithm is its class based manual decomposition. In fact, research has been carried out [8] shows empirically that optimum decomposition is problem dependent. While automatic algorithms have been developed to overcome this problem of manual decomposition [10], the net result is an algorithm which is computationally expensive. The other concern associated with the output parallelism is that it can only be applied to classification problems.

3) Genetic algorithm based partitioning algorithms

GA based algorithms are shown to be effective in partitioning the datasets into simpler subsets. One interesting observation was that using GA for partitioning leads to simpler and easily separable subsets [7].

However, the problem with these approaches is that the use of GA is computationally costly. Also, a criterion has to be established to separate the difficult patterns from the first, and in the case of various algorithms, criterions used include the history of difficulty, the degree of learning [11]etc.

4) Brute force methods

Brute force mechanisms include the use of a distance or similarity measure such as the Mahalanobhis distance [5]. Subsets are selected to ensure good separation of patterns in one output class from another, resulting in good separation between classes in each subset. A similar method, bounding boxes, has been developed [4] which clusters patterns based on their Euclidean distances and overlap from patterns of other classes.

While the objective of the approach is direct, much effort in involved in pretraining as it involves brute force distance computation. In [4], the authors have shown that the complexity of the brute force distance measure increases almost exponentially with the problem dimensionality.

5) Neural Network approaches

Neural network approaches to decomposition are similar to genetic algorithms based approaches. Multisieving [3], for instance, allows a neural network to learn patterns, after which it isolates the network and allows another network to learn the "unlearnt" patterns. The process continues until all the patterns are learnt.

Boosting [2] is similar in nature, except that, instead of isolating the "learnt" patterns, a weight is assigned to them. An unlearnt pattern has more weight and is thus concentrated on by the new network. Boosting is a successful method and has been regarded as one of the best off the shelf classifiers" in the world.

One problem with boosting, however, is the problem dependent number of weak learners in the system. Multisieving, while unhampered by this problem focuses on training accuracy, with no mention of generalization accuracy.

6) Clustering based approaches

Clustering based approaches to task decomposition for supervised learning are many fold [9]. Most of them divide the dataset into clusters and solve individual clusters separately. However, we feel that this particular approach is not very good as it creates a bias. Many subsets have several patterns in one class and few patterns in other classes. The networks created are therefore not sufficiently robust.

III. THE RSL-CC ALGORITHM

The RSL-CC algorithm can be described in two parts, *pre-training* and *training.* In this section, we explain these two aspects of training in detail.

1) Pretraining

1. We express $\mathbf{I}_{Tr}, \mathbf{O}_{Tr}, \mathbf{I}_{Val}$ and $\mathbf{O}_{Val}$ as a combination of *m* classes of patterns, i.e,

$$\mathbf{I}_{\mathbf{tr}} = \{\mathbf{I}^{\mathbf{C1}}, \mathbf{I}^{\mathbf{C2}}, ..., \mathbf{I}^{\mathbf{Cn}}\}$$

$$\mathbf{O}_{\mathbf{tr}} = \{\mathbf{O}^{\mathbf{C1}}, \mathbf{O}^{\mathbf{C2}}, ..., \mathbf{O}^{\mathbf{Cn}}\}$$

$$\mathbf{I}_{Val} = \{\mathbf{I}^{\mathbf{C1}}_{Val}, \mathbf{I}^{\mathbf{C2}}_{Val}, ..., \mathbf{I}^{\mathbf{Cn}}_{Val}\}$$

$$\mathbf{O}_{Val} = \{\mathbf{O}^{\mathbf{C1}}_{Val}, \mathbf{O}^{\mathbf{C2}}_{Val}, ..., \mathbf{O}^{\mathbf{Cn}}_{Val}\}$$

2. The datasets $\mathbf{I}_{Tr}, \mathbf{O}_{Tr}, \mathbf{I}_{Val}$ and $\mathbf{O}_{Val}$ are split into *n* subsets,

$$\left\{\mathbf{I}^{\mathbf{C1}}, \mathbf{O}^{\mathbf{C1}}, \mathbf{I}^{\mathbf{C1}}_{Val}, \mathbf{O}^{\mathbf{C1}}_{Val}\right\}, \left\{\mathbf{I}^{\mathbf{C2}}, \mathbf{O}^{\mathbf{C2}}, \mathbf{I}^{\mathbf{C2}}_{Val}, \mathbf{O}^{\mathbf{C2}}_{\mathbf{v}}\right\}, ..., \left\{\mathbf{I}^{\mathbf{Cn}}, \mathbf{O}^{\mathbf{Cn}}, \mathbf{I}^{\mathbf{Cn}}_{Val}, \mathbf{O}^{\mathbf{Cn}}_{Va;}\right\} \quad \textbf{(1)}$$

where each subset in expression (1) consists of only patterns from one class.

3. Each subset, $\left\{\mathbf{I}^{\mathbf{Ci}}, \mathbf{O}^{\mathbf{Ci}}, \mathbf{I}^{\mathbf{Ci}}_{Val}, \mathbf{O}^{\mathbf{Ci}}_{Val}\right\}, i \in n$, now undergoes a clustering treatment as below:

- Cluster $\mathbf{I}^{\mathbf{Ci}}$ into k^{Ci} partitions or *natural clusters*. Any clustering algorithm can be used, including SOMs [12], K-means [13], Agglomerative Hierarchical clustering [14], Bounding Boxes [4].
- Using a pattern distributor, patterns in $\mathbf{I}^{\mathbf{Ci}}_{Val}$ are assigned to one of the k^{Ci} patterns. In this paper, we implement the pattern distributor using the nearest neighbor algorithm [15].
- Each pattern, validation or training, in a given cluster $j^{\mathrm{Ci}}, j \in k$, has the same output pattern.

4. The total number of clusters is now the sum of the natural clusters formed in each class.

$$N_c = \sum_{i=1}^{n} k^{ci} \qquad \textbf{(2)}$$

2) Training

1. Number of recursions *r*=1.
2. A set of binary chromosomes are created, each chromosome having N_c elements, where N_c is defined as in (2).

An element in a chromosome is set at 0 or 1, 1 indicating that the corresponding cluster will be selected for solving using recursion *r*.

3. A genetic algorithm is executed to minimize E_{tot}, the average of the training and validation errors E_{Tr} and E_{Val}

$$E_{tot} = \frac{1}{2}\left(E_{Tr} + E_{Val}\right) \quad (3)$$

4. The best chromosome $Chrom_{best}$ is a binary string with a combination of 0s and 1s, with the size N_c. The following steps are executed.

a. $N_c^r = 0$, $\mathbf{Tr^r} = [], \mathbf{Val^r} = []$

b. For *e*=1 to N_c

i. if $Chrom_{best}(e) == 1$

$N_c^r ++$

$\mathbf{Tr^r} = \mathbf{Tr^r} + \mathbf{Tr_{chrom_{best}(e)}}$

$\mathbf{Val^r} = \mathbf{Val^r} + \mathbf{Val_{chrom_{best}(e)}}$

c. The data is updated as follows:

$\mathbf{Tr} = \mathbf{Tr} - \mathbf{Tr^r}$

$\mathbf{Val} = \mathbf{Val} - \mathbf{Val^r}$

$N_c = N_c - N_c^r$

$r ++$

d. $\mathbf{Tr^r}$ and $\mathbf{Val^r}$ are used to find S^r, the solution network solving the subset of data in recursion r.

5. Steps 2 to 4 are repeated with the new values of $\mathbf{Tr}$, $\mathbf{Val}$, N_c and r.

3) Simulation

Simulating and testing the RSL-CC algorithm is implemented using a Kth nearest neighbor (KNN) [15] based pattern distributor. KNN was used to implement the pattern distributor due to the ease of its implementation. At the end training, we have *K* subsets of data. A given test pattern is matched with its nearest neighbor. If the neighbor belongs to subset *i*, the pattern is also deemed as belonging to subset *i.* The solution for subset *i* is then used to find the output of the pattern. A multiplexer is used for this function. The KNN distributor provides the select input for the multiplexer, while the outputs of subnetworks 1 to K are the data inputs. This process is illustrated by Figure 2.

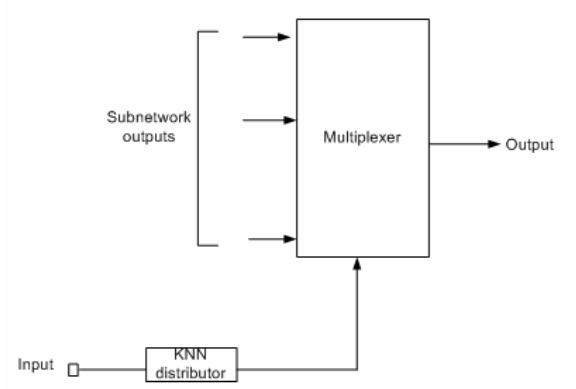

Figure 2. Using a KNN distributor to test the RSL system

IV. Details of the RSL-CC Algorithm

A. Termination criteria

The grouping of patterns means that clusters of patterns are selected for each subset. Further, in contrast with any other method, the GA based recursive subset selection proposed selects the optimal subset combination.

Theorem 1. *Given an infinite pool of chromosomes, the decomposition of data performed by RSL-CC is the optimal decomposition solution based on the available data.*

Proof:

If the subset chosen at recursion i is not optimal, an alternative subset will be chosen. The largest possible alternative subset is the training set for the recursion, $\mathbf{Tr^r} = \mathbf{Tr} - \sum_{i=1}^{r-1} \mathbf{Tr^i}$. Therefore is decomposition at a particular recursion is suboptimal, no decomposition will be performed and the training will complete.

Theorem 2. *The worst case generalization accuracy of RSL-CC is the generalization accuracy of the base learner*.

Proof:

Theorem 2 follows from Theorem 1. The generalization accuracy will equal that of the base learner if the largest subset is chosen at the end of global training in the first recursion, i.e, $\mathbf{P^1} = \mathbf{Tr}^1$.

We therefore have the following termination conditions:

Condition set 2: Termination conditions

Condition 1: No clusters of patterns are left in the system.
Condition 2: Only one cluster is left in the remaining data.
Condition 3: More than one cluster is present in the remaining data, but all the clusters belong to the same class.

Condition 1 occurs when the optimal choice in a system is to choose all the clusters as decomposition is not favorable. Conditions 2 and 3 describe dealing with cases when it is not necessary to create a classifier due to the homogeneity of output classes.

B. Fitness function for combinatorial optimization

In Equation (3), we defined the fitness function as an average of the training and validation errors obtained when training the subset selected by the chromosome, $\frac{1}{2}(E_{Tr} + E_{Val})$. The values of E_{Tr} and E_{Val} are calculated by designing a 3 layered neural network with an arbitrary number of hidden nodes (we use 10 nodes, for the purpose of this paper). The Training and Validation subsets selected by the corresponding chromosome are used to train the network. The best performing network is chosen as $Chrom_{best}$.

V. Experimental Results

A. Problems Considered

The table below summarizes the three classification problems considered in this paper. The problems were chosen such that they varied in terms of input and output dimensions as well as in the number of training, testing and validation patterns made available. All the datasets, other than the two-spiral dataset, are benchmark sets obtained from the UCI repository [18].

The results of the two-spiral dataset were compared with constructive backpropagation, multisieving and the topology based subset selection algorithms only. This was because the two spiral problem is two-class. Therefore implementing the output parallelism will not make a difference to the results obtained by CBP.

The two spiral dataset consists of 194 patterns. To ensure a fair comparison to the Dynamic subset selection algorithm [11], test and validation datasets of 192 patterns were constructed by choosing points next to the original points in the dataset as mentioned in the paper.

Table 1. Summary of the problems considered

Problem Name	VOWEL	LETTER RECOGNITION	TWO SPIRAL
Training set size	495	10000	194
Test set size	248	5000	192
Validation set size	247	5000	192
Number of Inputs	10	16	2
Number of Outputs	11	26	2

B. Experimental Parameters and Control Algorithms Implemented

Table 2 summarizes the parameters used in the experiments. As we wish for the RSL-CC technique to be as problem independent as possible, we make all the experimental parameters constant for all problems and as given below. Each experiment was run 40 times, with 4-fold cross validation.

For comparison purposes, the following algorithms were designed and implemented. The constructive backpropagation algorithm [16] was implemented as a single staged (non hybrid) algorithm which conducts gradient descent based search with the possibility of evolutionary training by the addition of a hidden node. The Multisieving algorithm [3] (recursive non hybrid pattern based selection) was implemented to show the necessity to find the correct pseudo global optimal solution.

The following control experiments were carried out based on the multisieving algorithm and the dynamic topology based subset finding algorithm. Both the versions of output parallelism implemented also show the effect of hybrid selection.

In order to illustrate the effect of the GA based combinatorial optimization, we also implement the single cluster algorithm explained in section 2 [9]. The algorithm, in contrast to the RSL-CC algorithm, simply divides the data into clusters and develops a network to solve each cluster separately.

Table 2 summarizes the parameters used. For clustering, the agglomerative hierarchical clustering is employed with complete linkage and cityblock distance mechanism. Using thresholding, the natural clusters of patterns in each class were obtained. AHC was preferred to other clustering methods such as K-means or SOMs due to its parametric nature and since the number of target clusters is not required beforehand.

1) GA implementation

Single point crossover is implemented with a probability of 0.9 and at each epoch, an element has a mutation probability of 0.1, i.e, each element can be converted from 0 to 1 or vice versa with a probability of 0.1. Parental selection is performed using the Roulette Wheel. The worst ranked parents are replaced with the created offspring.

Table 2. Summary of parameters used

Parameter		Value
Evolutionary search parameters	Population Size	50
	Offspring Population size	35
	Number of generations	100
	Crossover probability	0.9
	Mutation probability	0.1
Neural Network Parameters	Backpropagation learning rate	10^{-2}
	Number of neighbors in the KNN pattern distributor	1

C. Results

Tables 3 to 5 summarize the training time and generalization accuracies From the tables above, we can observe that the generalization error of the RSL-CC is a general improvement over other recent algorithms. A particularly

significant improvement can be observed in the vowel dataset.

Table 3. Summary of the results obtained from the VOWEL problem (38 clusters, 12 recursions)

Algorithm used	Training time (s)	C. error (%)
Constructive Backpropagation	237.9	37.16
Multisieving with KNN pattern distributor	318.23	39.43
Output Parallelism	418.9	25.54
Output parallelism with pattern distributor	534.3	24.89
Single clustering	458.43	25.24
RSL-CC	**547.37**	**9.84**

Table 4. Summary of the results obtained from the LETTER RECOGNITION problem (100 clusters, 16 recursions)

Algorithm used	Training time (s)	C. error (%)
Constructive Backpropagation	20845.05	21.672
Multisieving with KNN pattern distributor	55349	65.04
Output Parallelism	42785.4	20.06
Output parallelism with pattern distributor	45625.4	18.636
Single clustering	N.A	NA
RSL-CC	**12682**	**13.04**

Table 5. Summary of the results obtained from the TWO-spiral problem (4 clusters, 2 recursions)

Algorithm used	Training time (s)	C. error (%)
Constructive Backpropagation	15.58	49.38
Multisieving with KNN pattern distributor	35.89	23.61
Dynamic Topology Based subset selection (TSS)	**-**	***28.0***
Single clustering	14.35	10.82
RSL-CC	**30.61**	**10.82**

There is some tradeoff observed in terms of training time. The training time for the RSL-CC algorithm for the vowel and two-spiral problems are higher than other methods. However, it is interesting to note that the training time for the Letter Recognition problem is more than 50% less than any of the recent algorithms. It is felt that this reduction is training time comes from the reduction of the problem space from the selection of patterns to the selection of clusters, where clusters are selected from 100 possible clusters as opposed to selecting subsets from 10000 patterns, thereby reducing the solution space by 100 fold.

On the other hand, for the vowel problem, the problem space is reduced by only about 13 fold. The performance of RSL-CC is more efficient when the reduction of the problem space is more significant than the GA-based combinatorial optimization.

VI. Conclusions and Future Directions

In this paper, we present the RSL-CC algorithm which divides the problem space into class based clusters, where a combination of clusters will form a subset. The problem therefore becomes a combinatorial optimization problem, where the clusters chosen for the subset becomes the parameter to be optimized. Genetic algorithms are used to solve this problem to select a good subset.

The subset chosen is then trained separately, and the combinatorial optimization problem is repeated with the remaining clusters. The situation progresses recursively, incrementally adding neural networks to the solution set, until all the patterns are learnt. The subnetworks are then integrated using a KNN based pattern distributor and a multiplexer.

Results show that reducing the problem space into clusters produces generalization accuracies which are either comparable to or better than other recent algorithms in the same domain.

Future directions would include parallelizing the RSL-CC algorithm and exploring the use of other clustering methods such as K-means or SOMs on the algorithm. The study of the effect of various clustering algorithms will help us determine better the algorithm simplicity and the robustness. Also to be studied and determined are methods to further reduce the training time of combinatorial optimization, alternative fitness functions and ways to determine the robustness of class based clustering.

References

[1] Zhang BT, Cho D Y (1998), Genetic Programming with Active Data selection, Lecture notes in Computer Science, 1585, pp146-153G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[2] R.E. Schapire (1999). A brief introduction to boosting. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence

[3] Lu B L, Ito K, Kita H, Nishikawa Y(1995), Parallel and modular multi-sieving neural network architecture for constructive learning, In proceedings of the 4th International Conference on Artificial Neural Networks. 409, 92-97.

[4] Ramanathan K, Guan S U, Iyer L R (2006), Multi Learner Based Recursive Training, In Proc of IEEE conf on Cybernetics and Intelligent systems, pp1-5

[5] Fukunaga K (1990), Introduction to Statistical Pattern Recognition, Academic Press, Boston

[6] Guan S U and Li S (2002), Parallel Growing and Training of Neural networks using Output Parallelism, IEEE transactions on Neural Networks, 13(3), pp 542-550

[7] Guan S U and Ramanathan K (2004), Recursive Percentage Based Hybrid Pattern Training, In Proc of IEEE conf on Cybernetics and Intelligent systems, pp455-460
[8] Ramanathan Kiruthika and Sheng Uei Guan (2007), Clustering and combinatorial optimization in recursive supervised learning, Journal of Combinatorial Optimization, 13(2), pp 137-152
[9] Guan S U, Neo T N and Bao C (2004), "Task Decomposition Using Pattern Distributor", Journal of Intelligent Systems 13 (2), 123-150
[10] Engelbrechet A P and Brits R (2002), Supervised Training Using an Unsupervised Approach to Active Learning, Neural Processing Letters, 15(3), pp247-260
[11] Guan S U, Qi Y, Tan S K, Li S (2005), "Output Partitioning of Neural Networks", Neurocomputing, 68, 38-53
[12] Lasarzyck CWG Dittrich P, Banzhaf W (2004), Dynamic subset selection based on a Fitness Case Topology, Evolutionary Computation, 12(4) pp 223-242
[13] Kohonen, T (1997), Self organizing maps. Berlin: Springer-Verlag
[14] J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1, 281-297
[15] Jain A K, Murty M N, Flynn P J (1999), Data clustering: A review, ACM Computing Surveys, 31(3), pp 264-323
[16] Wong M A and Lane T (1983). A kth nearest neighbor clustering procedure. Journal of the Royal Statistical Society (B), 45(3), pp362-368
[17] Lehtokangas (1999), Modeling with constructive Backpropagation, Neural Networks, 12, pp 707
[18] The UCI Machine Learning repository: http://www.ics.uci.edu/~mlearn/MLRepository.html.