Novel approaches to clustering, biclustering algorithms based on adaptive resonance theory and intelligent control

Sejun Kim

NOVEL APPROACHES TO CLUSTERING, BICLUSTERING ALGORITHMS BASED ON ADAPTIVE RESONANCE THEORY AND INTELLIGENT CONTROL

by

SEJUN KIM

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2016

Approved by

PUBLICATION DISSERTATION OPTION

This dissertation has been prepared in the format of the publication option following the Missouri University of Science & Technology thesis template and four articles are presented.

Paper 1, Pages 12-24 "A GPU based Parallel Hierarchical Fuzzy ART Clustering," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, with Donald C. Wunsch II

Paper 2, Pages 26-49 "Hierarchical BARTMAP: A Novel Innovation in Biclustering Algorithms," submitted in *IEEE Transactions on Neural Networks and Learning Systems*, with Donald C. Wunsch II

Paper 3, Pages 50-70 "Value Gradient Learning based Semi-Supervised Support Vector Machines," submitted in *IEEE Transactions on Neural Networks and Learning Systems*, with Donald C. Wunsch II

Paper 4, Pages 71-91 "Biclustering based Prediction with Supervised Biclustering ARTMAP," to be submitted in *IEEE Transactions on Knowledge and Data Engineering*, with Donald C. Wunsch II

ABSTRACT

The problem of clustering is one of the most widely studied area in data mining and machine learning. Adaptive resonance theory (ART), an unsupervised learning clustering algorithm, is a clustering method that can learn arbitrary input patterns in a stable, fast and self-organizing way. This dissertation focuses on unsupervised learning methods, mostly based on variations of ART.

Hierarchical ART clustering is studied by generating a tree of ART units with GPU based parallelization to provide fast and finesse clustering. Experiment results show that the our method achieves significant training speed increase in generating deep ART trees compared with that from non-parallelized version.

In order to handle high dimensional, noisy data more accurately, a hierarchical biclustering ARTMAP (H-BARTMAP) is developed. The nature of biclustering, which considers the correlation of each members in clusters, combined with the concept of hierarchical clustering, provides highly accurate experimental results, especially in bioinformatics data sets.

The third paper focuses on applying the biclustering concept to a supervised learning method, named supervised BARTMAP (S-BARTMAP). Experimental results on high dimensional data sets show that S-BARTMAP is capable of making better predictions compared with those from other math based and machine learning methods

The final paper focuses on solving the semi-supervised support vector machine (S^3VM) optimization problem with the aid of value gradient learning (VGL). By applying a reinforcement learning method to a semi-supervised problem results in a solid classification performance in terms of cluster validation, better than algorithms from previous studies.

ACKNOWLEDGMENTS

Partial support for this research from the National Science Foundation, the NVIDIA corporation, the Army Research Office, the Missouri S&T Intelligent Systems Center, and the Mary K. Finley Missouri Endowment are gratefully acknowledged.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

1. INTRODUCTION

Nowadays, data stream from daily life; from social network; from computers, credit cards and mobile devices; from the infrastructure of cities; from sensors in buildings, planes, and factories. The data is so fast that the total accumulation of the past two years exceeds the prior record of human civilization[1]. While the quantity of data is a challenge, extracting information out of data, the so-called data mining, is the critical problem. While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. One of the most commonly used data mining tool is clustering, which ties the data items into groups according to distances or logical relationships[2, 3].

In[4], a cluster is defined as "a set of entities which are alike, and entities from different clusters are not alike", implying the internal homogeneity and external separation[5, 6]. The procedure of cluster analysis can be depicted with the following four steps[7]:

1. Feature selection or extraction,
2. Clustering algorithm design or selection,
3. Cluster validation,
4. Result interpretation.

The flow chart of the steps are shown in Fig.1.1. In this dissertation, we focus on various clustering methods, mostly based on a neural network based adaptive resonance theory and its application.

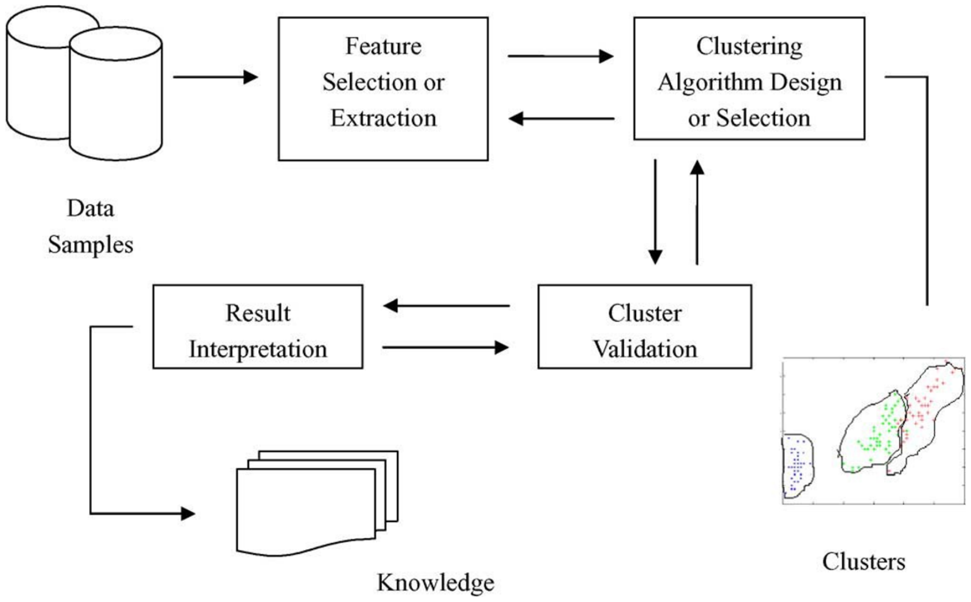

Feature Selection or Extraction
Clustering Algorithm Design or Selection
Data Samples
Result Interpretation
Cluster Validation
Knowledge
Clusters

Figure 1.1. Clustering procedure. The basic process of cluster analysis consists of four basic steps with feedback between each other

1.1. ADAPTIVE RESONANCE THEORY

The adaptive resonance theory (ART) was first introduced by Grossberg in 1976[8, 9] in order to analyze how brain networks can autonomously learn in real time about a changing world in a rapid but stable fashion. Various classes of ART neural network architectures such as ART1[10], ART2[11], Fuzzy ART[12], ARTMAP[13], Fuzzy ARTMAP[14] and Distributed ART and ARTMAP[15] were then developed with increasingly powerful learning and pattern recognition in either an unsupervised or a supervised mode.

In summary, an ART network includes a choice process and a match process as its key parts. The choice process picks up the most likely node (cluster) for an input pattern. If the template of the chosen node is sufficiently similar to the input pattern to satisfy a vigilance parameter ρ, then the node resonates and learns: its template is updated to respond to the new input pattern. Otherwise, the node is reset, and the next most likely node is chosen. If no existing node satisfies the match criterion, then a new uncommitted node is recruited. Thus, ART incrementally produces nodes necessary to represent clusters of input patterns. Fig.1.2 shows the simplified configuration of an ART structure, which involves an input processing field, F_1 layer, a clustering field F_2 layer, and a vigilance and reset subsystem.

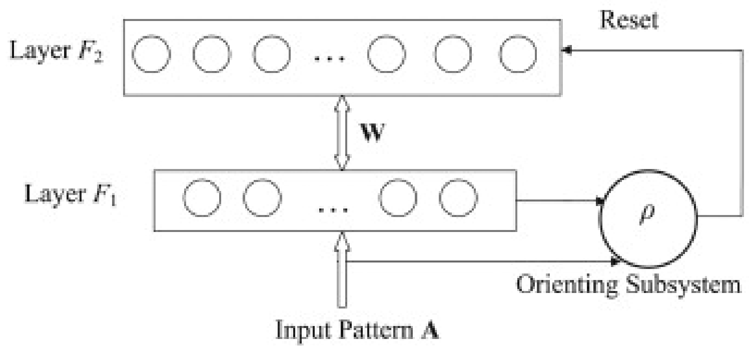


Figure 1.2. Simplified configuration of ART. The architecture consisting of an input layer F_1, a clustering layer F_2 and a reset subsystem.

There are two sets of connections between each node in F_1 layer and each node in F_2 layer. F_1 layer is connected to F_2 layer by bottom-up weights while F_2 layer is connected to F_1 layer by top-down weights, the so called templates. The connection weights between these two layers can be modified according to two different learning rules. The F_2 layer is a competitive layer which follows the winner-take-all paradigm: the node in F_2 with the largest net input becomes the candidate to learn the input pattern. Whether the candidate will learn the input pattern is decided by the vigilance and reset mechanism, which controls the degree of similarity of patterns placed in the same node.

The advantage of ART is that it does not assume the number of clusters in advance and allows the user to control the degree of similarity of patterns placed in the same cluster. Despite the great success of applying ART to clustering problems, ART architecture requires modification on handling high dimensional data sets[16]. In particular, ART focuses on similarity of patterns in the full dimensional space and thus may fail to find patterns formed in subspaces of higher dimensional space.

1.2. BICLUSTERING ARTMAP

Not only ART but also traditional clustering method often fails to recognize patterns in high dimensional data sets[17]. To overcome the limitation of clustering, the biclustering paradigm[18] was introduced and several methods have been developed based on this paradigm. In contrast to conventional clustering, biclustering, which is also called subspace clustering[19] or co-clustering[20], focuses on discovering clustering embedded in the subspaces of a data set.

Most clustering models define similarity among different objects by distance over all of the dimensions. However, distance functions are not always adequate in capturing correlations among the items. The correlation between two vectors X and Y that measures the grade of linear dependency is defined by:

$$\delta(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{n\sigma_X \sigma_Y}, \tag{1}$$

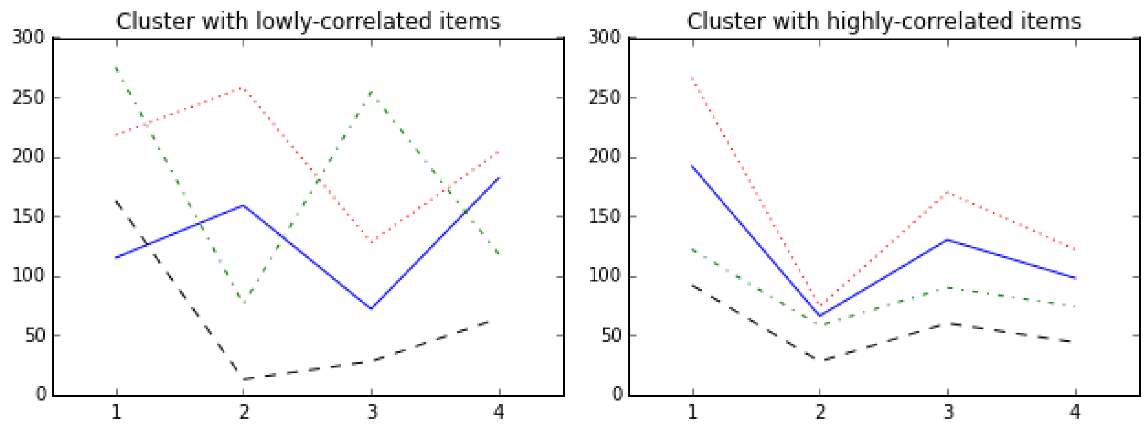
where $cov(X,Y)$ is the covariance of X and Y, and $\bar{x}$ and $\bar{y}$ are the mean of values of X and Y and σ_X and σ_Y are the standard deviations of X and Y, respectively.

Given a bicluster $\mathbf{B}$ composed of N items, $\mathbf{B} = \{s_1, ..., s_N\}$, the average correlation of $\mathbf{B}$, $\delta(\mathbf{B})$, is defined by:

$$\delta(\mathbf{B}) = \frac{1}{\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \delta(s_i, s_j). \tag{2}$$

Due to $\delta(s_i, s_j) = \delta(s_j, s_i)$, only $\binom{N}{2}$ elements are considered.

Fig.1.3 presents a bicluster with lowly-correlated features and highly-correlated features. Based on Eq.2, the average correlation $\delta(\mathbf{B})$ of Fig.1.3a is 0.003 while Fig.1.3b which has perfect shifting and scaling patters has an average correlation of 1.



(a) Example of lowly-correlated items

(b) Example of highly-correlated items

Figure 1.3. Examples of two sets of biclusters with lowly-correlated features and highly-correlated features.

Xu[21] developed a biclustering version of ART, named biclustering ARTMAP (BARTMAP) which is achieved in the form of simultaneous clustering by applying two separate ART modules, named ART_a and ART_b, to generate sample and feature clusters, respectively. Inspired by ARTMAP, an inter-ART module which resides between the ART modules, measures the correlation within clusters to determine whether a newly presented sample belongs to an existing cluster candidate. The structure of BARTMAP is shown in Fig.1.4 and the overall procedure of BARTMAP is described as follows:

- Gene (feature) ART clustering: The gene inputs are distributed to the ART_b module which functions as a standard fuzzy ART and K_g gene clusters are generated.

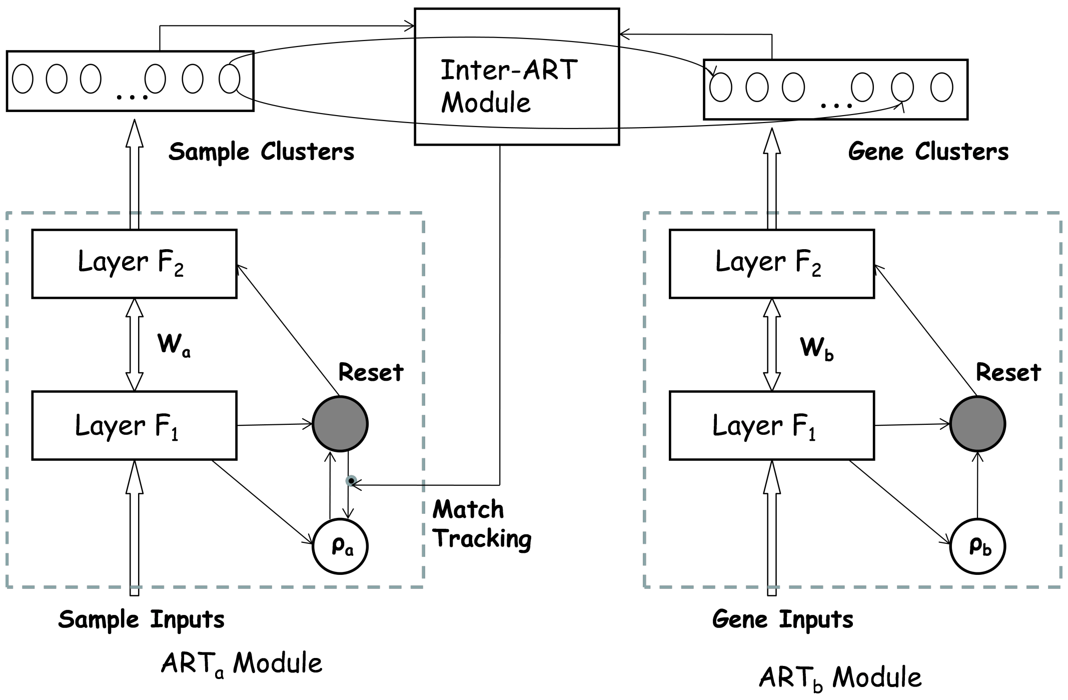


Figure 1.4. Topological structure of BARTMAP

- Sample inputs presentation: A new sample input s_k is registered to the ART_a module to find the best matching node J in ART_a and becomes a candidate cluster.
- Correlation check: The similarity between s_k and the candidate cluster $S_J = \{s_{J1}, ..., s_{JM_J}\}$ with M_J samples across every gene cluster is calculated with the average correlation function defined in Eq.2.
- Learning: If the similarity is above the correlation threshold η, the inter-ART module sends a signal to the ART_a module to associate s_k to node J and corresponding weights are updated.

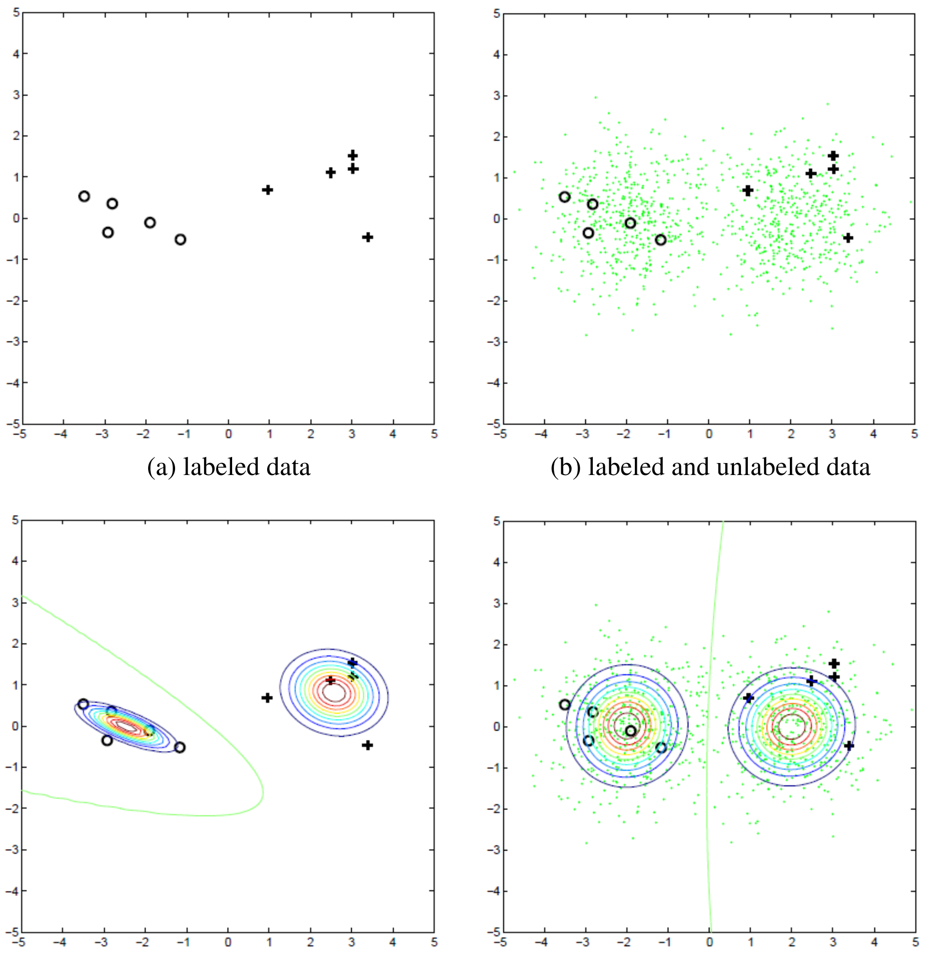
- Reset: However, if the similarity test does not pass, the inter-ART module forces ART_a to discard node J and repeats the correlation check step with the next best matching node.

BARTMAP experiments on bioinformatics data sets resulted in superior performance[21] compared to well known clustering and biclustering methods such as K-means, fuzzy ART, agglomerative hierarchical clustering and interrelated two-way clustering[22].

1.3. SEMI-SUPERVISED LEARNING

In real-world applications of machine learning, it is often the case that abundant unlabeled training examples are available, while the labeled ones are fairly expensive to obtain since labeling examples requires more human effort and expertise. Earlier versions of semi-supervised learning (SSL) had difficulties in incorporating unlabeled data directly into conventional supervised learning methods and the lack of a clear understanding of the value of unlabeled data in the learning process, the study of SSL attracted attention [23, 24]. As the demand of automatic exploitation of unlabeled data increases and the value of unlabeled data was disclosed by early analysis[25, 26], such learning method has become a hot topic.

The main difference between the outcome of supervised learning and SSL, is shown in Fig.1.5. With the addition of unlabeled items, semi-supervised learning expands the perimeter obtained from the initial supervised method to form ’soft clusters’. Note that the decision boundary shifts from Fig.1.5c to Fig.1.5d, similar to how support vector machines (SVM) learns to converge the separator where the margin between each classes is maximized. Due to its behavior, SVM was one of the earlier tools implemented for SSL[27]. By



(a) labeled data

(b) labeled and unlabeled data

(c) learned from labeled data

(d) learned from labeled and unlabeled data

Figure 1.5. Binary classification example of Gaussian mixture model on supervised cases (a), (c) and semi-supervised cases (b), (d).

maximizing the margin in the presence of unlabeled data, one learns a decision boundary that traverses through low data-density regions while respecting labels in the input space.

Assuming that semi-supervised learning problem is of binary classification, the training set consists of l labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $y_i = \pm 1$, and of u the unlabeled examples $\{\mathbf{x}_i\}_{i=l+1}^{n}$, with $n = l + u$ where $\mathbf{x}_i$ and y_i are the ith input vector and label, respectively. The goal of semi-supervised SVM (S^3VM) can be defined as a minimization problem[28] to solve

$$f(\mathbf{w},b,\mathbf{y}_u) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} V(y_i,o_i) + C^* \sum_{i=l+1}^{n} V(y_i,o_i), \tag{3}$$

where $\mathbf{y}_u = [y_{l+1},...,y_n]^T$ is the label vector of the unlabeled items, $(\mathbf{w},b)$ is the hyperplane parameters, $o_i = \mathbf{w}^T\mathbf{x}_i + b$, C and C^* are the weight parameters for labeled and unlabeled items, respectively, and V is the Hinge loss function defined as:

$$V(y_i,o_i) = \max(0, 1 - y_i o_i)^2. \tag{4}$$

1.4. OBJECTIVE

In this dissertation, an hierarchical version and a supervised modification are studied, named hierarchical BARTMAP (HBARTMAP) and supervised BARTMAP(S-BARTMAP). Experimental results on H-BARTMAP show improvement on the bioinformatics data sets, and on S-BARTMAP, the prediction rate outperforms high dimensional data sets such as sports statistics.

The study on combining two machine learning schemes - reinforcement learning and SSL - to solve the S^3VM optimization problem is discussed in this dissertation. The details on applying value gradient learning (VGL) to the S^3VM problem space , evaluation and comparison with published results are illustrated in Paper 4.

1.5. DISSERTATION ORGANIZATION

The research outcome of this study is presented by publication dissertation option. All the findings and conclusions of this research study have been submitted to technical journals and conference proceedings. The thesis is divided mainly into three sections: Introduction, paper, and conclusions and recommendations.

Introduction. An overview of the background methods with relevant studies are described. Then, the section summarizes the main objectives and motivations of this dissertation.

Paper. The main body of the thesis contains four technical papers.

Paper 1. The first paper introduces the parallel hierarchical fuzzy ART implementation on GPUs. The details of how the ART neural network is optimally parallelized on CUDA environment and presents the increase of runtime in deep neural network tree learning compared to sequential programming.

Paper 2. The second paper explains hierarchical BARTMAP (HBARTMAP) on bioinformatics data analysis. The details of how HBARTMAP overcomes the limitations of BARTMAP and methods on optimal layer selection with internal validation criteria are presented. The experimental results are justified with various evaluation methods while also compared with widely used clustering and biclustering algorithms.

Paper 3. The third paper proposes a hybrid machine learning approach to solve a semi-supervised learning optimization problem with a reinforcement learning method. Background of value gradient learning (VGL), a dual heuristic dynamic programming method and semi-supervised support vector machines (S^3VM) are detailed.

Paper 4. The final paper introduces a supervised classification and prediction method, supervised BARTMAP (S-BARTMAP) inspired by the structural conversion from ART to ARTMAP. The paper describes the utilization of biclustering similarity measurement on supervised learning and methods of the training and testing mode. The effectiveness of the proposed approach is demonstrated through experimental results on prediction with synthetic and real world statistics data sets.

Conclusion. This section summarizes the work that was accomplished in this dissertation. It also presents the key findings of all experiments and theoretical analyses, which were executed during this research study.

PAPER

I. A GPU BASED PARALLEL HIERARCHICAL FUZZY ART CLUSTERING

Sejun Kim and Donald C. Wunsch II

ABSTRACT

Hierarchical clustering is an important and powerful but computationally extensive operation. Its complexity motivates the exploration of highly parallel approaches such as Adaptive Resonance Theory (ART). Although ART has been implemented on GPU processors, this paper presents the first hierarchical ART GPU implementation we are aware of. Each ART layer is distributed in the GPU's multiprocessors and is trained simultaneously. The experimental results show that for deep trees, the GPU's performance advantage is significant.

1. INTRODUCTION

Graphics Processing Unit (GPU) programming, particularly using the NVIDIA CUDA(Compute Unified Device Architecture), has been of interest in computational intelligence, particularly for population based algorithms [29, 30, 31]. It would be of additional significant value to use GPU programming to apply its known advantages in hierarchical clustering [7].

Fuzzy Adaptive Resonance Theory (ART) is attractive for hierarchical clustering because of its speed, scalability and amenability to parallel implementation [32]. However, hierarchical fuzzy ART based on GPU engines has not been reported previously. One main constraint in CUDA is the inflexibility of memory inside the kernel meaning that the generation of dynamic arrays is limited only in the host(CPU) side. Typical tree structure algorithms implement pointers for both node creation and reference [33], which is inefficient to do in CUDA programming. The other constraint is that each ART unit is trained as data are fed sequentially. GPU implementation which focuses on the behavior of a single ART unit was achieved in [35, 36**?**], but hierarchical fuzzy ART required a different approach. The architecture is inspired from the structure of pipelining [37]. As shown in Fig. 1, even though ART networks were trained sequentially, the parallelization was accomplished successfully.

This paper describes the method used to adapt a multi-layer tree structure composed of FA units into CUDA platforms. The experimental results are presented to imply the performance boost on various data sets and parameters compared with those on conventional CPUs. Section II briefly explains FA followed by an overview of CUDA in Section III. Section IV mainly focuses on the proposed algorithm, the experimental data and results appear in Section V. Finally, conclusions and further research tasks are discussed in Section VI.

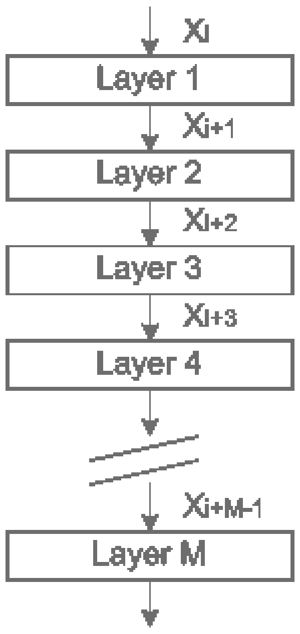


Figure 1. Data feeding example of CUDA based hierarchical fuzzy ART. The first layer which is also the root node starts with each sample. Once the training is finished, the root ART unit passes it to a child node corresponding to the winning category. Each layer loads the proper ART unit for the training for different samples as the winning category varies.

2. FUZZY ADAPTIVE RESONANCE THEORY AND THE HIERARCHICAL FUZZY ART NETWORK

Adaptive Resonance Theory (ART) is an unsupervised learning method that eliminates the “stability-plasticity dilemma". ART is capable of learning arbitrary data in both a stable and self-organizing manner [12]. ART1 deals with binary data, whereas Fuzzy ART deals with arbitrary data. Henceforth, we will be referring to Fuzzy ART.

Before the training, the data pass through a preprocess step during which they are scaled to fit into the range of [0,1]. The weight vectors $\mathbf{w}_j$ are initialized to be all 1. Let $\mathbf{x}$ be an input sample. When choosing a category, the competition in F_2 is calculated, defined as

$$T_i = \frac{|\mathbf{x} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{1}$$

where $\wedge$ is the fuzzy AND operator defined by

$$(\mathbf{x} \wedge \mathbf{y})_i = min(x_i, y_i), \tag{2}$$

and $\alpha > 0$ is the choice parameter. From the winner-take-all competition,

$$T_J = max\{T_j | \forall j\}. \tag{3}$$

The winning neuron J becomes activated and is fed back to layer F_1 for the vigilance test. If

$$\rho \leq \frac{|\mathbf{x} \wedge \mathbf{w}_J|}{|\mathbf{x}|}, \tag{4}$$

resonance occurs. Then, in layer F_2, the input $\mathbf{x}$ is categorized to J and the network is trained by the following learning rule,

$$\mathbf{w}_J(new) = \beta(\mathbf{x} \wedge \mathbf{w}_J(old)) + (1-\beta)\mathbf{w}_J(old), \tag{5}$$

where β ($0 \leq \beta \leq 1$) is the learning rate. If neuron J does not meet the match criterion, it will be reset and excluded during the presentation of the input within the vigilance test.

The hierarchical fuzzy ART network is composed of the FA[35]. The hierarchy of ART units illustrated in Fig. 2 allows the clusters to be split more finely by increasing the vigilance. An example of a modular multi-layer network architecture composed of ART networks (HART, for “Hierarchical ART") is discussed in [36].

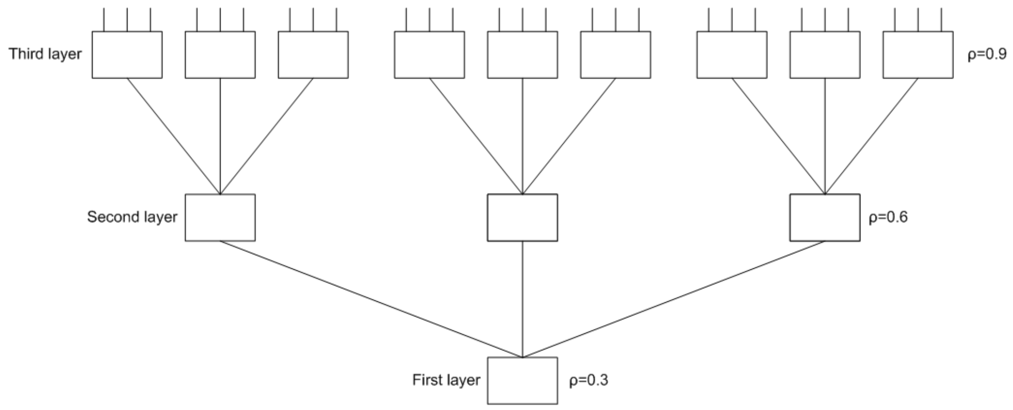


Figure 2. A hierarchy of ART units. The input pattern is registered at the bottom and is sequentially fed only to those ART units in the hierarchy of the "winning" F_2 units from the parent node. (Figure adapted from [37]).

3. GENERAL PURPOSE GRAPHICS PROCESSING UNIT WITH CUDA

The desire to display a 3D world on computers in realtime greatly increased the computational ability of graphics processors. Fig. 3 illustrates the design difference between CPUs and GPUs [41]. A kernel, which is the set of operations defined in GPU processors can be programmed and executed simultaneously in different threads. A single NVIDIA Fermi GPU theoretically is capable of containing up to 67,107,840 threads.

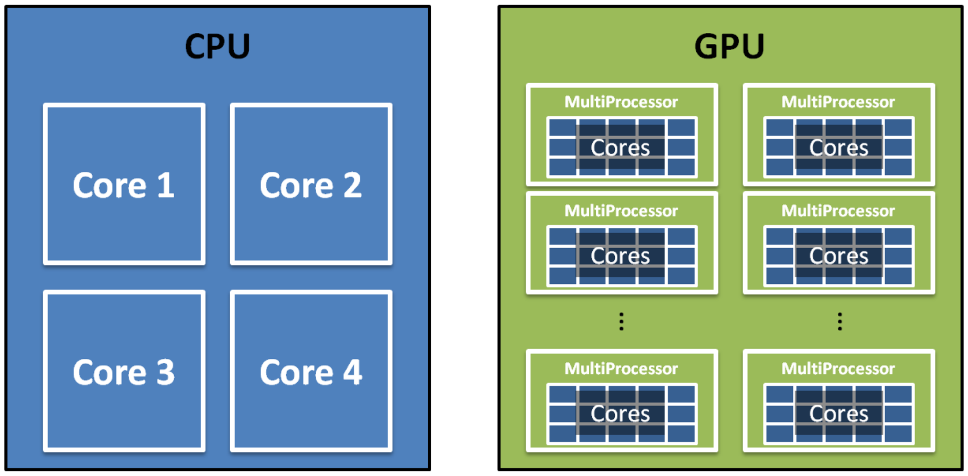


Figure 3. GPU and CPU architecture comparison. GPUs devote more transistors to data processing than CPUs.

But several constraints in GPGPU exist. Direct memory access between the host(CPU) and the device(GPU) is not possible. To handle certain data in other sides, data transfer is required either from the CPU to the GPU, or vice versa. Because the transfer rate is relatively slow, minimizing data transition is the critical concern. The lack of a dynamic pointer and array generation inside the kernel limits the GPU as well.

4. PARALLEL HIERARCHICAL FUZZY ART IN CUDA

To achieve the parallelization of the Parallel Hierarchical Fuzzy ART (PHF-ART), the layers, as shown in Fig. 1, were distributed among the GPU threads. Each layer is not an individual module but behaves as a controller to call up the required FA on each diverse state. Layer 1 is exclusively assigned to the root FA node. Every time an input passes through a layer, the working FA module in the layer emanates the adapted category back to the layer. Then it assigns the child FA node and broadcasts the node ID and the input ID to the adjacent lower layer while receiving the new assignment from the upper layer, which can be regarded as pipelining. Algorithm 1 is the pseudocode of the kernel in the program.

Algorithm 1 Layer Behavior

```
if $L_i$ assignment exists then
    call FA module
    call input
    do FA training
    set $L_{i+1}$:FA$_J$,input
end if
if layer is root then
    idData++
else
    wait assignment
end if
```

Defining the tree structure and achieving parallelization in the CUDA platform are also critical problems. After the initialization step, the first data will be registered in root FA. Once the training is completed, the layer will attempt to find the ID of the corresponding child's FA module, which is not set yet. In generic CPU programming, a child node can be generated easily by allocating a new pointer and cross referring between the parent and child node, or by employing vector template coding. As these methods are impossible in the kernel, a semi-dynamic pointer method is applied. Compared with dynamic arrays, semi-dynamic arrays have a fixed maximum size, and a tracking variable is defined to record the used amount.

The memory of the graphic card used for the experiment is 1.6 GB. The contents occupying the VRAM within the program are the data sample vectors, layer states and other very small entries such as the kernel itself and the registers and local variables in each kernel. A million samples of a 4 dimensional float vector take up only 32 MB, implying that the rest of the memory can be dedicated to the FA modules. The number of maximum FA modules depends on the dimension of the sample vector as well as the preset number of maximum categories allowed. Typically in the experiment, 1.5 million FA modules could be pre-declared.

Even though a semi dynamic array is applied, a parallel feature known as race condition [42] hinders the tracking of the maximum size. Assuming a certain situation in which all of the layers must generate a new child FA module, the threads will attempt to assign a child node in the same place because they are running in parallel. Thus, concurrent or sequential coding is required in order to correctly assign a child node and to keep the tracker in control. To reduce the non-parallelism, the throughput of the child ID finder, which runs right after the FA trainer, is limited as much as possible. The pseudocode is described in Algorithm 2. Once the child node ID is set up, the layer behavior kernel reruns to finish the task. The entire procedure using the child ID finder is depicted in Algorithm 3.

Algorithm 2 Child ID Finder

```
for $i = \forall$layer do
    if new child needed then
        idChild$\leftarrow$tracker
        tracker++
    end if
end for
```

Algorithm 3 Parallel Hierarchical Fuzzy ART

```
init setting
memcpy(host$\rightarrow$device)
for $i = 1$ to $nDATA + nLayer - 1$ do
    FA_Trainer()
    childIDFinder()
    setNextAssignment()
end for
memcpy(device$\rightarrow$host)
```

5. EXPERIMENTAL RESULTS

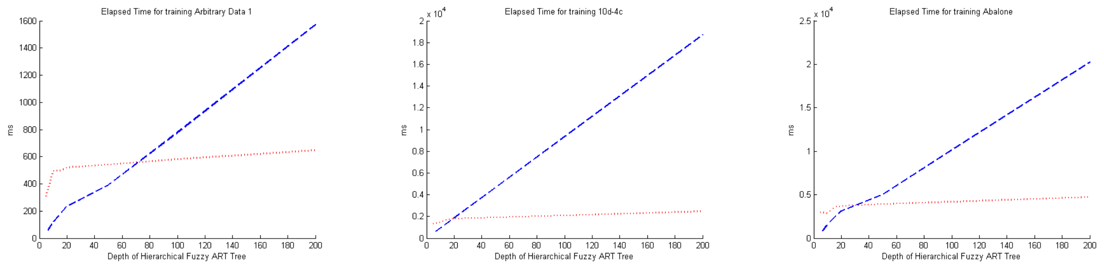
The experiments on both CPUs and GPUs were conducted on an Intel Xeon E5620 Quad Core CPU with 12 GB RAM and NVIDIA Geforce GTX 480. Two sets of arbitrarily generated data, “abalone" data from the UCI Machine Learning Repository [43] and 5 sets of the synthetic data developed by Handl and Knowles [44] are used for the performance testing. The depths of the hierarchy were set in the range of 5, 10, 15, 20, 50, 100 and 200. For the simulation, only the vigilances of each layer varied linearly in the range of [0.3, 0.9]. The learning rate and the choice parameter were set as 0.8 and 0.1, respectively.

The elapsed times on the CPU platform and the GPU platform were measured differently. The initial setup time for both platforms was excluded, but the time consumed copying data to and from the GPU was included on the GPU's performance aspect. The features of the data used for the simulation are summarized in Table 1.

Table 1. Description of the used data

Data Set	Attributes	Number of Data Points
Arbitrary 1	2	800
Arbitrary 2	2	40000
2d-10c	2	3630
2d-40c	2	2563
10d-4c	10	1482
10d-10c	10	3788
10d-40c	10	2707
Abalone	8	4177

Fig. 4 plots the elapsed time measured on each platform. When the tree depth is low, the CPU running speed is faster because the algorithm was based on layer pipelining. But as the depth grows to meet a certain value, the performance of the GPU version exceeds the CPU application. The point at which the GPU exceeds the CPU varies on each data set, as shown in Table 2. The time comparison chart implies that the larger the dimension of the data, the sooner the GPU surpasses the CPU. The maximum speed was boosted by 1170% on 2d-10c data with 200 layers. The average performance improvement is 859.37%, 527.95%, 294.74% and 140.46% on 200, 100, 50 and 20 layers, respectively.



(a) Arbitrary Data 1 (b) 10d-4c (c) Abalone

Figure 4. The elapsed time as a function of depth of a hierarchical fuzzy ART tree. The dotted line is the result acquired from the CPU while the dashed line is that from the GPU.

Table 2. Elapsed time (*ms*) comparison

Data Set	HF-ART Depth						
	5	10	15	20	50	100	200
Arbitrary1(GPU)	312	495	498	521	541	581	647
Arbitrary1(CPU)	40	119	174	234	392	778	1572
Arbitrary2(GPU)	4245	4788	6164	6503	7206	8286	10500
Arbitrary2(GPU)	1895	5879	8792	11672	20227	39093	76597
2d-10c(GPU)	968	628	752	783	853	962	1198
2d-10c(CPU)	349	694	1044	1392	3548	6954	14025
2d-40c(GPU)	478	508	597	617	669	751	927
2d-40c(CPU)	246	493	738	987	2463	4964	9907
10d-4c(GPU)	1342	1441	1750	1807	1924	2097	2462
10d-4c(CPU)	458	921	1379	1850	4647	9340	18719
10d-10c(GPU)	2807	3059	3866	4010	4259	4688	5460
10d-10c(CPU)	1186	2354	3539	4735	11925	23926	43974
10d-40c(GPU)	1980	2213	2784	2891	3038	3323	3834
10d-40c(CPU)	836	1681	2526	3343	8406	16863	31027
Abalone(GPU)	2972	2867	3582	3710	3929	4185	4715
Abalone(CPU)	519	1586	2370	3153	5018	10165	20214

6. CONCLUSIONS AND FUTURE STUDIES

Fig. 5 illustrates how finely the samples can be fragmented. The results also show that such deep clustering can be accomplished faster with GPU-based clustering than CPU-based algorithms.

Even though HF-ART on a GPU provides a noticeable speed improvement, a few obstacles remain. The limited size and the inflexibility of the graphics memory bind the total size of FA modules that can be generated. Furthermore high-dimensional data strains the distributed memory limits of the GPU, necessitating the investigation of hybridizing this approach with data reduction, such as principal component analysis, in preprocessing.

To the best of our knowledge, this is the first report of hierarchical ART clustering in GPU processors. Unlike previous research focusing on single ART unit parallelization on GPU platforms [34], this research enables multiple ART units in a tree structure to be trained simultaneously. We expect this contribution to impact applications where the need for hierarchical clustering is combined with high data loads and computational demands, such as in data mining and bioinformatics.

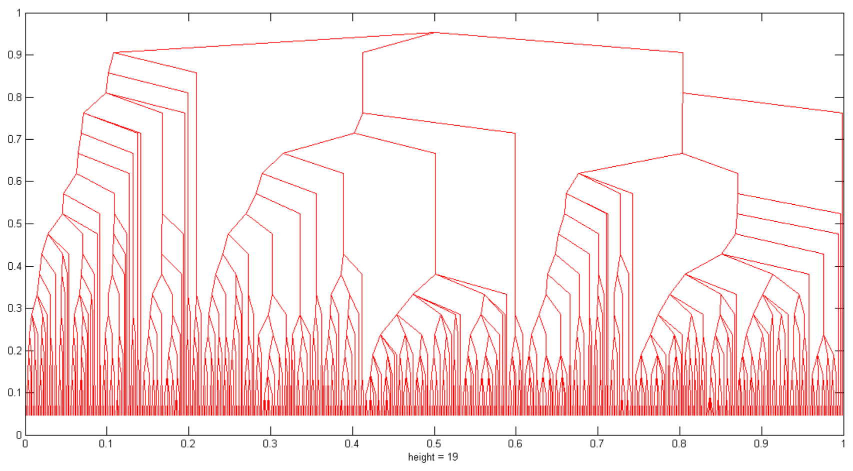


Figure 5. Generated fuzzy ART module tree throughout the training.

II. HIERARCHICAL BARTMAP : A NOVEL INNOVATION IN BICLUSTERING ALGORITHMS

Sejun Kim and Donald C. Wunsch II

ABSTRACT

Biclustering, a form of co-clustering or subspace clustering, has been demonstrated to be a more powerful method than conventional clustering algorithms for analyzing high-dimensional data, such as gene microarray samples. It involves finding a partition of the vectors and a subset of the dimensions such that the correlations among the biclusters are determined and automatically associated. Thus, it can be considered an unsupervised version of heteroassociative learning. Biclustering ARTMAP (BARTMAP) is a recently introduced algorithm that enables high-quality clustering by modifying the ARTMAP structure, and it outperforms previous clustering and biclustering approaches. Hierarchical BARTMAP (HBARTMAP), introduced here, offers a biclustering solution to problems in which the degrees of each attribute vary in association with different samples. We performed experiments on various synthetic and real data sets with other well known methods, including various clustering algorithms. Experimental results on multiple genetic datasets reveal that HBARTMAP can offer in-depth interpretation of microarrays, which other conventional biclustering or clustering algorithms do not achieve. Biclustering can be viewed as a data reduction technique, and its hierarchical version increases its capability of doing so. Thus, this paper contributes an hierarchical extension of biclustering algorithm, BARTMAP and comparatively analyzes their performance in the context of synthetic clustering and microarray data.

1. INTRODUCTION

Clustering is a common data-mining technique used to obtain information from raw data. However, major challenges arise when large numbers of samples must be analyzed, and these challenges escalate as the rate of data acquisition continues to increase, especially regarding the ability to gather high-dimensional data [45], such as gene expression microarray data. The curse of dimensionality renders the conventional clustering of high-dimensional data infeasible [7, 46, 47, 48]. The two critical traits of bioinformatics data are noise and high dimensionality, both of which diminish the robustness of clustering results [49]. Thus, biclustering was introduced to overcome computational obstacles and provide higher quality analyses [18, 50, 51, 52, 53, 54, 55]. This approach finds subsets of samples correlated to subsets of attributes. Due to the simultaneous row and column decomposition of the data matrix, biclustering, unlike clustering, can generate various correlated segments within a matrix.

The amount of biological data being produced is increasing at a significant rate [56, 57, 58]. For instance, since the publication of the H. influenzae genome [59], complete sequences for over 40 organisms have been released, ranging from 450 genes to over 150,000 genes. This is one of many examples of the enormous quantity and variety of information being generated in gene expression research. The surge in data has resulted in the indispensability of computers in biological research. Other data sets, such as earth science data and stock market measures, are also collected at a rapid rate [60, 61]. The discovery of biclusters has allowed sets with coherent values to be searched across a subset of transactions or examples. An important example of the utility of biclustering is the discovery of transcription modules from microarray data, which denote groups of genes that show coherent activity only across a subset of all conditions constituting the data set, and may reveal important information about the regulatory mechanisms operating in a cell [62].

Neural networks have played a major role in data mining and clustering [63, 64, 65, 66]. Adaptive Resonance Theory (ART) [10] is a neural network-based clustering algorithm and ARTMAP [13] is a neural network for supervised learning composed of two ART modules and an inter-ART module. This, particularly the inter-ART mechanism, was revised to develop Biclustering ARTMAP (BARTMAP) [21]. Biclustering through BARTMAP is achieved by performing row-wise and column-wise Fuzzy ART clustering with the intervention of correlation calculations. ART has advantages of speed, low memory utilization, ease of implementation and analysis, solution of stability-plasticity dilemma, and no need to determine the number of clusters in advance [32]. These strengths, particularly the latter, also apply to BARTMAP, as the number of biclusters is adjusted automatically.

This paper introduces Hierarchical BARTMAP (HBARTMAP), which inherits the advantages of BARTMAP. HBARTMAP also automatically generates a BARTMAP tree with attention given to each cluster obtained on every node, starting from the root BARTMAP node. After generating the tree, this technique uses a correlation comparison method to recursively calculate the measurement of row and column clusters from every terminal node, eventually creating a full hierarchical bicluster classification.

The remainder of the paper is organized as follows. Section 2 introduces the brief summary of Fuzzy ART and ARTMAP, followed by details of BARTMAP. In Section 3, the main topic, HBARTMAP approach is presented. In Section 4, the experimental setup, data description, comparison methods are explained and the results are shared. Finally, the conclusion is provided in Section 5. Fuzzy adaptive resonance theory and ARTMAP, which are the base methods of HBARTMAP are described in detail in the Appendix.

2. BACKGROUND

2.1. FUZZY ADAPTIVE RESONANCE THEORY AND ARTMAP

Adaptive Resonance Theory (ART) is a neural network-based unsupervised learning method developed by Carpenter and Grossberg [67] inspired by the autonomous and cognitive brain function. One of the most important problems in clustering is the stability-plasticity dilemma [68] and ART provides the solution by proposing how top-down expectations focus attention on salient combinations of cues, and characterizes how attention may operate via a form of autonomous normalizing competition. For other relevant extensions of ART, see [12, 14, 69, 70, 71, 72]. [12] and [14] are particularly relevant to this paper and are summarized in Appendices A and B.

Most details are provided in these Appendices and especially in the original references, but a few details are reviewed here for ease of reading the rest of the paper. The base structure of ART/FA is presented in Fig. 1. The vigilance parameter ρ determines whether a newly introduced pattern fits into existing neurons. Once a neuron passes the test, the algorithm will update the weights of the winning neuron. However, if all neuron fail to meet the criteria, an uncommitted new neuron will be automatically created and updated correspondingly. More detail about FA is presented in Appendix A.

ARTMAP [14] is a variant of ART, which learns to associate arbitrary sequences of input and output pattern pairs. It is achieved by incorporating two ART modules, which receive input patterns (ART_a) and corresponding labels (ART_b), respectively, with an inter-ART module, hence making it a supervised learning algorithm. The method is capable of fast, online, incremental learning, classification and prediction. The base architecture

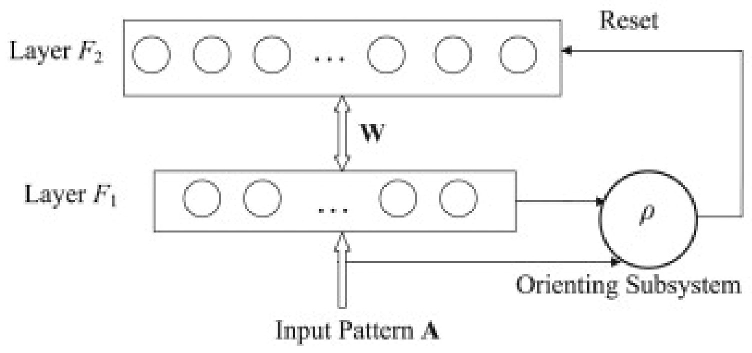


Figure 1. ART architecture. Two layers are included in the attentional subsystem, connected via bottum-up and top-down adaptive weights. The interaction between the neuron layers are controlled by a vigilance parameter ρ.

of ARTMAP has inspired the development of Biclustering ARTMAP. A description of ARTMAP is provided in Appendix B.

2.2. BICLUSTERING ARTMAP (BARTMAP)

The BARTMAP architecture is derived from Fuzzy ARTMAP, which also consists of two Fuzzy ART modules communicating through the inter-ART module, as shown in Fig. 2. However, the inputs to the ART_b module are attributes (rows) instead of labels. The inputs to the ART_a module are samples (columns), although the inputs to the modules can be exchanged, without otherwise affecting any properties of the algorithm. The objective of BARTMAP is to combine the clustering results of the attributes and samples of the data matrix from each ART_a and ART_b module to create biclusters that project the correlations of attributes and samples.

The first step of BARTMAP is to create a set of K_g gene clusters G_i, $i = 1, \cdots, K_g$, for N genes by using the ART_b module, which behaves like standard Fuzzy ART. The goal of the following step is to create K_s sample clusters S_j, $j = 1, \cdots, K_s$, for M samples

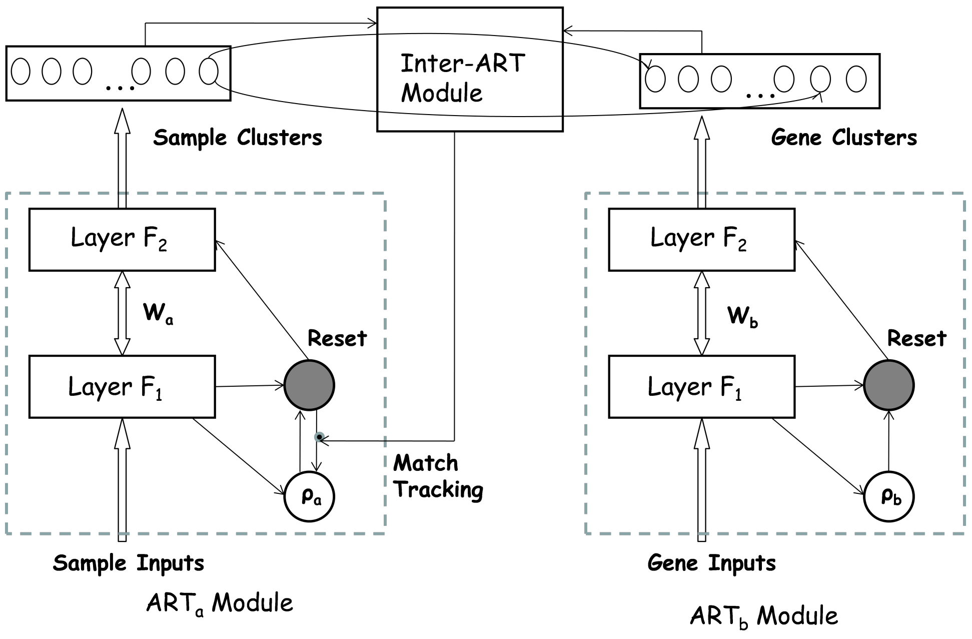


Figure 2. Structure of BARTMAP. Gene clusters first form in the ART_b module, and sample clusters form in the ART_a module with the requirement that members of the same cluster behave similarly across at least one of the formed gene clusters. The match tracking mechanism will increase the vigilance parameter of the ART_a module if this condition is not met.

within the ART_a module while calculating the correlations between the attribute and sample clusters. When a new data sample is registered to the ART_a module, the candidate sample cluster that is eligible to represent this sample is determined based on the winner-take-all rule using the standard Fuzzy ART vigilance test. If this candidate cluster corresponds to an uncommitted neuron, learning will occur to create a new one-element sample cluster that represents this sample, as in Fuzzy ART. Before updating the weights of the winning neuron, it will check whether the following condition is satisfied: A sample is absorbed into an existing sample cluster if and only if it displays behavior or patterns similar to the other members in the cluster across at least one gene cluster formed in the ART_b module.

The similarity between the new sample s_k and the sample cluster $S_j = \{s_{j1}, \cdots, s_{jM_j}\}$ with M_j samples across a gene cluster $G_i = \{g_{i1}, \cdots, g_{iN_i}\}$ with N_i genes is calculated as the average Pearson correlation coefficient between the sample and all the samples in the cluster,

$$r_{kj} = \frac{1}{M_j} \sum_{l=1}^{M_j} r_{k,jl}, \tag{1}$$

where

$$r_{k,jl} = \frac{\sum_{t=1}^{N_i} (e_{s_k g_{it}} - \overline{e}_{s_k G_i})(e_{s_{jl} g_{it}} - \overline{e}_{s_{jl} G_i})}{\sqrt{\sum_{t=1}^{N_i} (e_{s_k g_{it}} - \overline{e}_{s_k G_i})^2} \sqrt{\sum_{t=1}^{N_i} (e_{s_{jl} g_{it}} - \overline{e}_{s_{jl} G_i})^2}}, \tag{2}$$

and

$$\overline{e}_{s_k G_i} = \frac{1}{N_i} \sum_{t=1}^{N_i} e_{S_k g_{it}}, \tag{3}$$

$$\overline{e}_{s_{jl} G_i} = \frac{1}{N_i} \sum_{t=1}^{N_i} e_{S_{jl} g_{it}}. \tag{4}$$

The sample s_k is enclosed in cluster S_j only when r_{kj} is above some threshold η; learning will occur following the Fuzzy ART updating rule.

If the sample does not show any behaviors similar to those of the sample cluster that the winning neuron represents for any clusters of genes, the match tracking mechanism will increase the ART_a vigilance parameter ρ_a from its baseline vigilance to just above the current match value to disable the current winning neuron in ART_a. This shut-off will force the sample to be included into some other cluster or will create a new cluster for the sample if no existing sample cluster matches it well.

3. HIERARCHICAL BARTMAP

The basic idea of Hierarchical BARTMAP (HBARTMAP) is to reiterate BARTMAP within the obtained BARTMAP results in order to obtain sub-biclusters, as shown in Fig. 3. Such subdivision provides insight into reinterpreting the generated biclusters by combining or disbanding sub-biclusters of the initial results. The overall procedure is presented in Alg. 1.

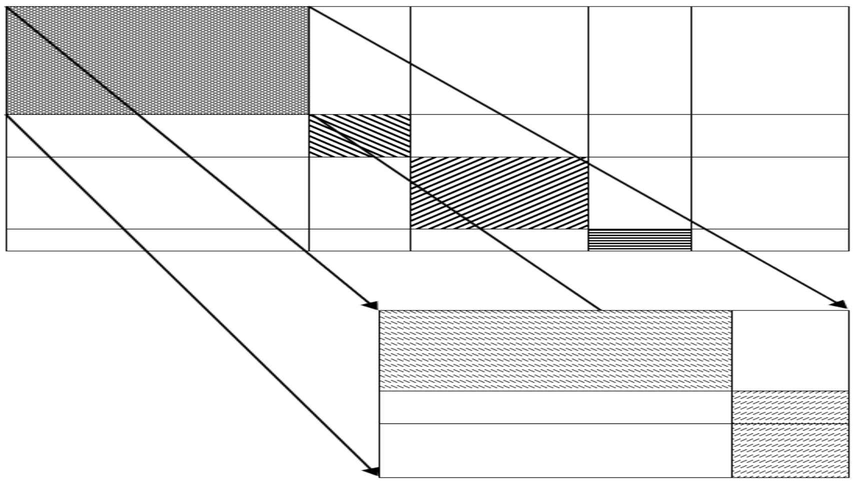


Figure 3. Main idea of hierarchical biclustering. Within a subset, the biclustering procedure is repeated to discover finer detail. In HBARTMAP, increasing the vigilances of the ART_a and ART_b modules as well as the correlation threshold by a preset interval enables diversification. The result can be visualized as a block diagonal matrix, where the blocks themselves can also be visualized as block diagonal matrices. The most salient data is closest to the diagonal. so the technique can be considered a data reduction method.

The first step of HBARTMAP is performing BARTMAP on the data. Data is preprocessed by rescaling to the range of $[0,1]$ using the formula:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{5}$$

Algorithm 1 Pseudo Code of Overall HBARTMAP Algorithm

```
Initialize BARTMAP
Load data $\mathbf{X_{raw}}$
$\mathbf{X} \leftarrow preprocess(\mathbf{X_{raw}})$
$X_n, X_d \leftarrow shape(\mathbf{X})$
Run BARTMAP($\mathbf{X}$, $\rho_a$, $\rho_b$, $\eta$)
{Bicluster Selection}
for $k = 1$ to $N_{SampleCluster}$ do
    for $l = 1$ to $N_{AttrCluster}$ do
        $\mathbf{B} \leftarrow Bicluster[k, l]$
        if $f(\mathbf{B}) \leq \xi$ then
            $\mathbf{Bic}[k] \leftarrow append(\mathbf{Bic}[k], B)$
        end if
    end for
end for
{Recursive ChildBARTMAP}
for $i = 1$ to $N_{SampleCluster}$ do
    Run ChildBARTMAP($\mathbf{Bic}[i]$, $X_n$, $X_d$, $\rho_a$, $\rho_b$, $\eta$)
end for
```

After the data preprocessing is completed, the BARTMAP module goes through the entire set with preset parameters. Typically, the preset parameters of the root module are set with low vigilance values so that the initial biclustering result includes a relatively small number of biclusters with a large number of members in each bicluster. The vigilance ρ_a and ρ_b for each ART_a andART_b unit are both set as 0.1. The correlation threshold η, the main factor of BARTMAP which decides to include or exclude a newly introduced sample into an existing cluster, is also set as 0.1. These settings allow the root module to generate large size clusters.

Unlike BARTMAP, which lacks the ability to select and form biclusters, the bicluster selection step is initiated after each BARTMAP module finishes the biclustering to evaluate and pair the attribute and sample biclusters, as defined by $f(B) \leq \xi$, where ξ and $f(B)$ are the bicluster matching threshold and the correlation fitness function, respectively.

The correlation coefficient between two variables p and q measures the grade of linear dependency between them. Given a bicluster $\mathbf{B}$ composed of N samples and M attributes, $B = [g_1, \cdots, g_N]$, the average correlation of $\mathbf{B}$, $\delta(\mathbf{B})$, is defined as

$$\delta(\mathbf{B}) = \frac{1}{\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \delta(g_i, g_j) \tag{6}$$

where $\delta(g_i, g_j)$ is the correlation coefficient between samples i and j.

With the calculated average correlation of bicluster B, the correlation fitness function is applied, which is defined by

$$f(\mathbf{B}) = (1 - \delta(\mathbf{B})) + \sigma_\delta + \frac{1}{N} + \frac{1}{M}, \tag{7}$$

where σ_δ is the standard deviation of the values $\delta(g_i, g_j)$. The standard deviation is included in order to avoid the value of the average correlation being high. The best biclusters are those with the lowest fitness function values.

During the bicluster evaluation process, once the fitness of every bicluster of a sample group is calculated, the most highly correlated attributes begin to be sorted out in accordance with a preset threshold. If the fitness of an attribute is smaller than the fitness threshold, it is selected. Once the attribute scan is complete, the process advances to the next sample group and progresses through the selection step again. However, to avoid previously-selected attributes overlapping in different sample groups, they are excluded from the search.

The next step is to register the acquired biclusters to child BARTMAP modules. Once each child node receive the pass information, the vigilance and correlation thresholds are first adjusted by

$$\rho_{a,i} = \rho_a \cdot (1 + \gamma\sqrt{M_i}), \tag{8}$$

$$\rho_{b,i} = \rho_b \cdot (1 + \gamma\sqrt{N_i}), \tag{9}$$

$$\eta_i = \eta \cdot (1 + \gamma\delta(B_i)), \tag{10}$$

where M_i and N_i are the number of attributes and samples included in the ith bicluster $\mathbf{B}_i$, respectively, $\rho_{a,i}$, $\rho_{b,i}$ and η_i are the vigilances of ART_a, ART_b and correlation threshold for the ith child BARTMAP node and $\gamma = 1/(X_n \cdot X_d)$ being the recursive node control factor, where X_n and X_d are the number of samples and attributes of the data set $\mathbf{X}$ processed through the parent BARTMAP node.

Fig. 4 depicts how the HBARTMAP tree is formed. The child node generation is completely recursive, thus the information available for each child node are the parameters and data sets processed through its parent node. The pseudo-code of ChildBARTMAP function shown in Alg. 2 is a recursive function used to generate a tree of BARTMAP modules that solely compute the subset.

The final phase of the algorithm is to decide which layer of the bicluster tree provides the most meaningful result. For many clustering problems, the external criteria is unknown, so we applied the Caliński and Harabasz index [73], which is defined as,

$$CH(K) = \frac{Tr(\mathbf{S}_B)}{K-1} / \frac{Tr(\mathbf{S}_W)}{N-K}, \tag{11}$$

where N is the number of objects, K is the number of clusters / biclusters and $Tr(\mathbf{S}_B)$ and $Tr(\mathbf{S}_W)$ are the traces of the between and within-class scatter matrix, respectively. The

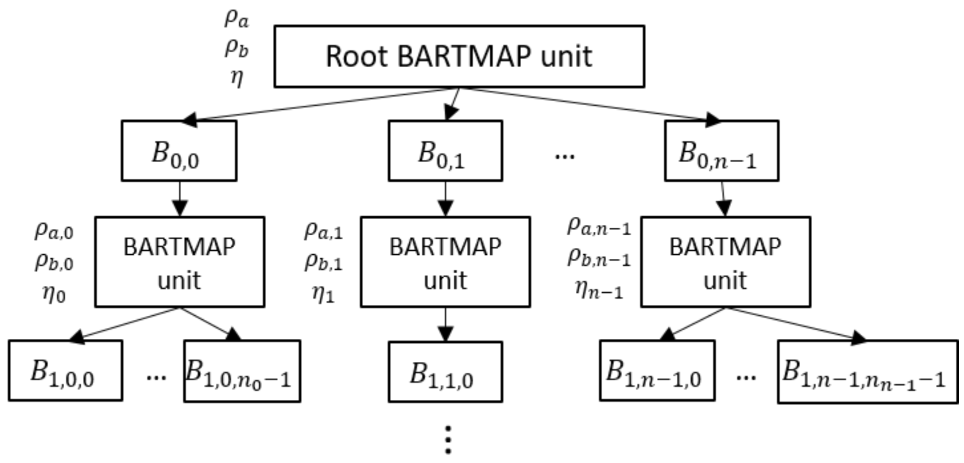


Figure 4. Tree formation of HBARTMAP. Each discovered biclusters from the parent BARTMAP node are processed to corresponding child node with the parameters adjusted by Eq. 8, 9 and 10.

traces of the matrices are defined as,

$$Tr(\mathbf{S}_B) = \sum_{i=1}^{K} N_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T, \tag{12}$$

$$Tr(\mathbf{S}_W) = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij}(\mathbf{x}_j - \mathbf{m}_i)(\mathbf{x}_j - \mathbf{m}_i)^T, \tag{13}$$

where

$$\gamma_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \in \text{cluster } i \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

with $\sum_{i=1}^{K} \gamma_{ij} = 1 \forall j$, where $\mathbf{x}_j, j \in 1, ..., N$ represents each sample.

Algorithm 2 Pseudo Code of ChildBARTMAP Function

```
$M, N \leftarrow shape(\mathbf{X})$
$\rho_a$, $\rho_b$, $\eta$ = AdjustParameters($X_{parent,n}$, $X_{parent,d}$, $\rho_{parent,a}$, $\rho_{parent,b}$, $\eta_{parent}$)
if $\rho_a$ ==1 or $\rho_b$ ==1 or $\eta$==1 then
    Return
end if
Run BARTMAP($\mathbf{X}$, $\rho_a$, $\rho_b$, $\eta$)
if $N_{SampleCluster}$ or $N_{AttrCluster} \leq 2$ then
    return
end if
for $k = 1$ to $N_{SampleCluster}$ do
    for $l = 1$ to $N_{AttrCluster}$ do
        $\mathbf{B} \leftarrow Bicluster[k, l]$
        if $f(\mathbf{B}) \leq \xi$ then
            $\mathbf{Bic}[k] \leftarrow append(\mathbf{Bic}[k], B)$
        end if
    end for
end for
{Recursive ChildBARTMAP}
for $i = 1$ to $Num\_SampleCluster$ do
    Run ChildBARTMAP($\mathbf{Bic}[i]$, $M$, $N$, $\rho_a$, $\rho_b$, $\eta$)
end for
return
```

The objective of applying the index is to find the best layer that maximizes $CH(K)$ as K increases and differs per layer. As the layer that gives the maximum Caliński and Harabasz index doesn't always necessarily translate into the best result, the five highest results are given for analysis.

4. EXPERIMENTAL RESULTS

4.1. DATA SETS

Various sizes of the synthetic data set developed by Handl and Knowles [41] were used for the simulation to compare the clustering performance. The features of the applied data set are given in Table 1.

Table 1. Characteristics of synthetic data set

No.	Samples	Attributes	Clusters
1	1286	100	4
2	2117	860	10
3	9841	1377	27

For real world data experiments, the leukemia data set [72] was applied. The set consists of 72 samples, including bone marrow samples, peripheral blood samples and childhood AML cases. Twenty-five of these samples are acute myeloid leukemia (AML), and 47 are acute lymphoblastic leukemia (ALL), each of which is composed of two subcategories due to the influences of T-cells and B-cells. The expression levels for 7,129 genes were measured across all of the samples by high-density oligonucleotide microarrays.

The second experiment was done with the yeast cell cycle data set [73]. It demonstrates the oscillation of expressions of 2,884 genes and 17 conditions, which were selected according to [74]. The value entries were transformed by scaling and logarithm $x \to 100\log(10^5 x)$ so that the matrix was composed of integers in $[0,600]$ range.

The SRBCT data set [75] presents diagnostic research on the Small, Round, Blue-Cell Tumors of childhood cancers. It consists of 83 samples from four categories, known as Burkitt lymphomas, the Ewing family of tumors, neuroblastoma and rhabdomyasarcoma.

Gene expression levels of 6,567 genes were measured during cDNA microarrays for each samples, 2,308 of which pass the filter that requires the red intensity of a gene to be greater than 20 and were kept for further analysis. A logarithm was taken to linearize the relations between different genes and to lower very high expression.

In the experiments, the choice parameters α for both ART modules are set to 0.001 as they do not affect the clustering performance. The learning rates β and γ for both ART modules are set to 1 to encourage fast learning. The baseline vigilance ρ and correlation threshold η are both set to 0.1 in the root node. The bicluster matching threshold ξ is set to 0.6.

4.2. COMPARISON METHODS

We compared the performance of HBARTMAP with various clustering / biclustering algorithms. Fuzzy ART (FA), Biclustering ARTMAP (BARTMAP) were chosen, both of which are mentioned in Section II. K-means and agglomerative hierarchical clustering with three different linkage modes were also implemented for comparison.

The statistical algorithmic method for bicluster analysis (SAMBA) algorithm [78] shifts the problem domain from finding sub-matrices with coherent behavior to probabilistic modeling and graph theory. The data set is represented as a bipartite graph, where the vertices correspond to genes and deletions. The problem of identifying biclusters is then transformed into trying to find heavy sub-graphs in this graph representation. Then the algorithm attempts to identify the heaviest bicliques by applying a hashing technique. The final phase of SAMBA is attempting to make local improvements by adding or deleting vertices until no further improvement is possible.

In interrelated two-way clustering (ITWC), the gene cluster and sample cluster are viewed simultaneously by dynamically using the relation between the groups of genes and samples while iteratively clustering both [22]. As this method works on the principle of finding the most important genes, the clustering of samples can be far better than clustering without using information on the clusters of samples. The steps of the algorithm are done by first performing clustering among genes with K-means or self-organizing map. Then the sample clustering is performed with K-means. Both results are combined to determine heterogeneous groups based on correlation and finally during the cleanup phase, the least important genes are dropped, until the given termination condition.

Co-regulated Biclustering (CoBi) is a relatively fast biclustering algorithm achieved by using a BiClust tree based technique [79]. The process consists of pruning and expanding the generated cluster decision tree. In the cluster expansion phase, clusters are merged by an intersection operation between two clusters. The method is shown in Algorithm 3.

Algorithm 3 Pseudo Code of CoBi [79]

```
  Construct initial BiClust tree $BT$
  Prune cluster $C_i$ from $BT$, if $|C_i| < MinGene$
  $BiClust = ExpandCluster(BT, MinGene, \theta)$
  $BiClust = RemoveSubCluster(BiClust)$
```

In the *ExpandCluster* function, two subtrees from the BiClust tree are merged and pruned if the number of genes is lesser than the threshold *MinGene*. After the iterative process, *RemoveSubCluster* function cleans out redundant clusters, where genes in the clusters are same.

The parameters used on the methods for comparison are presented in Table 2. For ranged parameters, the best performance from evaluation metrics are reported.

Table 2. Parameter setup for clustering and biclustering methods

Method	Parameters (best performance reported)
HBARTMAP	$\rho_a = 0.1, \rho_b = 0.1, \eta = 0.3$
KM	$K = 2 \sim 50$
FA	$\rho = 0.3 \sim 0.7$
HC	Single, Complete, Average linkage
BARTMAP	$\rho_a = 0.3, \rho_b = 0.3 \sim 0.7, \eta = 0.4$
CoBi	$MinGene = 3 \sim 5$ $\theta = 50\%$, $\tau = 20$
ITWC	$K = 2 \sim 50$
SAMBA	$k = 20, L = 20 \sim 40, D = 40, N_1 = 2, N_2 = 5 \sim 10$

4.3. EVALUATION

The results of HBARTMAP and comparison methods are evaluated by several validation methods. See chapter 10 of [7] for an overview of various cluster evaluation methods. To compare the resulting clusters with the real structures in terms of external criteria, the Rand index and the adjusted Rand index [78] were applied.

We assume that P is a pre-specified partition of dataset X with N data objects, which also is independent from a clustering structure C resulting from the use of the algorithm being evaluated. Therefore, a pair of data objects $\mathbf{x}_i$ and $\mathbf{x}_j$, will yield four different cases based on how $\mathbf{x}_i$ and $\mathbf{x}_j$ are placed in C and P.

- Case 1 $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster of C and the same category of P.
- Case 2 $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster of C and different categories of P.
- Case 3 $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters of C and the same category of P.
- Case 4 $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters of C and different categories of P.

The Rand index [78] then can be defined as follows, with larger values indicating greater similarity between C and P:

$$R = \frac{(a+d)}{L}, \tag{15}$$

where a, b, c and d are the quantities of case 1 to 4 with all pairs of points, respectively and $L = a+b+c+d$. The adjusted Rand index [81] assumes that the model of randomness takes the form of the generalized hypergeometric distribution, which is written as,

$$Adj_R = \frac{\binom{N}{2}(a+d) - ((a+b)(a+c)+(c+d)(b+d))}{\binom{N}{2}^2 - ((a+b)(a+c)+(c+d)(b+d))}. \tag{16}$$

The adjusted Rand index has demonstrated consistently good performance in previous studies compared to other indices.

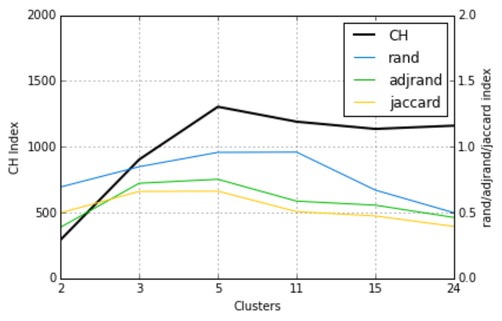
Jaccard coefficient is a common method in measuring the species diversity between two different clusters. It is similar to the Rand index, but it disregards the pairs of elements that are in different clusters. It is defined by,

$$J = a/(a+b+c). \tag{17}$$

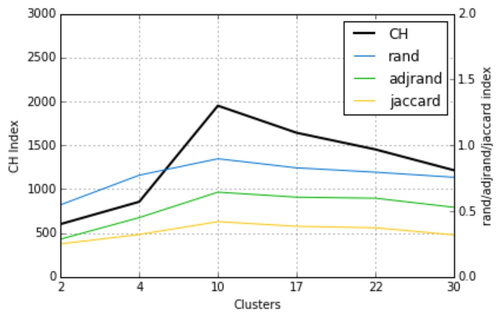
4.4. RESULTS

The relationship of the external and internal criteria obtained through HBARTMAP on the synthetic data sets are shown in Fig. 5. The results state that HBARTMAP correctly picks the optimal layer in the bicluster tree, as the peak of CH index matches each peak of all external criteria.

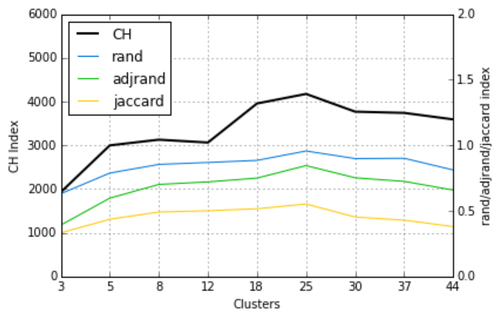
Fig. 6 shows the clustering result of a synthetic data set. On the root node, HBARTMAP divides the data set into two major clusters. Then, the algorithm performs BARTMAP within each cluster to split it into two subclusters.



(a) Synthetic Data Set 1

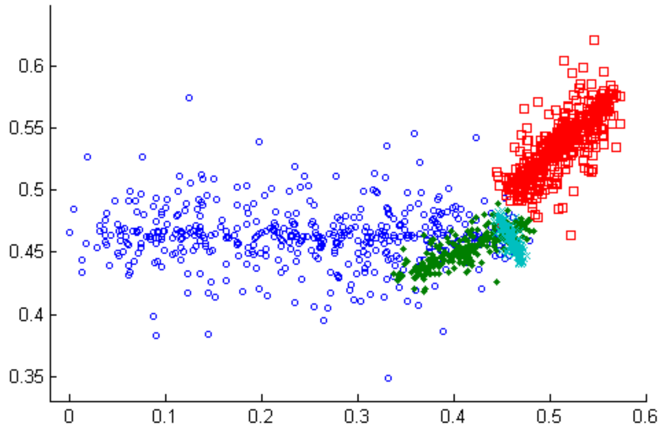


(b) Synthetic Data Set 2

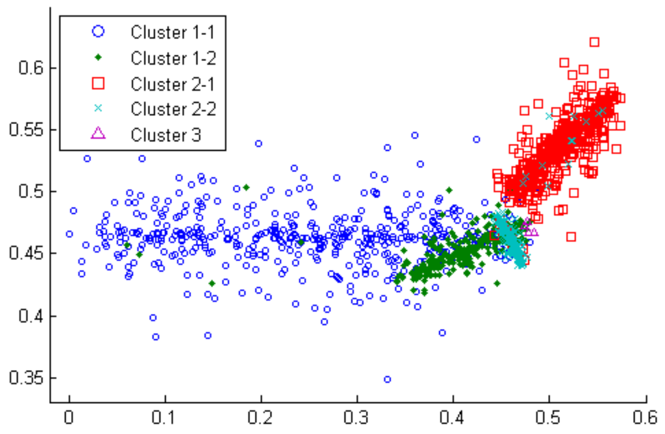


(c) Synthetic Data Set 3

Figure 5. Evaluation results of all synthetic data sets showing the relationship between the external (CH index) and internal (rand, adjrand, jaccard) criteria. Various validation indexes are presented per layer. The x-axis values are the numbers of clusters found in each layer.



(a) Tag of given synthetic data



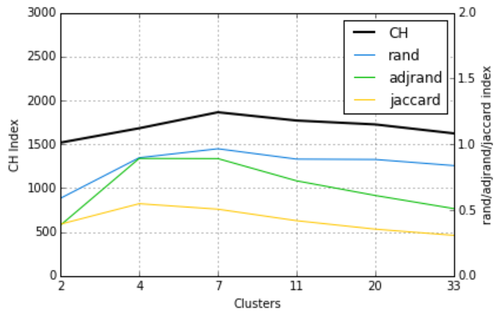
(b) HBARTMAP clustering result

Figure 6. The result of HBARTMAP running the synthetic data set 1 with 4 given clusters in a 2-dimensional plane. HBARTMAP results in 5 layers. The cluster index shows that HBARTMAP initially acquires 3 clusters, then two of each (cluster 1 and 2) splits into two clusters after the exploring into two more layers in the HBARTMAP bicluster tree, resulting in 5 biclusters.

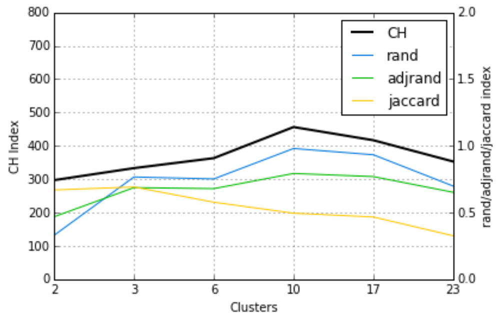
The results from real world data sets are presented in Fig. 7. While the CH index correctly discovers the layer where the external criteria is highest on the leukemia and the yeast data set, the results from SRBCT does not match.

The main focus of the real data set simulation is to perform biclustering with HBARTMAP and to judge how precisely the condition that this approach computes matches external criteria. Fig. 7 depicts the Rand index and the adjusted Rand index result of each layer, beginning with the biclustering result from the root HBARTMAP node. Because the initial vigilance and threshold parameters are set low, the result is rough. While the leukemia data set has three conditions, the root node only found two. As a result, deeper layers were evaluated, and the growth of both the Rand index and the adjusted Rand index is obvious. At layer 3, where 5 biclusters were discovered, the Rand index was 0.9711, and the adjusted Rand index was 0.8881, both of which values are higher than the second best result 0.7666 by CoBi.

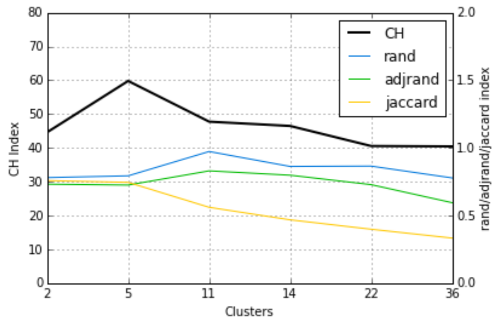
A comparison of the evaluation results among tested algorithms is shown in Table 3. The methods used for comparison are Rand index (R) and Adjusted Rand index (AR). HBARTMAP clearly performs better than the tested methodologies among all data sets studied.



(a) Leukemia



(b) Yeast



(c) SRBCT

Figure 7. Evaluation results of real world data sets (leukemia, yeast and SRBCT) showing the relationship between the external (CH index) and internal (rand, adjrand, jaccard) criteria. Various validation indexes are presented per layer. The x-axis values are the numbers of clusters found in each layer.

Table 3. Comparison on various clustering / biclustering methods. The abbreviations of the methods are as follows: K-means (KM), fuzzy ART (FA), biclustering ARTMAP (BAM), hierarchical clustering with complete linkage (HC-C), co-regulated biclustering (CoBi), interrelated two-way clustering (ITWC) and statistical algorithmic method for bicluster analysis (SAMBA). R and AR are abbreviations for Rand index and Adjusted rand index, respectively.

Methods		Data Sets					
		Synth 1	Synth 2	Synth 3	Leukemia	Yeast	SRBCT
HBARTMAP	R	0.9576	0.8948	0.9401	0.9427	0.9815	0.9652
	AR	0.7422	0.7437	0.7997	0.8881	0.7893	0.8247
KM	R	0.7323	0.7486	0.7945	0.8776	0.8583	0.7632
	AR	0.6123	0.5486	0.6921	0.5780	0.5943	0.2679
FA	R	0.8404	0.8222	0.7996	0.8709	0.8492	0.8792
	AR	0.7020	0.6977	0.7171	0.6874	0.6398	0.7146
BAM	R	0.9385	0.8292	0.9080	0.9109	0.9388	0.8939
	AR	0.7351	0.7077	0.7247	0.7573	0.7794	0.7485
HC-C	R	0.8566	0.8191	0.8277	0.8662	0.8145	0.7640
	AR	0.5958	0.6036	0.6172	0.5299	0.4936	0.2247
CoBi	R	0.8921	0.8118	0.8750	0.9001	0.8990	0.9101
	AR	0.6595	0.5978	0.6972	0.7666	0.7481	0.7592
ITWC	R	0.9105	0.8660	0.9219	0.7625	0.7709	0.7251
	AR	0.5389	0.6161	0.7453	0.6956	0.3882	0.4705
SAMBA	R	0.7928	0.8863	0.9158	0.9293	0.8420	0.8906
	AR	0.2774	0.7317	0.5594	0.2801	0.5716	0.7613

5. CONCLUSION

In this paper, we introduce hierarchical BARTMAP, an hierarchical approach for biclustering utilizing BARTMAP. The results indicate that the performance of HBARTMAP clearly exceeds these other clustering and biclustering methods. In particular, the experimental results demonstrate that HBARTMAP provides better biclustering than BARTMAP, which had previously shown the best published results we had found. The increase in the adjusted Rand index while searching each layer indicates that the hierarchical version of BARTMAP can be implemented effectively in high-dimensional data analysis. It suggests that utilizing the hierarchical approach on biclustering was the major factor of successful experiments. The superiority of HBARTMAP over BARTMAP, and BARTMAP over other approaches, is mostly consistent across a range of problems and of validation criteria.

III. VALUE GRADIENT LEARNING BASED SEMI-SUPERVISED SUPPORT VECTOR MACHINES

Sejun Kim and Donald C. Wunsch II

ABSTRACT

Semi-supervised support vector machines (S^3VM) are a modification of support vector machines (SVM) that allows labeled and unlabeled data to be used together for training. In this research, we further modified the S^3VM optimization by using adaptive dynamic programming (ADP) to achieve better classification. Value gradient learning (VGL), a relatively new and powerful ADP algorithm, was applied; it allows faster convergence by learning value gradients directly with lower cost. In order to apply S^3VM in an ADP structure, the state and action vectors were set as the label and hyperplane vectors, respectively. During the initial training process, which was supervised, ordinary SVM was used, and the acquired hyperplane coefficients were registered to the action network. As the semi-supervised learning stage began, the unlabeled test set was applied, and the hyperplane and label vectors of the entire data set were adjusted until the cost function converged to a preset threshold. The experiments demonstrated that Value Gradient Learning S^3VM ($VGLS^3VM$) can perform more accurate semi-supervised clustering compared with conventional S^3VM algorithms. In so doing, this paper demonstrates one of the few architectures to combine supervised, reinforcement, and unsupervised learning.

1. INTRODUCTION

Semi-supervised learning [24] combines the advantages of both supervised and unsupervised learning. Generally, it is easier and more cost efficient to tag only a small portion of the data set, which forms the labeled set, while the majority of the data set remains untagged, forming the unlabeled set. In semi-supervised learning, the supervised stage handles the labeled set and then analyzes the remaining data with the knowledge acquired from the previous step. Generative probabilistic models, semi-supervised support vector machines and graph-based semi-supervised learning are some widely used methods.

Support vector machines (SVM) [82] have been used for various types of classification and clustering, such as biological analysis, statistics and pattern recognition. Semi-supervised support vector machines (S^3VM) can process partially labeled data [83, 84]. The unlabeled set is handled using additional optimization points. The two broad optimization strategies are combinatorial optimization and continuous optimization, which adjust the decision boundary and the label vector, respectively [85, 86, 87, 88, 89].

Adaptive dynamic programming (ADP) is a tool typically used for solving optimization and optimal control problems in the presence of noise, uncertainty and non-stationarity. The main goal is to learn the cost-minimizing actions of an agent on each state. An action network is a neural network that generates an action for a given state. The critic function, another neural network in the ADP structure, is trained to approximate the long-term cost. Heuristic dual programming (HDP) and TD(λ) [90, 91] are value learning (VL) methods, while dual heuristic dynamic programming (DHP) and globalized DHP are value gradient learning (VGL) [92] methods. VL methods tend to be slow because they must cover a representative portion of the state space, while VGL methods only require a single trajectory, making their convergence faster. A thorough review of these terms is beyond the scope of this paper, but VGL is explained in Section 2.2, and the other terms are explained in [93].

This research focuses on solving S^{3}VM using VGL to perform the optimization motivated by ADP based methods on non-control problems [94, 95, 96]. The state of the ADP structure is the set of the label vectors. The action network generates the hyperplane vector, which is defined as the action in the system. The cost function is set as the S^{3}VM fitness function, so the S^{3}VM can be solved as an optimal control problem. The structure of the combined model, referred to as VGL based S^{3}VM (VGLS^{3}VM), appears in Fig. 1. The mathematical configuration and notations are further discussed in Section 3.

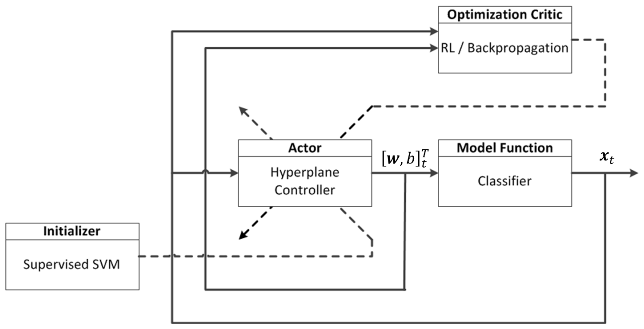


Figure 1. General structure of the VGLS^{3}VM system. The initializer sets up the actor neural network with the supervised SVM results. Afterwards, the adaptive critic design (ACD) is implemented for the S^{3}VM process by applying VGL.

This paper contains a discussion of the proposed method, VGLS^{3}VM, which inherits the advantages of VGL. Before plugging each ADP component into the VGLS^{3}VM system, the initial supervised learning step trains the *action network* with the labeled set. Then, the S^{3}VM process begins, searching through the trajectory to find the optimal *action (hyperplane vector)* iteratively. The *state (label vector)* is adjusted correspondingly by the *model function*. Then, the *cost (fitness)* determines whether the adjusted values are appropriate. Finally, the *critic function* feeds the trajectory back to the action network to reflect the result, thus converging to the optima.

This paper is therefore one of few (e.g., [96, 97]) that successfully demonstrate the combination of supervised, unsupervised and reinforcement learning.

The remainder of the paper is organized as follows. Section 2 introduces S^3VM and VGL, followed by an explanation of the VGLS^3VM approach. In Section 4, the experimental setup, data description, results and comparison are presented. Finally, the conclusion is provided in Section 5.

2. BACKGROUND

2.1. SUPPORT VECTOR MACHINE AND SEMI-SUPERVISED SVM

Support vector machines (SVM) were originally introduced to geometrically interpret classification problems by finding a separating hyperplane in a multidimensional space [7, 83, 98]. With respect to sparse and noisy data, SVM is well suited for applications such as pattern recognition and biological data analysis. The main goal is to find the hyperplane that is maximally distant from each cluster. It is also possible to apply SVM on a non-linear separation with the *kernels* technique by converting the mapping into a linear space.

Given a training set of n data points with s dimensions, the set D can be defined by:

$$D = \{(\mathbf{d}_i, c_i) | \mathbf{d}_i \in \mathbb{R}^s, c_i \in (-1, 1), 1 \leq i \leq n\}, \tag{1}$$

where c_i indicates the label to which the ith pattern $\mathbf{d}_i$ belongs, such that $c_i = \text{hardlims}(f(\mathbf{d}_i))$, where *hardlims* is the symmetrical hard limit function and f is the hyperplane function defined by:

$$f(\mathbf{d}_i) = \mathbf{w}^T \cdot \mathbf{d}_i + b, \tag{2}$$

where $\mathbf{w}$ is the vector lying perpendicular to the hyperplane, and b is the intercept of the hyperplane. The goal of SVM is to find a classifier that maximizes the margin of each decision boundary. The distance d between the hyperplane and two decision boundaries, $\mathbf{w}^T \cdot \mathbf{d}_a + b = 1$ and $\mathbf{w}^T \cdot \mathbf{d}_b + b = -1$, can be calculated by:

$$d = 2|\mathbf{w}^T \cdot \mathbf{d}_a + b| / \|\mathbf{w}\| = 2/\|\mathbf{w}\|. \tag{3}$$

The optimal solution is to maximize d, thus minimizing $\|\mathbf{w}\|$ subject to $t_i(\mathbf{w}^T \cdot \mathbf{d}_i + b) \geq 1$.

The semi-supervised SVM (S^{3}VM) learning model aims to handle a limited amount of labeled data by combining supervised and unsupervised learning. A wide spectrum of techniques have been applied to solve the non-convex optimization problem associated with S^{3}VMs [84, 85, 86, 87, 88, 89, 99, 100, 101].

The training set consists of n_l labeled patterns $\{\mathbf{d}_i, c_i\}_{i=1}^{n_l}$, and n_u unlabeled patterns $\{\mathbf{d}_i\}_{i=n_l+1}^{n}$, with $n = n_l + n_u$. In S^{3}VM, the following optimization problem must be solved over both the label vector $\mathbf{t}'_u = [c_{n_l+1} ... c_n]^T$ and the hyperplane parameters $(\mathbf{w}, b)$,

$$I(\mathbf{w}, b, \mathbf{t}'_u) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n_l} V(c_i, o_i) + C^* \sum_{i=n_l+1}^{n} V(c_i, o_i) \tag{4}$$

where $o_i = \mathbf{w}^T \cdot \mathbf{d}_i + b$, and V, a hinge loss function used for maximum-margin classification, is defined by:

$$V(c_i, o_i) = \max(0, 1 - c_i o_i)^2, \tag{5}$$

which heavily penalizes mismatch on the labeled patterns, as presented in the example shown in Fig. 2.

The first two terms in Eq. 4 represent a standard SVM, the third term incorporates unlabeled data, and C and C^* are weights that are predefined to reflect the confidence in the labeled and unlabeled patterns, respectively.

The two main strategies for optimizing I are combinatorial optimization and continuous optimization. The former method explicitly uses $\mathbf{t}'_u$, the binary labels of the unlabeled patterns to find the minima of I. Branch-and-Bound (BB) [84] is one of the easiest methods; it forms a decision tree through every possible selection and then searches through it

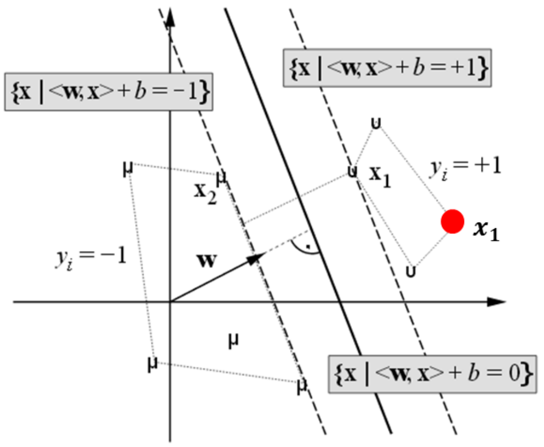


Figure 2. Example of hinge loss function. The correct label of pattern x_1 is 1. Assuming $o_1 = 2$, if $c_1 = 1$ (correct), then $V(c_i, o_i) = 0$, while $V(c_i, o_i) = 9$ if $c_1 = -1$ (incorrect), resulting in a high penalty.

to find the best solution. Due to the exhaustive nature of this method, it will only work with small data sets. Several other combinatorial methods, such as S^{3}VM^{light} and Convex relation [86, 101], are expensive and scale poorly. On the other hand, continuous optimization aims to adjust $(\mathbf{w}, b)$, the hyperplane, and the label vector is simply the symmetrical hard limit of o_i. Therefore, by eliminating $\mathbf{t}'_u$, the target function I can be rewritten as:

$$I(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n_l} V(c_i, o_i) + C^* \sum_{i=n_l+1}^{n} \max(0, 1 - |o_i|)^2. \quad (6)$$

The concave convex procedure (CCCP), ∇S^{3}VM and Newton S^{3}VM [99, 102, 103] are methods for discovering the global optima, as Eq. 6 is a non-convex function. Continuous optimization methods tend to perform better and faster than combinatorial methods.

2.2. VALUE GRADIENT LEARNING (VGL)

Value Gradient Learning (VGL) methods [92] were developed in an attempt to eliminate the requirements of the Bellman equation, namely, the exploration of the state space. In contrast, VGL only requires the value gradient along a single trajectory [104]. Therefore, it is likely that using VGL over value learning (VL) is significantly more efficient.

The VGL system consists of two main neural networks - the critic function and the action network. Let $\mathbf{x}$ and $\mathbf{u}$ be the state vector and the action vector, respectively, then the typical critic function $\widetilde{J}(\mathbf{x},\mathbf{w})$ in VL methods is redefined in VGL as $\widetilde{G}(\mathbf{x},\mathbf{w}) = \frac{\partial \widetilde{J}(\mathbf{x},\mathbf{w})}{\partial \mathbf{x}}$. Then, the VGL algorithm can be defined as a weight update of:

$$\Delta \mathbf{w} = \alpha \sum_t \left(\frac{\partial \widetilde{G}}{\partial \mathbf{w}} \right)_t (G'_t - \widetilde{G}_t) \tag{7}$$

where α is the learning rate and G'_t is the *target value gradient* defined by:

$$G'_t = \left(\frac{DU}{D\mathbf{x}} \right)_t + \gamma \left(\frac{Df}{D\mathbf{x}} \right)_t (\lambda G'_{t+1} + (1-\lambda)\widetilde{G}_{t+1}) \tag{8}$$

where U is the cost function, and $\lambda \in [0,1]$ is a constant. If $\lambda = 0$, VGL is equivalent to DHP. However, if $\lambda > 0$, the stability of learning improves. The learning speed is another factor affected by λ. When λ is high, the critic uses a longer look-ahead along the trajectory, resulting in faster learning. However, if λ is too large, the variance of the target function increases and consequently lowers the speed, especially in stochastic environments. Thus, the optimal λ is often set in the middle range, $\lambda \in [0,1]$ [105]. $\frac{D}{D\mathbf{x}}$ is the equivalent of:

$$\frac{D}{D\mathbf{x}} \equiv \frac{\partial}{\partial \mathbf{x}} + \frac{\partial A}{\partial \mathbf{x}} \frac{\partial}{\partial \mathbf{u}}. \tag{9}$$

The recursion of Eq. 8 is guaranteed to converge when $\gamma < \frac{1}{\lambda}$. The action network with the weight vector $\mathbf{z}$ is defined by:

$$\mathbf{u}_t = A(\mathbf{x}, \mathbf{z}), \tag{10}$$

and is trained using the method employed in [93], which uses the following weight update for the action network at each time step t:

$$\Delta \mathbf{z} = -\beta \left(\frac{\partial A}{\partial \mathbf{z}} \right)_t \left(\left(\frac{\partial U}{\partial \mathbf{u}} \right)_t + \gamma \left(\frac{\partial f}{\partial \mathbf{u}} \right)_t \widetilde{G}_{t+1} \right), \tag{11}$$

where β is a separate learning rate for the action network, and f is the model function, which predicts the next state $\mathbf{x}_{t+1}$.

Algorithm 1 depicts the overall pseudo code of the VGL implementation. This algorithm makes a forward pass through the trajectory, storing all states and actions, followed by a backward pass through the trajectory, accumulating G'_t by the recursion of Eq. 8.

Algorithm 1 Pseudo Code of VGL implementation [92]

```
$t \leftarrow 0$
{Unroll trajectory...}
while not terminated($\mathbf{x}$) do
    $\mathbf{u} \leftarrow A(\mathbf{x}_t, \mathbf{z})$
    $\mathbf{x}_{t+1} \leftarrow f(\mathbf{x}_t, \mathbf{u}_t)$
    $t \leftarrow t+1$
end while
$F \leftarrow t$
$\mathbf{p} \leftarrow \left(\frac{\partial U}{\partial \mathbf{x}}\right)_t, \Delta\mathbf{w} \leftarrow \mathbf{0}, \Delta\mathbf{z} \leftarrow \mathbf{0}$
{Backwards pass...}
for $t = F-1$ to 0 step $-1$ do
    $G'_t \leftarrow \left(\frac{\partial U}{\partial \mathbf{x}}\right)_t + \lambda \left(\frac{\partial f}{\partial \mathbf{x}}\right)_t \mathbf{p}$
        $+ \left(\frac{\partial A}{\partial \mathbf{x}}\right)_t \left(\left(\frac{\partial U}{\partial \mathbf{u}}\right)_t + \gamma \left(\frac{\partial f}{\partial \mathbf{u}}\right)_t \mathbf{p}\right)$
    $\Delta\mathbf{w} \leftarrow \Delta\mathbf{w} + \left(\frac{\partial \widetilde{G}}{\partial \mathbf{w}}\right)_t (G'_t - \widetilde{G}_t)$
    $\Delta\mathbf{z} \leftarrow \Delta\mathbf{z} - \left(\frac{\partial A}{\partial \mathbf{z}}\right)_t \left(\left(\frac{\partial U}{\partial \mathbf{u}}\right)_t + \gamma \left(\frac{\partial f}{\partial \mathbf{u}}\right)_t \widetilde{G}_{t+1}\right)$
    $\mathbf{p} \leftarrow \lambda G'_t + (1-\lambda)\widetilde{G}_t$
end for
$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Delta\mathbf{w}$
$\mathbf{z} \leftarrow \mathbf{z} + \beta\Delta\mathbf{z}$
```

3. VALUE GRADIENT LEARNING SEMI-SUPERVISED SUPPORT VECTOR MACHINES (VGLS^{3}VM)

The VGLS^{3}VM method was devised to achieve faster and better convergence to the optimal S^{3}VM solution. To achieve this objective , a hybrid machine learning model was designed by applying the S^{3}VM optimization problem with VGL. The hyperplane parameters and the label vector interact continuously until the result converges to an optimum, so VGL is capable of serving as an appropriate tool for S^{3}VM optimization.

As noted previously, the state vector **x** is the combination of the label vectors of both the labeled and unlabeled patterns with their respective sizes.Thus the state vector **x** is defined by:

$$\mathbf{x} = [\tau_l, n_l, \tau_u, n_u]^T, \tag{12}$$

where τ_l and τ_u are the tag vectors of the labeled and unlabeled patterns, respectively and n_l and n_u are the number of labeled and unlabeled patterns, respectively. The action vector **u** is the combination of the hyperplane parameters defined by $\mathbf{u} = [w_s, ..., w_1, b]^T$, where s is the dimension of the pattern.

The model function f is defined by:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = [\tau_{l_{t+1}}, n_l, \tau_{u_{t+1}}, n_u]^T \tag{13}$$

where

$$\tau_{l_{t+1}} = \{\text{hardlims}\left(\mathbf{w}_t^T \cdot \mathbf{d}_i + b\right) | i \in \mathbb{Z}, 1 \leq i \leq n_l\}, \tag{14}$$

$$\tau_{u_{t+1}} = \{\text{hardlims}\left(\mathbf{w}_t^T \cdot \mathbf{d}_i + b\right) | i \in \mathbb{Z}, n_l + 1 \leq i \leq n\}, \tag{15}$$

and the cost function $U(\mathbf{x}_t, \mathbf{u}_t)$ is a version of Eq. 6 modified to fit the notation, which is defined by:

$$U(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n_l} V'(c_i, o_i') + C^* \sum_{i=n_l+1}^{n} \max(0, 1 - |o_i'|)^2, \tag{16}$$

where $\mathbf{w} = [w_s, ...w_1]^T$, $c_i = o_i', \forall i \in (n_l, n)$ and, $o_i' = \text{hardlims}\left(\mathbf{w}^T \cdot \mathbf{d}_i + b\right)$.

This contribution reflects continuous optimization because the VGL method trains the action network, the output of which is the hyperplane parameter. One prerequisite of continuous optimization is the balancing constraint, which limits the number of patterns belonging to each cluster because of the possibility of the hyperplane being adjusted such that all of the unlabeled patterns belong to the same cluster. Thus, o_i' is modified such that the first n_r or fewer patterns are set to 1 by sorting $m_i = \left(u_1 + \sum_{j=1}^{s} u_{j+1} d_{i_j}\right)$ in a descending order as long as $m_i \geq 1$. The restraint factor n_r is typically set to $n/2$.

For the action network and critic function, a general neural network architecture, as shown in Fig. 3, is applied, which is a fully connected feed-forward neural network with all shortcut connections. The number of internal nodes is adjusted based on the input dimension s.

The learning rates of the critic and the actor neural network are adjusted based on the number of iterations. Since the supervised SVM process sets up the actor initially as shown in 1, high learning rate of the actor, β, causes the actor to deviate quickly from the initially set weights. While preventing the actor from rapidly diverging, the critic needs to converge faster with higher critic learning rate, α. Thus, in the early stage of iteration,

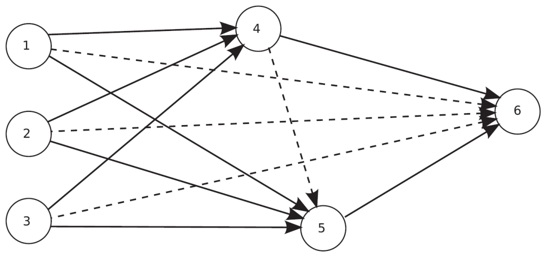


Figure 3. Example neural network including shortcut connections with 3 input nodes and 1 output node. Removing the dashed lines by setting those weights to 0 yields a more traditional layered network with a single hidden layer.

the learning rates α and β are initially set near 0.9 and 0.5, respectively and gradually converge to a preset value, 0.1 on each iteration. To prevent the hyperplane being stuck in the local optima, we added a neural network reset feature while keeping track of the discovered local optimum. When the reset is triggered, learning rate β is set back to 0.5, with the same decay rate.

In the proposed method, the progress towards the converged optima is less considered, as long as VGLS^{3}VM can discover the best optima while exploring the problem space. Thus the discount factor γ was to 1.0. The hyper parameter λ was set to 0.5 based on [105] while $\gamma < \frac{1}{\lambda}$ to ensure convergence to an optima.

4. EXPERIMENTAL RESULTS

4.1. SETUP

We began the simulation with the data sets summarized in Table 1. The *wine* data set was derived from the chemical analysis of 13 chemical substances found in each of three types of wines. The *Swiss roll* and *g50c* data sets were artificially generated based on Gaussians. The *Swiss roll* data set was generated by converting Gaussian distributed data points with the equation given by:

$$(x,y) \rightarrow (x\cos x, y, x\sin x), \tag{17}$$

which results in the plot of data points shown in Fig. 4. The *g50c* set was generated from two standard normal multivariate Gaussians, with different Gaussians and means for each class. The shape of the patterns for another artificially generated data set, *2moons*, is shown in Fig. 5.

Table 1. Characteristics of data sets [86, 104, 105, 106, 107, 108, 109, 110]

Name	Dimension	Samples	Labeled
Wine	13	130	10
Swiss Roll	3	2000	50
SecStr	314	83,679	1,000
g50c	50	550	50
Text	7511	1946	50
Uspst	256	2007	50
Isolet	617	1620	50
Coil20	1024	1440	40
2moons	102	200	4
ml1m	4000	6000	600
ml10m	10000	100000	1000

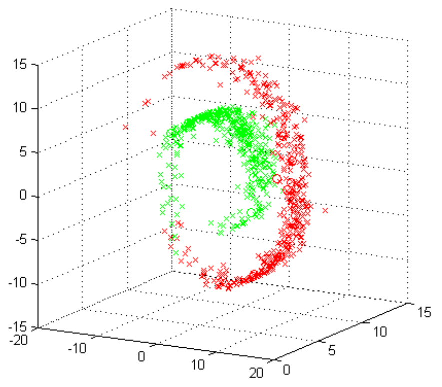


Figure 4. Two-class *Swiss roll* data set represented on a 3-dimensional plane.

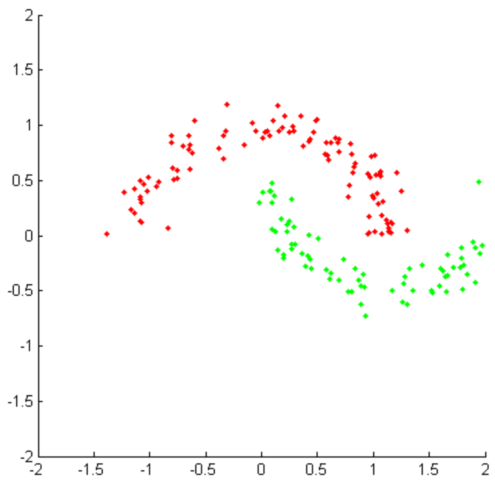


Figure 5. Two-class *2moons* data set.

The *Text* data set was defined using the classes *mac* and *mswindows* in the *Newgroup20* data set. The *Uspst* set is a collection of handwritten digit recognition features from USPS data. The *Isolet* data set is a subset of the ISOLET spoken letter database containing the sets of 1, 2, 3, and 9 confusing letters { B, C, D, E, G, P, T, V, Z} spoken. The *Coil20* set is a combination of gray-scale images of 20 different objects taken from different angles, at 5-degree intervals.

All experiments used the same Gaussian kernel given by:

$$k(x,y) = \exp(-\|x-y\|^2/2\sigma^2) \tag{18}$$

where σ was adjusted based on the estimated standard deviation within neighborhoods. For multi-class data sets, we applied the one-vs-rest approach. The balancing constraint was set to follow the number of patterns labeled 1 from the tag information.

The protein secondary structure prediction task (SecStr) [111] is an extensive data set in terms of features. The classification problem involves predicting the secondary structure of a given amino acid in a protein based on a sequence window centered around that amino acid.

The *ml1m* and *ml10m* [110] are the movie ratings data sets collected by the GroupLens research group. In order to make it a classification setup with two labels, we preprocessed the ratings by giving 1 for scores of 3 or above and -1 otherwise. For the experiments, 10% of the each data set were randomly chosen as the labeled set.

4.2. RESULTS

Table 2 presents the error rate of the *wine*, *Swiss roll*, *ml1m* and *ml10m* data sets. The BB and CCCP methods were coded for comparison on the same experimental platform.

Table 2. Unlabeled error rates of investigated methods in percentage

Method	*wine*	*Swiss roll*	*SecStr*	*ml1m*	*ml10m*
VGLS^{3}VM	6.7	8.3	29.86	14.26	23.44
BB	6.7	14.8	39.11	49.39	n/a
CCCP	7.3	9.4	31.79	22.95	37.18

Unlike conventional S^{3}VM methods, VGLS^{3}VM requires more than one iteration in order to allow VGL to explore several trajectories until it converges to the optima. Despite this requirement, VG3SVM performs better than BB and CCCP in terms of error rate. BB worked slower than both CCCP and VGLS^{3}VM and often failed to find any optima on *SecStr*, *ml1m*, which are larger data sets than *wine* and *Swiss roll* in terms of sample size and dimension. For *ml10m*, BB did not give any proper result due to both the dimension and quantity of the data set being the largest. The *wine* and *Swiss roll* sets required 30 and 80 iterations, respectively, to reach the optima. Fig. 6 shows the transformations of the hyperplane from the initial SVM state to the completed state.

For *SecStr*, *ml1m*, and *ml10m*, the iteration counts needed to converge to the best optima are shown in Table 3. Due to the randomness in the neural network reset feature, the required iteration varied throughout each trial but eventually converged to the exact same hyperplane.

Table 3. Required number of iterations to converge on larger datasets in 20 trials

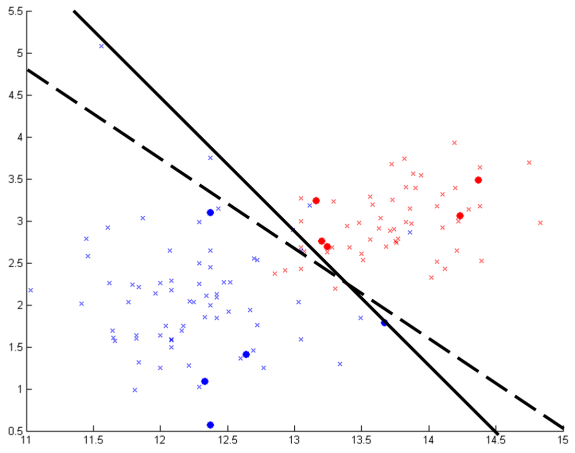
Dataset	Average	Min	Max	StdDev
SecStr	942.8	786	1410	152.36
ml1m	837.65	521	1548	258.71
ml10m	2382.45	1749	4461	686.45

The cost per iteration during the training process is shown in Fig. 7, which presents the neural network reset mechanism allowing VGLS^{3}VM to avoid being stuck in local optima and eventually converges to the global optima.

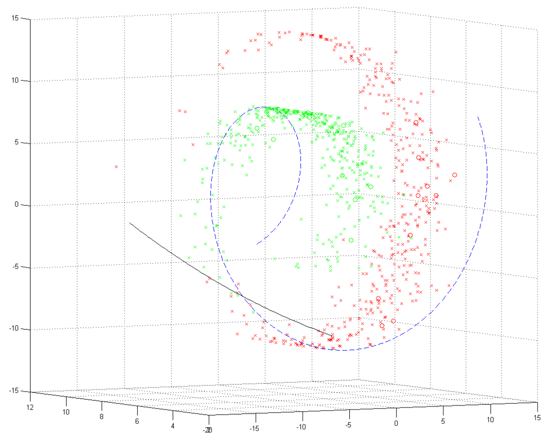
The experiments on the remaining data sets were compared with the results and setups from [28]. The results appear shown in Table 4, with the best methods for each data set shaded in gray. VGLS^{3}VM performed better than other S^{3}VM methods on some, but not all, of the data sets.

Table 4. Comparison of unlabeled error rate with published results [28]

Method	*g50c*	*Text*	*Uspst*	*Isolet*	*Coil20*	*2moons*
∇S^{3}VM	7.2	6.8	24.1	48.4	35.4	62.2
cS^{3}VM	6.6	5	41.5	58.3	51.5	33.7
CCCP	6.7	12.8	24.3	43.8	34.5	55.6
S^{3}VM^{light}	7.5	9.2	24.4	36	25.3	68.8
∇DA	8.4	8.1	29.8	46	12.3	22.5
Newton	5.8	6.1	25	45.5	25.4	8.9
VGLS^{3}VM	5.1	4.9	21.7	36.2	9.8	8.2
SVM	9.1	23.1	24.2	38.4	26.2	44.4

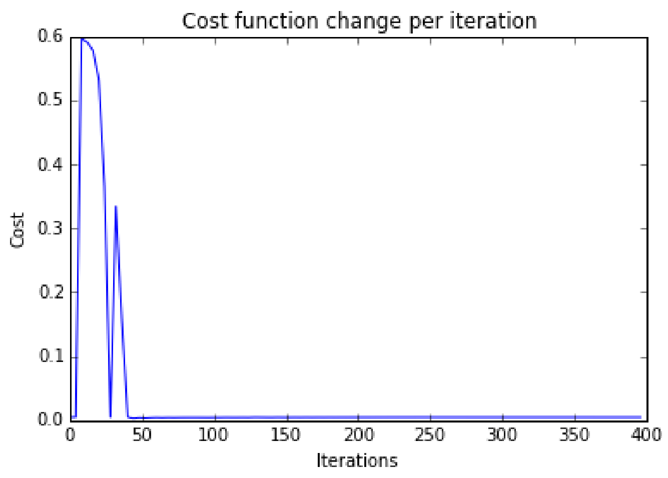


(a) *wine* data set

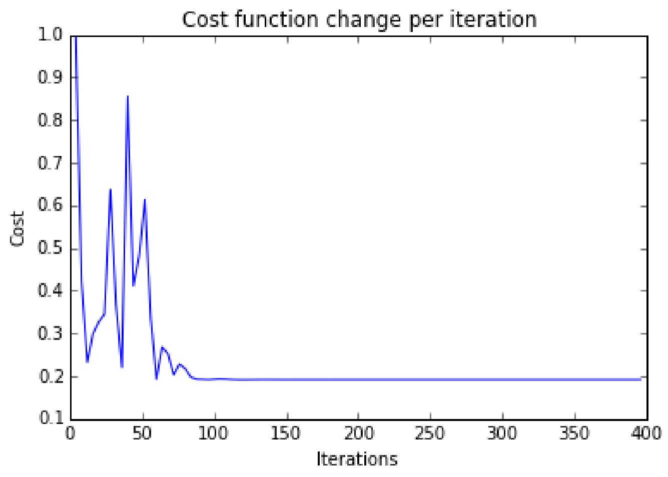


(b) *Swiss roll* data set

Figure 6. Hyperplane transition of *wine* and *Swiss roll* data sets. The dashed line represents the initial hyperplane acquired from the supervised SVM step. After 30 and 80 iterations, respectively, the hyperplanes converged to the solid lines, which represent the discovered optima.



(a) *wine* data set



(b) *Swiss roll* data set

Figure 7. The cost change of *wine* and *Swiss roll* data sets shown per iteration. The dashed line represents the initial hyperplane acquired from the supervised SVM step. In both cases, the cost increases throughout the training phase due to the reset feature which allows the neural network to escape the local optima.

5. CONCLUSION

VGLS^{3}VM uses the VGL approach to ADP in order to modify the S^{3}VM algorithm. The results indicate that VGLS^{3}VM can perform as good as, and in some cases better than, conventional S^{3}VM algorithms. Modifications to the procedure described in this paper, such as using modified kernel functions, warrant further exploration. These promising results open the possibility of combining reinforcement learning into supervised and unsupervised learning, and demonstrate that VGL should be considered in semi-supervised learning approaches.

IV. BICLUSTERING BASED PREDICTION WITH SUPERVISED BICLUSTERING ARTMAP

Sejun Kim and Donald C. Wunsch II

ABSTRACT

This paper presents a novel supervised learning algorithm based on an extension of biclustering. The hypothesis was that by forming associative priming connections between discovered biclusters and supervisory signals, better inference accuracy could be achieved. The approach was tested on synthetic data and sports statistics. The approach is called supervised biclustering ARTMAP (S-BARTMAP), a supervised biclustering based prediction made by modifying biclustering ARTMAP (BARTMAP), an unsupervised learning method, which itself is an extension of Adaptive Resonance Theory. In order to build a supervised version, an additional ART unit and an inter-ART module on BARTMAP were implemented which controls BARTMAP based on the supervised signal. Experiments were performed mainly on sports match forecasting with baseball, football and basketball statistics. The results show that S-BARTMAP is more precise than other approaches and is capable of providing insight on feature to label correlation, such as the weight of features on contribution to winning.

1. INTRODUCTION

Sports statistics has become a technology driver for various analytical techniques [114, 115, 116, 117, 118]. Analysts began building mathematical models to predict performances of the players and invent new features to find a better way to evaluate them. For example, a new metric to measure baseball pitchers is walks plus hits per inning pitched (WHIP) [119]. Utilizing data mining tools to discover the relevancy of features towards winning has been an important factor when optimizing team performance within a budget.

Biclustering, first used by Cheng and Church [18] in the community of bioinformatics, is a variant of clustering by considering the local relationship between subsets of samples and subsets of features. In bioinformatics, biclustering indicates gene groups that display similar patterns across a set of conditions (important to gene functional annotations and co-regulated gene identification[78, 120, 121, 122, 123]) or gene groups that are related to certain cancer types [123, 124, 125, 126]. In fields other than bioinformatics, biclustering is also known as subspace clustering, co-clustering or block clustering [50, 51, 127].

Biclustering ARTMAP (BARTMAP) [21] is a variant of adaptive resonance theory (ART) which performs simultaneous clustering by applying a correlation factor through an interconnected module, inspired by the design of ARTMAP [13, 14]. In comparison with alternative approaches [78, 79, 128], experimental results acquired from BARTMAP presented strong performance when handling high dimensional samples, such as Leukemia or Small, Round, Blue-Cell Tumors of childhood cancer (SRBCT) data sets [77, 129].

In this paper, a supervised biclustering model named Supervised Biclustering ARTMAP (S-BARTMAP) is developed to achieve clustering and correlation based feature selection and forecasting through supervised signals. The method is composed of three ART networks: ART_a and ART_b unit for biclustering in the BARTMAP module and the ART_c unit for supervised signal based control. The features and samples go through clustering in the BARTMAP and the labels are registered at the ART_c unit to determine whether the trained unsupervised biclusters are correctly matching the supervised signal. Online training is also implemented by the inter-BARTMAP module that is built to update the weights of BARTMAP received from the ART_c neural network.

In Section 2, ART, ARTMAP and BARTMAP are introduced. The algorithm and implementation is then given in Section 3, followed by experimental results on synthetic and real world sports data sets in demonstrated Section 4.

2. BACKGROUND

2.1. ADAPTIVE RESONANCE THEORY AND ARTMAP

Adaptive resonance theory (ART) is an unsupervised learning method which overcomes the "stability-plasticity dilemma" [10]. The first introduced ART, named ART1, deals with binary data and the variant to handle arbitrary data was developed which is known as Fuzzy ART [12].

The basic FA architecture is composed of two-layer of neurons, the feature representation field F_1 and the category representation field F_2, as shown in Fig. 1. The neurons in layer F_1 are activated by an input pattern, normalized with the complement coding rule to avoid category proliferation [12]. The prototypes of formed clusters are stored in layer F_2, which are initially composed of one uncommitted category set as 1. The neurons in layer F_2 that are already being used as representations of input patterns are said to be *committed*. The two layers are connected via adaptive weight $\mathbf{w_j} \in \mathbf{W}$, emanating from node j in layer F_2.

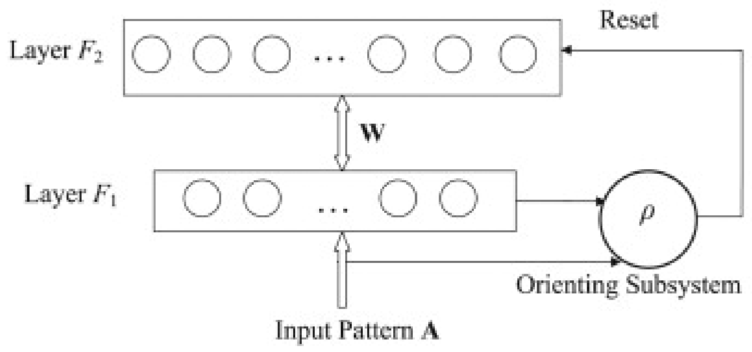


Figure 1. Topological architecture of ART. Two layers are included in the attentional subsystem, connected via bottum-up and top-down adaptive weights. The interaction between the neuron layers are controlled by a vigilance parameter ρ.

The summarized steps of FA are as follows:

- Category choice: When a new input pattern $\mathbf{A}$ is appointed, the nodes in layer F_2 compete by calculating the category choice function, defined as

$$T_j = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{1}$$

  where $\wedge$ is the fuzzy AND operator and α is the choice parameter to break the tie when more than once prototype vector is a fuzzy subset of the input pattern.

- Category selection: Once all T_j is calculated, neuron J becomes activated with the winner-take-all rule by $T_J = \max\{T_j | \forall j\}$.

- Category match: The winning neuron is then tested with the vigilance criterion. If

$$\rho \leq \frac{|\mathbf{A} \wedge \mathbf{w}_J|}{|\mathbf{A}|}, \tag{2}$$

  weight adaption occurs, which is called resonance. In case the winning neuron J does not meet 2, the corresponding neuron is removed from the competition and repeats the category match step with the neuron with the next largest T_j.

- Learning: The weight vector of the winning neuron is updated by the learning rule,

$$\mathbf{w}_J(new) = \beta(\mathbf{A} \wedge \mathbf{w}_J(old)) + (1-\beta)\mathbf{w}_J(old), \tag{3}$$

  where $\beta \in [0,1]$ is the learning rate parameter.

In the case that an uncommitted neuron is selected during the category match step, a new uncommitted neuron is created to represent a potential new cluster. This will maintain a consistent supply of uncommitted neurons.

ARTMAP is a fast, stable learning method in a supervised setting derived from ART [13]. For hetero-associative tasks, two connected FA units are required with each unit receiving either the input or output component of each pattern pair to be associated. Thus, the input and output spaces are organized into distinct categorized sets during processing. Fuzzy ARTMAP uses layer of nodes, called the inter-ART module or the map field [14], to link the two FA units, ART_a and ART_b. The main function of the map field is to associate compressed representations of the original input and output components.

In the context of supervised classification, the input pattern is presented to the ART_a unit and the corresponding label is presented to the ART_b unit. The vigilance parameter ρ_b of ART_b is set to 1, so that each label represents a specific cluster. The input-output association is stored in the weights $\mathbf{w}^{ab}$ of the inter-ART module. The jth row of the inter-ART module weights $\mathbf{w}_j^{ab}$ denotes the weight vector from the jth neuron in the ART_a to the map field. When the map field is activated, the output vector of the map field is

$$\mathbf{x}^{ab} = \mathbf{y}^b \wedge \mathbf{w}_j^{ab}, \tag{4}$$

where $\mathbf{y}^b$ is the binary output vector of field F_2 in ART_b and $\mathbf{y}_j^b = 1$ if and only if the jth category wins in ART_b.

Similar to the category match mechanism in FA, the map field also performs a vigilance test, such that if

$$\rho_{ab} > \frac{|\mathbf{x}^{ab}|}{|\mathbf{y}^b|}, \tag{5}$$

where $\rho_{ab}(0 \leq \rho_{ab} \leq 1)$ is the map field vigilance parameter, a match tracking procedure is activated, where the ART_a vigilance parameter ρ_a is increased from its baseline vigilance $\bar{\rho_a}$ by a preset value $\sigma(0 < \sigma \ll 1)$. This procedure occurs when the current winning neuron in ART_a does not comply with the label represented in ART_b. The unmatched winning neuron will be removed from the competition and remaining neurons will continue to compete until both units find a match. If none of the committed nodes wins, ART_a will assign the input to an uncommitted neuron, indicating that a new category has been created.

In the test phase where only an input pattern is provided to ART_a without the corresponding label to ART_b, no match tracking occurs and the prediction is obtained through the map field weights of the winning ART_a neuron. In case the predicted neuron is an uncommitted node, then the input pattern cannot be classified solely from the training set.

2.2. BICLUSTERING ARTMAP

Biclustering ARTMAP (BARTMAP) is an FA based biclustering algorithm which has performed better in bioinformatics data sets [21]. BARTMAP is composed of two FA units which focus on the sample inputs and the gene (feature) inputs, respectively, putting it into the category of two-way clustering that is considered to be conceptually simpler than other biclustering algorithms [51]. The overall structure of BARTMAP is shown in Fig. 2.

The first phase of BARTMAP is to perform FA clustering on the features by using the ART_b module, generating K_f clusters $\mathbf{F}_i$, $i = 1, ..., K_f$, for N features. In the following phase, each sample is presented into the ART_a module to check with the existing committed neurons (clusters). If an uncommitted neuron passes the category match test, learning will occur to create a new single element sample cluster as is in FA. However, if an already committed neuron is picked as a winning neuron candidate, the learning will occur if and only if it passes the correlation test through the Inter-ART module.

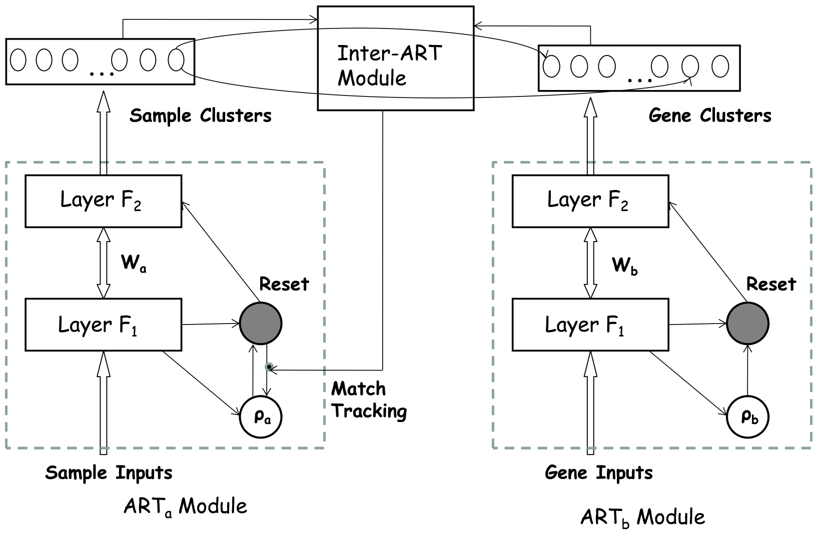


Figure 2. Structure of BARTMAP. Gene clusters are formed in the ART_b module and sample inputs are processed through the ART_a module. Before generating samples clusters with the winning neuron in ART_a, the Inter-ART module checks the correlation with other members in the category across the gene clusters in ART_a. If the condition is not met, the Inter-ART module adjusts the vigilance ρ_a of the ART_a module.

The similarity between the new sample s_K and the sample cluster $\mathbf{S}_j = \{s_{j1}, s_{j2}, ..., s_{jM_j}\}$ with M_j samples across a feature cluster $\mathbf{F}_i = \{f_{i1}, f_{i2}, ..., f_{iN_i}\}$ with N_i genes is calculated as the average Pearson correlation coefficient between the samples and all other samples in the cluster by:

$$r_{kj} = \frac{1}{M_j} \sum_{i=1}^{M_j} r_{k,jl}, \tag{6}$$

where

$$r_{k,jl} = \frac{\sum_{t=1}^{N_i} (e_{s_k f_{it}} - \bar{e}_{s_k F_i})(e_{s_{jl} f_{it}} - \bar{e}_{s_{jl} F_i})}{\sqrt{\sum_{t=1}^{N_i} (e_{s_k f_{it}} - \bar{e}_{s_k F_i})^2} \sqrt{\sum_{t=1}^{N_i} (e_{s_{jl} f_{it}} - \bar{e}_{s_{jl} F_i})^2}}, \tag{7}$$

and

$$\bar{e}_{s_k F_i} = \frac{1}{N_i} \sum_{t=1}^{N_i} e_{S_k f_{it}}, \tag{8}$$

$$\bar{e}_{s_{jl} F_i} = \frac{1}{N_i} \sum_{t=1}^{N_i} e_{S_{jl} f_{it}}. \tag{9}$$

The new sample s_K is enclosed in the cluster $\mathbf{S}_j$ only when the average Pearson correlation r_{kj} is above some threshold η and learning will occur correspondingly following the weight update rule of FA.

If the correlation is below the threshold, the match tracking mechanism in the Inter-ART module will increase the vigilance parameter ρ_a of the ART_a module to disable the current winning neuron and search for other candidates until a sample cluster is found which passed the category match test and correlation test. If no existing neuron matches any criteria, a new sample cluster will be automatically generated.

3. SUPERVISED BICLUSTERING ARTMAP

Similar to Fuzzy ARTMAP, the supervised biclustering ARTMAP (S-BARTMAP) consists of two modules, BARTMAP and ART_c, that are linked together via an additional inter-ART module, as shown in Fig. 3 .

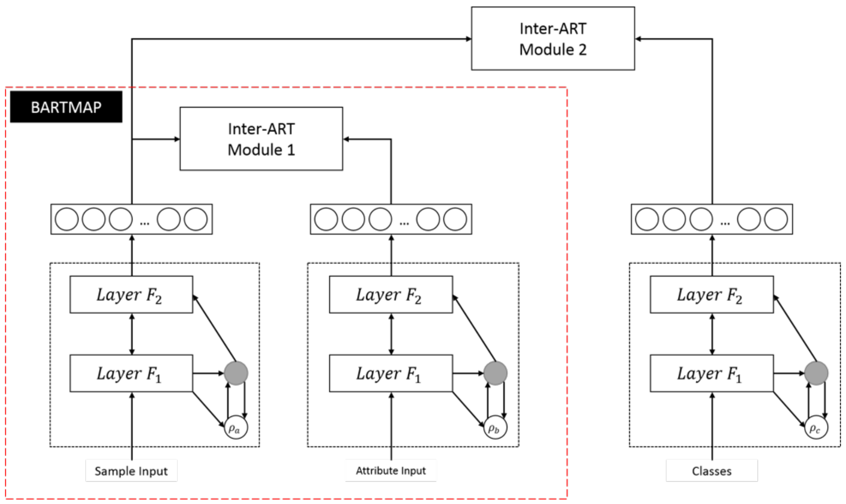


Figure 3. Overall design of S-BARTMAP. The BARTMAP unit takes the samples and features as inputs while the labels are presented to the ART_c unit, making it a supervised learning method. The inter-ART module checks whether a winning bicluster node matches the associated label and controls the behavior of BARTMAP correspondingly. The hypothesis is that the improved information content of biclusters will provide better heteroassociative matches than those coming from clusters, as is done in ARTMAP.

The main concept of S-BARTMAP is an expanded version of fuzzy ARTMAP, so that supervised classification is achieved through biclustering, with the aid of not only distances among samples but also correlation based similarity.

3.1. TRAINING

The first step of training is creating a set of K_f feature clusters $\mathbf{F}_i$, $i = 1, ..., K_f$, for N features by using the ART_b unit of the BARTMAP module. The goal of the following step is to create K_s sample biclusters $\mathbf{S}_j$, $j = 1, ..., K_s$, for M samples within the ART_a module while building the local relations between the samples and feature clusters, associated with the label input from the ART_c unit.

Upon the presentation of a new sample s_k, HBARTMAP proceeds to find a candidate neuron within the ART_a unit by testing the category match with Eq. 2 and average Pearson correlation coefficient with Eq. 6 across all other samples in the candidate cluster.

If a committed neuron j is selected as the candidate, the inter-ART module 2 compares the label input from ART_c with the label associated with the neuron. A sample is then absorbed into an existing cluster, updating the weights of ART_a with Eq. 3 if the label matches. Otherwise, the inter-ART module 2 increases the vigilance ρ_a of the ART_a unit to seek for other possible candidates until all three criteria - ART_a category match, similarity test and associated label match - meet the each condition. In case an uncommitted neuron is picked, a new cluster will be generated and the label input from ART_c is associated to the new cluster.

After all input samples for training is presented to the BARTMAP module, S-BARTMAP begins to form biclusters, which is the final step of training. The bicluster formation is performed by checking the evaluation fitness function described in [130].

Given a candidate bicluster B_{ij} composed of sample cluster $\mathbf{S}_{j_i} = \{s_{j_i1}, ..., s_{j_iM_j}\}$ across the features in the feature cluster $\mathbf{F}_i = \{f_{i1}, ..., f_{iN_i}\}$, the average correlation of B_{ij}, $\delta(B_{ij})$, is defined as,

$$\delta(B_{ij}) = \frac{1}{\binom{M_j}{2}} \sum_{k=1}^{M_j-1} \sum_{l=k+1}^{M_j} \delta(s_{j_ik}, s_{j_il}), \tag{10}$$

and

$$\delta(x,y) = \frac{cov(x,y)}{\sigma_x \sigma_y}, \tag{11}$$

where $cov(x,y)$ is the covariance of the variables x and y and σ_x and σ_y are the standard deviations of x and y, respectively. Since $\delta(s_{j_ik}, s_{j_il}) = \delta(s_{j_il}, s_{j_ik})$, only $\binom{M_j}{2}$ combinations in each bicluster are considered.

The fitness function $f(B_{ij})$, which prefers large volume biclusters, is defined by,

$$f(B_{ij}) = (1 - \delta(B_{ij})) + \sigma_\delta + c_1(1/M_j) + c_2(1/N_i), \tag{12}$$

where c_1 and c_2 are penalty factors to control the volume of the bicluster B_{ij}, and σ_δ is the standard deviation of all $\delta(s_{j_ik}, s_{j_il})$ from Eq. 10. If $f(B_{ij}) < \phi$, where ϕ is the bicluster formation factor, the features in $\mathbf{F}_i$ are associated with the sample cluster $\mathbf{S}_j$ to form a bicluster. In case multiple feature clusters meet the criterion, all the members in satisfied feature clusters will be included. All the features associated with sample cluster $\mathbf{S}_j$ are stored in $\mathbf{B}[j]$.

The overall training procedure is detailed in Algorithm 1. Note that there are two FA functions in the algorithm, named $faV1$ and $faV2$. The former behaves as an ordinary FA while the latter, $faV2$, is modified so that the weight update is not immediately reflected during the learning step since the inter-ART module 1 and 2 need to check the correlation coefficient and label input, respectively before updating the neurons.

Table 1. Change of input and output behaviors for ART modules in S-BARTMAP

Unit	Mode	
	Training	Testing
ART_a	Accept sample input	
ART_b	Accept feature input	No input taken
ART_c	Accept label input	Provide label output

3.2. TESTING

In the testing mode, the input and output behaviors of ART units are changed as depicted in Table 1. While the ART_a unit remains the same for training and testing mode, the ART_b unit will not accept any feature input and utilizes the trained neurons for prediction and the ART_c unit will provide the predicted label as an output.

Once a sample s_p is presented to S-BARTMAP, the ART_a unit first performs the category choice by calculating $T_j = \frac{|s_p \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, j = 1, ..., K_s$. The winning node J, obtained by $T_J = \max\{T_j | \forall j\}$, becomes the candidate and goes through Eq. 2 with the prediction vigilance ρ_{a_p}.

If node J passes the category match test, the result is sent to the inter-ART module 1, where the trained bicluster definitions are stored. Assuming that s_p belongs to the sample cluster $\mathbf{S}_J$, a bicluster B'_J is formed which contains $M_J + 1$ samples in $\mathbf{S}'_J = \{s_{J1}, ..., s_{JM_J}, s_p\}$ and features $f_J \in \mathbf{B}[J]$. The bicluster fitness is calculated and if $f(B'_J) < \phi_p$, where ϕ_p is the prediction fitness threshold, node J becomes the winning neuron and the ART_c unit returns the associated label l_J as the prediction for sample input s_p. On the other hand, if the $f(B'_J)$ does not meet the prediction fitness threshold, then the prediction vigilance ρ_{a_p} is increased by ε, and repeats the entire testing steps until S-BARTMAP finds a winning neuron.

The testing procedure is summarized in Algorithm 2.

Algorithm 1 Pseudo-code of S-BARTMAP training process

```
1: Get data set $\mathbf{A}$, labels $\mathbf{t}$
2: $\mathbf{X} \leftarrow$normalize$(\mathbf{A})$
3: Initialize parameters
4: procedure $ART_b$ FEATURE CLUSTERING($\mathbf{X}^T$)
5:     for each row $f_i \in \mathbf{X}^T$ do
6:         $\mathbf{F} \leftarrow faV1(f_i, \rho_b)$          ▷ $\mathbf{F_i} \in \mathbf{F}, i = 1, ..., K_f$
7:     end for
8: end procedure
9: procedure $ART_a$ SAMPLE CLUSTERING($\mathbf{X}$)
10:    for each row $s_k \in \mathbf{X}$ do
11:        $\rho_a \leftarrow \bar{\rho}_a$          ▷ Base vigilance $\bar{\rho}_a$
12:        $chk \leftarrow False$, $newNode \leftarrow False$
13:        $l' \leftarrow ART_c(t_j \in \mathbf{t})$          ▷ Get label from $ART_c$
14:        while $chk == False$ do
15:            $J', \mathbf{w}' \leftarrow faV2(s_k, \rho_a, \mathbf{w})$
16:            if $J'$ is uncommitted then
17:                $chk \leftarrow True$
18:                $newNode \leftarrow True$
19:            else
20:                if $l_{J'} == l'$ and $r_{kJ'} \geq \eta$ then
21:                    $chk \leftarrow True$
22:                else
23:                    $\rho_a \leftarrow \frac{|s_k \wedge \mathbf{w}'_{J'}|}{|s_k|} + \varepsilon$
24:                end if
25:            end if
26:        end while
27:        Register $s_k \rightarrow S_{J'} \in \mathbf{S}$
28:        $\mathbf{w} \leftarrow \mathbf{w}'$          ▷ Update weight
29:        if $newNode == True$ then
30:            $l_{J'} \leftarrow l'$
31:            $appendNewNode(\mathbf{w})$
32:        end if
33:    end for
34: end procedure
```

```
35: procedure FormBicluster($\mathbf{F}, \mathbf{S}, \phi$)
36:     $\mathbf{B} \leftarrow []$
37:     for $j = 1 \rightarrow K_s$ do
38:         $\mathbf{B}' \leftarrow []$                                    ▷ List of features for $\mathbf{S}_j$
39:         for $i = 1 \rightarrow K_f$ do
40:             Calculate $f(B_{ij})$
41:             if $f(B_{ij}) < \phi$ then
42:                 $\mathbf{B}' \leftarrow append(\mathbf{B}', \mathbf{F}_i)$
43:             end if
44:         end for
45:         $\mathbf{B}[j] \leftarrow \mathbf{B}'$
46:     end for
47: end procedure
```

Algorithm 2 Pseudo-code of S-BARTMAP testing mode

```
1: Get sample input $a_p$
2: $s_p \leftarrow$ normalize $a_p$
3: $pass \leftarrow False$
4: while $pass == False$ do
5:     procedure $ART_a$_GetCandidateNode($s_p$, $\rho_{a_p}$)
6:         $J \leftarrow faV2(s_p, \rho_{a_p}, \mathbf{w})$                    ▷ Get winning node $J$
7:     end procedure
8:     $B'_J \leftarrow (append(\mathbf{S}_J, s_p), \mathbf{B}[J])$
9:     procedure InterART2_GetFitness($B'_J$)
10:        $M_{J'}, N_{J'} \leftarrow sizeof(B'_J)$
11:        $f(B'_J) \leftarrow (1 - \delta(B'_J)) + \sigma_\delta + c_1 \frac{1}{M_{J'}} + c_2 \frac{1}{N_{J'}}$
12:    end procedure
13:    if $f(B'_J) < \phi_p$ then
14:        $pass \leftarrow True$
15:    else
16:        $\rho_{a_p} \leftarrow \rho_{a_p} + \varepsilon$
17:    end if
18: end while
19: $l_p \leftarrow l_J$
20: return $l_p$
```

4. EXPERIMENTAL RESULTS

4.1. DATA SETS

We first performed the experiment with various sizes of the synthetic data set developed by Handl and Knowles [41]. The characteristics of the applied data set are given in Table 2.

Table 2. Characteristics of synthetic data set

No.	Samples	Features	Labels
1	1286	100	4
2	2117	860	10
3	9841	1377	27

For each experiment with the synthetic data sets, half of each set was used for training and the other half for testing, implying that none of the samples in the testing phase was presented during the training phase.

For real world data experiments, we focused on sports statistics. The first is *baseball* data set which is composed of cumulative stats of players in every Major League Baseball (MLB) matches in the 2014 and 2015 season gathered from [129]. For better match result forecasting, we built two separate data sets, defense set and offense set which are based on how many runs the home team allowed and scored, respectively. Each sample in the defense set was aligned as offensive stats of away team in batting order, defensive stats of home team in specific position order. The offense set was arranged similarly with offensive stats of home team followed by defensive stats of away team. The labels were preprocessed into predefined buckets based on the runs. The match results of the entire 2014 season was used for training and that of 2015 season for testing and evaluation.

Similar to the *baseball* set, we collected the statistics of National Basketball Association (NBA) and National Football League (NFL) matches [130, 131] in the 2013-2014 season and the 2014-2015 season, named *basketball* set and *football* set, respectively. The sizes of sports statistics sets are detailed in Table 3.

Table 3. Characteristics of sports statistics sets

Name		Samples	Features	Labels
baseball	Offense	2430	144	5
	Defense	2430	144	5
basketball	Offense	1230	96	8
	Defense	1230	96	8
football	Offense	256	176	7
	Defense	256	154	7

With the points scored and allowed predicted with the offense and defense set, respectively, we eventually built home team win-loss predictor to calculate x_{win} based on the category match function and the bicluster fitness function defined in Eq. 2 and Eq. 10. The three possible outcome with the offense prediction label x_o and defense prediction label x_d are described as follows:

- $x_o > x_d$: Clearly the points scored by the home team is greater than the points allowed, thus $x_{win} = True$,
- $x_o < x_d$: Clearly the points scored by the home team is lesser than the points allowed, thus $x_{win} = False$,
- $x_o == x_d$: In this case, where the range of points scored by the home team is identical to that of points allowed. The category match value of all clusters in the offense and defense set associated to the higher adjacent label is derived and the node J_{eo} and J_{ed} with the highest value is chosen. Then the fitness for bicluster $B'_{J_{eo}}$ and $B'_{J_{ed}}$

are calculated. In case $f(B'_{J_{eo}}) > f(B'_{J_{ed}})$, implying the likelihood towards points allowed, $x_{win} = False$. Otherwise, $x_{win} = True$.

4.2. RESULTS

For the synthetic data set, we performed 10 trials by randomly selecting samples half of each set for S-BARTMAP training and the remaining half for testing. The average misprediction rates with minimum-maximum cases and standard deviation are shown in Fig. 4.

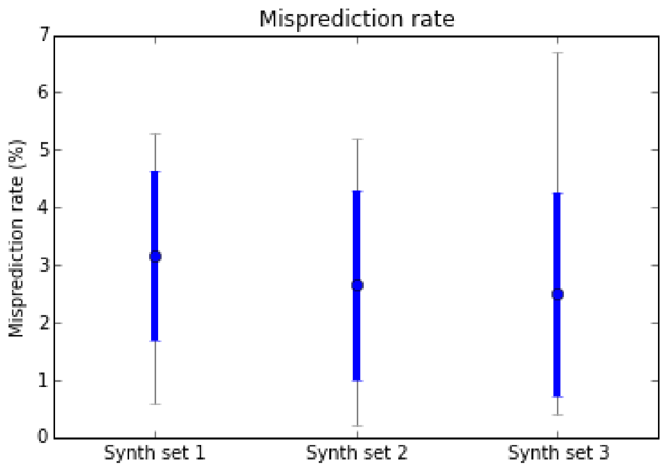


Figure 4. The misprediction rate on the S-BARTMAP experiment on synthetic data sets for 10 trials. The gray top and bottom peak represents the maximum and minimum errors, respectively and the thick line is the standard deviation.

The results on the synthetic data sets implies that as the size of the data set increases in terms of numbers of both samples and features, the more accurate prediction S-BARTMAP performs.

The experimental results on sports statistic data sets are presented in Table 4. The *baseball* set, which is the highest in terms of features used, predicted the runs scored and allowed with the best misprediction rate compared to other sports type. For the *basketball* set and *football* set, the defensive stats for players were much lower than those in the *baseball* set, resulting in significantly higher misprediction rates.

Table 4. Misprediction rates on sports statistics sets

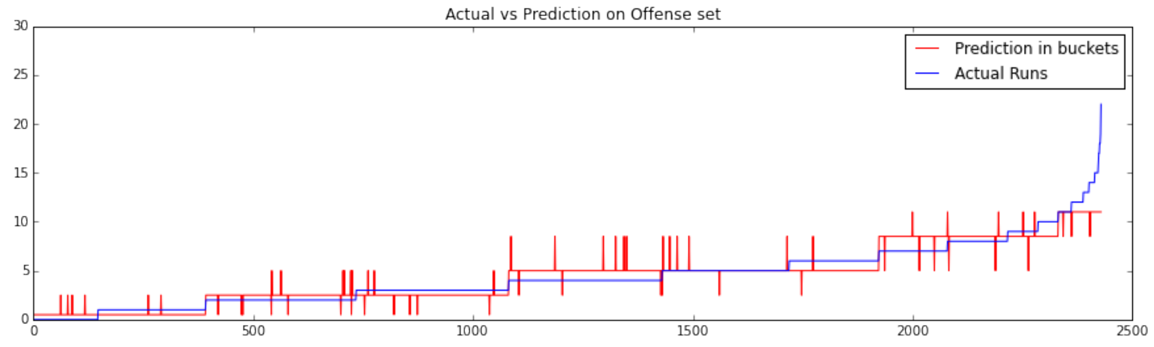
Data Set	Offense	Defense	W/L Prediction
baseball	2.49%	3.21%	2.20%
basketball	13.32%	21.64%	12.03%
football	16.91%	20.71%	14.85%

More details from the *baseball* set experiment result are shown in Fig. 5. Each spike in the red line implies S-BARTMAP misprediction compared to the actual scores.

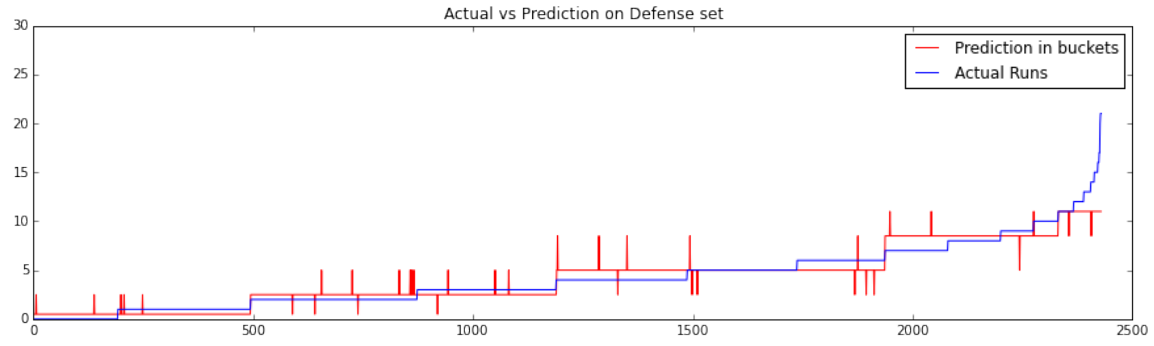
We compared the results from S-BARTMAP with other supervised learning tools, which are the classification and regression tree (CART), K nearest neighbor (KNN) and backpropagation neural network(BPNN) [134, 135, 136]. The misprediction rate with the comparison methods are shown in Table 5.

Table 5. Comparison of misprediction rate on win-loss prediction with other supervised learning methods

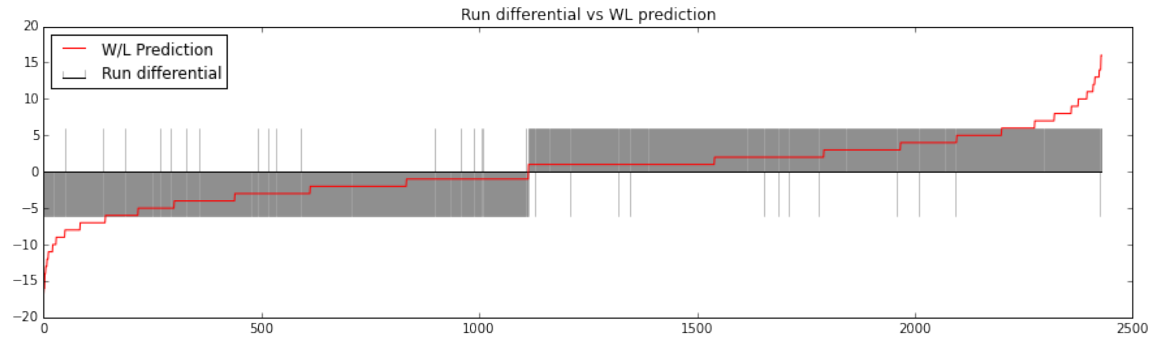
Data Set	CART	KNN	BPNN	Fuzzy ARTMAP	S-BARTMAP
baseball	9.94%	23.14%	31.44%	10.15%	2.20%
basketball	17.20%	31.62%	29.71%	16.77%	12.03%
football	21.88%	20.67%	37.02%	26.50%	14.85%



(a) *baseball* Offense set



(b) *baseball* Defense set



(c) *baseball* W/L prediction

Figure 5. The S-BARTMAP experimental results from *baseball* data set. The blue lines in Fig. 5a and Fig. 5b indicate the actual points scored and allowed, respectively while the red lines indicates the median of the range of each predicted label. Fig. 5c presents the win-loss prediction where the blue line is the point differential by $(points_{scored} - points_{allowed})$ and the gray bars are the win-loss predictions from 4.1. For better presentation, the results are sorted in ascending order based on the actual points and differential.

5. CONCLUSION

Inspired by the supervised classification system ARTMAP, we showed that an advanced version based on biclustering, the S-BARTMAP, is competitive on high dimensional sports statistics data analysis, for match result forecasting. Experimental results on simulated and real world data indicates the superior performance of this biclustering based classification method over commonly used supervised learning tools.

SECTION

2. CONCLUSION

In this dissertation, novel approaches on clustering and biclustering algorithms are presented that show how modifications on adaptive resonance theory based methods can improve the data mining performance on high dimensional data sets or data with limited informations. The first paper studied the ART tree on a parallelized GPU platform. While the improvement on the ART neural network training speed was significant, the possibility on the hierarchical implementation was discovered for deep, finesse clustering.

To efficiently perform clustering on high dimensional, noisy data sets, especially genetic data sets in bioinformatics, hierarchical BARTMAP was introduced in the second paper. As BARTMAP essentially being a co-clustering method with correlation based similarity measures and forming biclusters, a bicluster fitness function was developed to generate biclusters from the discovered sample and feature clusters from the two ART modules. The autonomous BARTMAP parameters - vigilance ρ and correlation coefficient threshold η - adjustment function based on the size of each bicluster allowed to discover higher quality biclusters as the tree is formed, through a layer suggestion function also included in HBARTMAP that calculates the internal criteria fitness for each layer in the HBARTMAP tree. Experimental results indicates that HBARTMAP is superior consistently across a range of problems, both synthetic and real world data sets.

Paper three suggests a hybrid learning approach on SSL problems, to overcome the challenge of large data sizes and limited amount of supervised labels. By utilizing the nature of VGL which is capable of efficiently converging to minima without exploring the entire data space and appropriately associating the S^3VM optimization problem to the cost function, VGLS^3VM performed as good as, and in some cases better than conventional, published S^{3}VM optimization methods. These promising results also opened the possibility of combining various learning schemes for more complicated problems.

In the final paper, a supervised classification and prediction method on high dimensional data sets by utilizing BARTMAP was proposed. Inspired by the transformation of ART to ARTMAP, an additional ART unit was implemented to take the supervised signal input and the mechanisms to control the behavior of BARTMAP and to generate biclusters associated with the labels were developed. The experimental results on match result forecasting with sports statistics data sets showed higher prediction rate, compared to traditional methods.

APPENDIX A

FUZZY ART

The Fuzzy ART neural network architecture is composed of two layers of neurons, which include the feature representation field F_1, and the category representation field F_2. The neurons in layer F_1 are activated by the input pattern, while the prototypes of the formed clusters are stored in layer F_2. The neurons in layer F_2 that already represent input patterns are said to be committed. Correspondingly, the uncommitted neuron encodes no input patterns. The two layers are connected via adaptive weights w_j, emanating from node j in layer F_2, which are initially set as 1. Once an input pattern $\mathbf{A}$ is registered, the neurons in layer F_2 compete by calculating the category function

$$T_j = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{13}$$

where $\alpha > 0$ is the choice parameter that breaks the tie when more than one prototype vector is a fuzzy subset of the input pattern, based on the winner-take-all rule,

$$T_J = max\{T_j | \forall j\}, \tag{14}$$

where J is the winning neuron. Then, it becomes activated, and an expectation is reflected in layer F_1 and compared with the input pattern. The orienting subsystem with the pre-specified vigilance parameter $\rho (0 \leq \rho \leq 1)$ determines whether the expectation and the input pattern are closely matched. If the match meets the vigilance criterion, that is, if

$$\rho \leq \frac{|\mathbf{A} \wedge \mathbf{w}_J|}{|\mathbf{A}|}, \tag{15}$$

weight adaptation occurs, where learning begins and the weights are updated using the following learning rule,

$$\mathbf{w}_J(new) = \beta(\mathbf{A} \wedge \mathbf{w}_J(old)) + (1-\beta)\mathbf{w}_J(old), \tag{16}$$

where β $(0 \leq \beta \leq 1)$ is the learning rate parameter, and $\beta = 1$ corresponds to fast learning.

On the other hand, if the vigilance criterion is not met, a reset signal is sent back to layer F_2 to ignore the winning neuron. A new competition will occur among the remaining neurons, excluding the ignored neurons. This process repeats until the vigilance criterion is met. If no existing neuron is selected for coding, a new uncommitted neuron is created to represent a new cluster, thus maintaining a consistent supply of uncommitted neurons.

APPENDIX B

FUZZY ARTMAP

By incorporating two fuzzy ART modules, which receive input patterns (ART_a) and corresponding labels (ART_b), respectively, with an inter-ART module, the resulting ARTMAP system can be used for supervised classifications. The vigilance parameter of ART_b is set to 1, which causes each label to be represented as a specific cluster. The information regarding the input-output associations is stored in the weights w^{ab} of the inter-ART module. The j^{th} row of the weights of the inter-ART module w_j^{ab} denotes the weight vector from the jth neuron in ART_a to the map field. When the map field is activated, the output vector of the map field is

$$\mathbf{x}^{ab} = \mathbf{y}^b \wedge \mathbf{w}_j^{ab}, \tag{17}$$

where y^b is the binary output vector of field F_2 in ART_b and $y_i^b = 1$ only if the i^{th} category wins in ART_b. Similar to the vigilance mechanism in ART_a, the map field also performs a vigilance test such that a match tracking procedure is activated if

$$\rho_{ab} > \frac{|\mathbf{x}^{ab}|}{|\mathbf{y}^b|}, \tag{18}$$

where ρ_{ab} $(0 \leq \rho_{ab} \leq 1)$ is the map field vigilance parameter. In this case, the ART_a vigilance parameter ρ_a is increased from its baseline vigilance to a value just above the current match value. This procedure ensures the shut-off of the current winning neuron in ART_a, whose prediction does not comply with the label represented in ART_b. Another ART_a neuron then will be selected, and the match tracking mechanism again will verify its appropriateness. If no such neuron exists, a new ART_a category is created. Once the map field vigilance test criterion is satisfied, the weight w_J^{ab} of the neuron J in ART_a is updated using the following learning rule:

$$\mathbf{w}_J^{ab}(new) = \gamma(\mathbf{y}^b \wedge \mathbf{w}_J^{ab}(old)) + (1-\gamma)\mathbf{w}_J^{ab}(old), \tag{19}$$

where $\gamma (0 \leq \gamma \leq 1)$ is the learning rate parameter of ART_a. Note that with fast learning ($\gamma = 1$), once neuron J learns to predict the ART_b category I, the association is permanent, i.e., $\mathbf{w}_{JI}^{ab} = 1$ for all input pattern presentations.

In the test phase in which only an input pattern is provided to ART_a without the corresponding label to ART_b, no match tracking occurs. The class prediction is obtained from the map field weights of the winning ART_a neuron. However, if the winning neuron is uncommitted, the input pattern cannot be classified solely based on prior experience. It would simply form a new, unclassified cluster.

BIBLIOGRAPHY

[1] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. The parable of google flu: traps in big data analysis. *Science*, 343(14 March), 2014.

[2] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[3] Brian S Everitt. Unresolved problems in cluster analysis. *Biometrics*, pages 169–181, 1979.

[4] B Everitt. Cluster analysis. 1980.

[5] AD Gordon. Classification. 1999. *Monographs on Statistics and Applied Probability*, 82, 1999.

[6] Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Mathematical programming*, 79(1-3):191–215, 1997.

[7] Rui Xu and Don Wunsch. *Clustering*, volume 10. John Wiley & Sons, 2008.

[8] Stephen Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological cybernetics*, 23(3):121–134, 1976.

[9] Stephen Grossberg. Adaptive pattern classification and universal recoding: Ii. feedback, expectation, olfaction, illusions. *Biological cybernetics*, 23(4):187–202, 1976.

[10] Gail A Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[11] Gail A Carpenter and Stephen Grossberg. Art 2: Self-organization of stable category recognition codes for analog input patterns. *Applied optics*, 26(23):4919–4930, 1987.

[12] Gail A Carpenter, Stephen Grossberg, and David B Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991.

[13] Gail A Carpenter, Stephen Grossberg, and John H Reynolds. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural networks*, 4(5):565–588, 1991.

[14] Gail Carpenter, Stephen Grossberg, Natalya Markuzon, John H Reynolds, And-David B Rosen, et al. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *Neural Networks, IEEE Transactions on*, 3(5):698–713, 1992.

[15] Gail A Carpenter. Distributed learning, recognition, and prediction by art and artmap neural networks. *Neural networks*, 10(8):1473–1494, 1997.

[16] Yongqiang Cao and Jianhong Wu. Projective art for clustering data sets in high dimensional spaces. *Neural networks*, 15(1):105–120, 2002.

[17] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is âĂIJnearest neighborâĂİ meaningful? In *Database TheoryâĂŤICDTâĂŹ99*, pages 217–235. Springer, 1999.

[18] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[19] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[20] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.

[21] Rui Xu and Donald C Wunsch II. Bartmap: A viable structure for biclustering. *Neural Networks*, 24(7):709–716, 2011.

[22] Chun Tang and Aidong Zhang. Interrelated two-way clustering and its application on gene expression data. *International Journal on Artificial Intelligence Tools*, 14(04):577–597, 2005.

[23] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[24] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. Semi-supervised learning. 2006.

[25] David J Miller and Hasan S Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in neural information processing systems*, pages 571–577, 1997.

[26] Tong Zhang and F Oles. The value of unlabeled data for classification problems. In *Proceedings of the Seventeenth International Conference on Machine Learning,(Langley, P., ed.)*, pages 1191–1198. Citeseer, 2000.

[27] V Vapnik and A Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

[28] Olivier Chapelle, Vikas Sindhwani, and Sathiya S Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

[29] Darren M Chitty. A data parallel approach to genetic programming using programmable graphics hardware. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1566–1573. ACM, 2007.

[30] Zhongwen Luo, Hongzhi Liu, and Xincai Wu. Artificial neural network computation on graphic process unit. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 1, pages 622–626. IEEE, 2005.

[31] JM Li, DL Wan, ZX Chi, and XP Hu. A parallel particle swarm optimization algorithm based on fine-grained model with gpu-accelerating. *Journal of Harbin Institute of Technology*, 38(12):2162–2166, 2006.

[32] Donald C Wunsch et al. Art properties of interest in engineering applications. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 3380–3383. IEEE, 2009.

[33] D Knuth. The art of computer programming 1: Fundamental algorithms 2: Seminumerical algorithms 3: Sorting and searching, 1968.

[34] Mario Martínez-Zarzuela, FJ Díaz Pernas, A Tejero de Pablos, M Antón Rodríguez, JF Díez Higuera, D Boto Giralda, and D González Ortega. Adaptative resonance theory fuzzy networks parallel computation using cuda. In *Bio-Inspired Systems: Computational and Ambient Intelligence*, pages 149–156. Springer, 2009.

[35] M Gorchetchnikov, H Ames, and M Versace. Simulating biologically realistic neural models on graphics process units. *Boston: ICCNS*, 2008.

[36] Ryan J Meuth. Gpus surpass computers at repetitive calculations. *Potentials, IEEE*, 26(6):12–23, 2007.

[37] Alain J Martin. The design of an asynchronous microprocessor. 1989.

[38] Donald Coolidge Wunsch et al. A optoelectronic learning machine. 1991.

[39] Guszti Bartfai. An art-based modular architecture for learning hierarchical clusterings. *Neurocomputing*, 13(1):31–45, 1996.

[40] Donald C Wunsch, Thomas P Caudell, C David Capps, Robert J Marks, R Aaron Falk, et al. An optoelectronic implementation of the adaptive resonance neural network. *Neural Networks, IEEE Transactions on*, 4(4):673–684, 1993.

[41] CUDA Nvidia. C programming guide version 3.2. *NVIDIA Corporation, Santa Clara, CA*, 2010.

[42] Robert HB Netzer and Barton P Miller. What are race conditions?: Some issues and formalizations. *ACM Letters on Programming Languages and Systems (LOPLAS)*, 1(1):74–88, 1992.

[43] Andrew Frank, Arthur Asuncion, et al. Uci machine learning repository. 2010.

[44] Julia Handl and Joshua Knowles. Improvements to the scalability of multiobjective clustering. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 3, pages 2372–2379. IEEE, 2005.

[45] T.C. Havens, J.C. Bezdek, C. Leckie, L.O. Hall, and M. Palaniswami. Fuzzy c-means algorithms for very large data. *Fuzzy Systems, IEEE Transactions on*, 20(6):1130 – 1146, dec. 2012.

[46] Richard Ernest Bellman. *Dynamic Programming*. Courier Dover Publications, 1957.

[47] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[48] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[49] S. Kesh and W. Raghupathi. Critical issues in bioinformatics and computing. *Perspect Health Inf Manag*, 1:9, 2004.

[50] S. Busygin, O. Prokopyev, and P. M. Pardalos. Biclustering in data mining. *Computers and Operations Research*, 35(9):2964–2987, 2008.

[51] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[52] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafe, A. Perez, and V. Robles. Machine learning in bioinformatics. *Brief. Bioinformatics*, 7(1):86–112, Mar 2006.

[53] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[54] J. Keller and M. Popescu. Soft computing in bioinformatics. In *Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on*, pages 3–3. IEEE, 2005.

[55] J. Zhang, J.J. Wang, and H. Yan. A neural-network approach for biclustering of gene expression data based on the plaid model. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 2, pages 1082–1087. IEEE, 2008.

[56] T. Reichhardt. It's sink or swim as a tidal wave of data approaches. *Nature*, 399(6736):517–520, June 1999.

[57] N. M. Luscombe, D. Greenbaum, and M. Gerstein. What is bioinformatics? A proposed definition and overview of the field. *Methods Inf Med*, 40(4):346–358, 2001.

[58] Rui Xu and Donald C Wunsch. Clustering algorithms in biomedical research: A review. *Biomedical Engineering, IEEE Reviews in*, 3:120–154, 2010.

[59] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, and J. M. Merrick. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269(5223):496–512, Jul 1995.

[60] Gaurav Pandey, Gowtham Atluri, Michael Steinbach, Chad L. Myers, and Vipin Kumar. An association analysis approach to biclustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 677–686, New York, NY, USA, 2009. ACM.

[61] Rui Xu, Steven Damelin, Boaz Nadler, and Donald C Wunsch. Clustering of high-dimensional gene expression data with feature filtering methods and diffusion maps. In *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, volume 1, pages 245–249. IEEE, 2008.

[62] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nat. Genet.*, 31(4):370–377, Aug 2002.

[63] G. Lim and J.C. Bezdek. Small targets in ladar images using fuzzy clustering. In *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, volume 1, pages 61–66, May 1998.

[64] H.M. Kim and B. Kosko. Neural fuzzy motion estimation and compensation. *Signal Processing, IEEE Transactions on*, 45(10):2515–2532, 1997.

[65] Z.G. Hou, M.M. Polycarpou, and H. He. Editorial to special issue: Neural networks for pattern recognition and data mining. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 12(7):613–614, 2008.

[66] P.J. Werbos. Neurocontrol and elastic fuzzy logic: Capabilities, concepts, and applications. *Industrial Electronics, IEEE Transactions on*, 40(2):170–180, 1993.

[67] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[68] Stephen Grossberg. Processing of expected and unexpected events during conditioning and attention: a psychophysiological theory. *Psychological review*, 89(5):529, 1982.

[69] Slavko Vasilic and Mladen Kezunovic. Fuzzy art neural network algorithm for classifying the power system faults. *Power Delivery, IEEE Transactions on*, 20(2):1306–1314, 2005.

[70] Gülşen Aydın Keskin, Sevinç İlhan, and Coşkun Özkan. The fuzzy art algorithm: A categorization method for supplier evaluation and selection. *Expert Systems with Applications*, 37(2):1235–1240, 2010.

[71] NC Suresh and S Kaparthi. Performance of fuzzy art neural network for group technology cell formation. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 32(7):1693–1713, 1994.

[72] Massimo Pacella, Quirico Semeraro, and Alfredo Anglani. Manufacturing quality control by means of a fuzzy art network trained on natural process data. *Engineering Applications of Artificial Intelligence*, 17(1):83–96, 2004.

[73] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[74] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[75] Raymond J Cho, Michael J Campbell, Elizabeth A Winzeler, Lars Steinmetz, Andrew Conway, Lisa Wodicka, Tyra G Wolfsberg, Andrei E Gabrielian, David Landsman, David J Lockhart, et al. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular cell*, 2(1):65–73, 1998.

[76] Saeed Tavazoie, Jason D Hughes, Michael J Campbell, Raymond J Cho, and George M Church. Systematic determination of genetic network architecture. *Nature genetics*, 22(3):281–285, 1999.

[77] Javed Khan, Jun S Wei, Markus Ringner, Lao H Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R Antonescu, Carsten Peterson, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679, 2001.

[78] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(suppl 1):S136–S144, 2002.

[79] Swarup Roy, Dhruba K Bhattacharyya, and Jugal K Kalita. Cobi: Pattern based co-regulated biclustering of gene expression data. *Pattern Recognition Letters*, 34(14):1669–1678, 2013.

[80] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[81] D. Steinley. Properties of the Hubert-Arabie adjusted Rand index. *Psychol Methods*, 9(3):386–396, Sep 2004.

[82] Vladimir Vapnik. *The nature of statistical learning theory*. Springer, 2000.

[83] Vladimir Vapnik. Statistical learning theory, 1998.

[84] Kristin Bennett, Ayhan Demiriz, et al. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, pages 368–374, 1999.

[85] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[86] Tijl De Bie and Nello Cristianini. Semi-supervised learning using semi-definite programming. *Semi-Supervised Learning. MIT Press: Cambridge*, 32, 2006.

[87] Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum margin clustering. In *NIPS*, volume 16, page 2, 2004.

[88] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, volume 1, pages 57–64, 2005.

[89] Glenn Fung and Olvi L Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15(1):29–44, 2001.

[90] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[91] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards.* PhD thesis, University of Cambridge, 1989.

[92] Michael Fairbank and Eduardo Alonso. Value-gradient learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[93] Danil V Prokhorov and Donald C Wunsch. Adaptive critic designs. *Neural Networks, IEEE Transactions on*, 8(5):997–1007, 1997.

[94] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[95] J K Udupa, S Samarasekera, and W a Barrett. Boundary detection via dynamic programming. In Richard A. Robb, editor, *Visualization in Biomedical Computing*, volume 1808, pages 33–39. International Society for Optics and Photonics, September 1992.

[96] John Seiffertt and Donald C Wunsch. *Unified Computational Intelligence for Complex Systems*. Springer, 2010.

[97] NG Brannon, John E Seiffertt, TJ Draelos, and Donald C Wunsch. Coordinated machine learning and decision support for situation awareness. *Neural Networks*, 22(3):316–325, 2009.

[98] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

[99] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive SVMs. *The Journal of Machine Learning Research*, 7:1687–1712, 2006.

[100] Annabella Astorino and Antonio Fuduli. Nonsmooth optimization techniques for semisupervised classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2135–2142, 2007.

[101] Vikas Sindhwani and S Sathiya Keerthi. Large scale semi-supervised linear SVMs. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 477–484. ACM, 2006.

[102] Olivier Chapelle, Mingmin Chi, and Alexander Zien. A continuation method for semi-supervised SVMs. In *Proceedings of the 23rd international conference on Machine learning*, pages 185–192. ACM, 2006.

[103] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[104] M. Fairbank and E. Alonso. The Local Optimality of Reinforcement Learning by Value Gradients, and its Relationship to Policy Gradient Learning. *ArXiv e-prints*, January 2011.

[105] Andrew G Barto. *Reinforcement learning: An introduction.* MIT Press, 1998.

[106] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[107] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. In *NIPS*, volume 17, pages 529–536, 2004.

[108] Ron Cole, Yeshwant Muthusamy, and Mark Fanty. The isolet spoken letter database. 1994.

[109] Martin Szummer and Tommi Jaakkola. Partially labeled classification with Markov random walks. In *NIPS*, pages 945–952, 2001.

[110] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[111] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). Technical report, Technical Report CUCS-005-96, 1996.

[112] GroupLens Research. Movielens data sets, 2015. [Online; accessed 29-May-2015].

[113] James A Cuff and Geoffrey J Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 34(4):508–519, 1999.

[114] Bill James. Baseball abstract 1983, 1983.

[115] Michael Lewis. *Moneyball: The art of winning an unfair game*. WW Norton & Company, 2004.

[116] Jahn K Hakes and Raymond D Sauer. An economic evaluation of the moneyball hypothesis. *The Journal of Economic Perspectives*, 20(3):173–185, 2006.

[117] Francis T Cullen, Andrew J Myer, and Edward J Latessa. Eight lessons from moneyball: The high cost of ignoring evidence-based corrections. *Victims and Offenders*, 4(2):197–213, 2009.

[118] J Scott Armstrong. Predicting job performance: The moneyball factor. *Foresight: The International Journal of Applied Forecasting*, 25:31–34, 2012.

[119] John Charles Bradbury. Does the baseball labor market properly value pitchers? *Journal of Sports Economics*, 8(6):616–632, 2007.

[120] Sara C Madeira, Miguel C Teixeira, Isabel Sa-Correia, and Arlindo L Oliveira. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 7(1):153–165, 2010.

[121] Guojun Li, Qin Ma, Haibao Tang, Andrew H Paterson, and Ying Xu. Qubic: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic acids research*, page gkp491, 2009.

[122] Peter A DiMaggio, Scott R McAllister, Christodoulos A Floudas, Xiao-Jiang Feng, Joshua D Rabinowitz, and Herschel A Rabitz. Biclustering via optimal re-ordering of data matrices in systems biology: rigorous methods and comparative studies. *BMC bioinformatics*, 9(1):458, 2008.

[123] Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl 1):S243–S252, 2001.

[124] Gad Getz, Hilah Gal, Itai Kela, Daniel A Notterman, and Eytan Domany. Coupled two-way clustering analysis of breast cancer and colon cancer gene expression data. *Bioinformatics*, 19(9):1079–1089, 2003.

[125] TM Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, volume 8, pages 77–88. World Scientific, 2003.

[126] Alain B Tchagang, Ahmed H Tewfik, Melissa S DeRycke, Keith M Skubitz, and Amy PN Skubitz. Early detection of ovarian cancer using group biomarkers. *Molecular Cancer Therapeutics*, 7(1):27–37, 2008.

[127] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

[128] C. Tang, L. Zhang, A. Zhang, and M. Ramanathan. Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. In *Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on*, pages 41–48. IEEE, 2001.

[129] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. U.S.A.*, 95(25):14863–14868, Dec 1998.

[130] Juan A Nepomuceno, Alicia Troncoso, Jesús S Aguilar-Ruiz, et al. Biclustering of gene expression data by correlation-based scatter search. *BioData mining*, 4(3), 2011.

[131] Baseball-Reference.com. Baseball reference, 2015.

[132] Basketball-Reference.com. Basketball-reference.com, 2015.

[133] Pro-Football-Reference.com. Pro football reference, 2015.

[134] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[135] Thomas M Cover and Peter E Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[136] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE, 1989.

VITA

Sejun Kim was born in August 23rd, 1981 in South Korea and spent his youth years in Raleigh, North Carolina. From Spring of 2000 to Summer of 2006, he attended Seoul National University in Seoul, Korea to earn a Bachelor of Science degree in Electrical Engineering. He received his Ph.D. in Computer Engineering from Missouri University of Science & Technology in May 2016. After his graduation, he started working for Intel Corporations at Hillsboro, Oregon.