IBM Research Report

Online Adaptive Decision Trees: Self-Organization

Jayanta Basak
IBM Research Division
4, Block-C, Institutional Area (ISID Campus),
Vasant Kunj, Phase - II,
New Delhi - 110070, India.
e-mail : bjayanta@in.ibm.com

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

Recently we have shown that decision trees can be trained in the online adaptive (OADT) mode (Basak, 2004, 2006) for supervised pattern classification and function approximation leading to better generalization score. The better generalization of OADT was mainly attributed to the unique property that the sum of its leaf node activations is always a constant. In this paper, we show that OADT can be used in the unsupervised mode leading to self-organization within a tree structure. We provide an architecture based on OADT, SOOADT (self-organized OADT), where we extend the basic adaptive tree with a code formation layer. The SOOADT adapts the tree parameters as well as the codes together in the online adaptive mode based on the minimization of an objective functional. We also discuss that the behavior of SOOADT is governed by a control parameter α. We do not incorporate any competitive learning in SOOADT, rather we utilize the property of a constant sum of leaf node activations in obtaining the self-organizing behavior of SOOADT. Experimentally, we demonstrate the behavior of the SOOADT in identifying different groups from the data in the adaptive mode, and subsequently show its capability in class discovery through unsupervised classification on real-life data.

1 Introduction

Recently we have shown that online adaptive decision trees (OADT) are capable of doing pattern classification and function approximation (Basak, 2004, 2006), and we have shown the effectiveness of OADT in handling the real-life data as well as complex classification tasks like multi-dimensional Gaussian parity problem. In handling the pattern classification and function approximation tasks, OADT has been trained in the supervised mode where each sample is associated with a predictor variable. In this paper, we discuss the unsupervised learning mechanism of the online adaptive decision trees where the samples do not have any predictor variable (a real-valued predictor as in the function approximation or a class label as in the pattern classification). In the unsupervised mode, OADT not only clusters or groups the data depending on their similarity (say Euclidian metric) but effectively self-organizes the tree by constructing the separating hyperplanes between the different groups in the online adaptive mode.

Pattern clustering (Jain & Dubes, 1988; Jain, Murty, & Flynn, 1999) is a well-studied topic in the pattern recognition literature where samples are grouped into different clusters based on certain self-similarity measure. Depending on different criteria such as data representation, similarity/dissimilarity measure, interpretation, domain knowledge and modalility (e.g., incremental/batch mode), different clustering algorithms have been developed – a comprehensive discussion on which can be found in (Jain et al., 1999). Decision trees have also been used for clustering (Liu, Xia, & Yu, 2000; Basak & Krishnapuram, 2005). As an example, in (Liu et al., 2000), the process of construction of unsupervised decision tree was mapped onto that of a supervised decision tree where samples were artificially injected such that empty regions contain sparse injected samples. The decision tree was then constructed by discriminating between the original samples and the artificially injected samples. At every level of the tree, the number of the injected samples was controlled depending on the number of actual data samples available

at that node. In (Basak & Krishnapuram, 2005), on the other hand, the dataset was iteratively partitioned at each level of the tree by measuring the entropy from the histogram of the data distribution. However, in these algorithms, a decision tree is constructed based only on the local splitting criteria at each node as in the other supervised decision tree based algorithms (Duda, Hart, & Stork, 2001; Durkin, 1992; Fayyad & Irani, 1992; Friedman, 1991; Breiman, Friedman, Olshen, & Stone, 1983; Quinlan, 1993, 1996; Brodley & Utgoff, 1995) . Moreover, the clustering algorithms in (Liu et al., 2000; Basak & Krishnapuram, 2005) are not adaptive, i.e., it does not address the changeability of the decision hyperplanes in the tree if the data distribution changes, and operate in the batch mode. With a given tree topology, a stochastic approximation of the data distribution through deterministic annealing was performed in (Held & Buhmann, 1998) to obtain groups of data samples at the leaf nodes of the tree.

A closely related field to pattern clustering is self-organization. Self-organization in general refers to the automatic organization of a system with increased complexity exhibiting certain emergent behavior. Since neural networks (Haykin, 1999) inherently exhibit online adaptive behavior, self-organizing properties of neural assemblies have been widely studied. A widely different literature is available on self-organization in the domain of neural networks in various contexts such as topographic map formation, code formation, structure formation and class discovery; a few representaives are available in (Malsburg, 1973; Kohonen, 1988; Grossberg, 1987; Carpenter & Grossberg, 1987, 1987; Amari, 1972, 1977; Linsker, 1986; Mao & Jain, 1996). Cooperative and competitive learning meachnisms are often employed in various forms in order to infuse the emergent characteristics in the network from the observed samples.

Possibly one of the classic approach in the literature of self-organization in neural network domain is the self organized map (Kohonen, 1988) where the data samples are mapped in the online mode onto an assembly of neurons. The neurons in this assembly organize themselves in such a way that the topographical properties of the data samples in the original space is mapped onto the reduced physical dimension of the assembly. The samples which are close to each other fire neurons which are physically close and vice-versa, where firing of a neuron is governed by the corresponding weight vector of that neuron from the input space. The physical organization of the neurons in the assembly reveal the intrinsic characteristic of the data in the original higher dimensional space. Self-organizing maps (SOM) were derived based on the neighborhood interaction between the neurons such that a firing neuron in the assembly induce similar weight vectors for neurons in its neighborhood. Self-organizing map does not explicitly induce any partitioning on the dataset, rather produce a physical interpretation of the topographic arrangement between the samples.

The process of self-organizing code formation is governed by the stability and plasticity dilemma (Grossberg, 1987). If a neural assembly becomes highly stable then it is not capable of adapting to new data distribution i.e., the algorithm loses plasticity. On the other hand, if the map formation is plastic enough then it gets perturbed by small variations in the data distribution and therefore loses the stability. An analogy of stability-plasticity dilemma can be drawn from the bias-variance dilemma (Duda et al., 2001) in the supervised learning domain. If a learning machine has a high bias then it loses varaince and is not able to capture finer details in the model fitting whereas with a large variance, the machine is highly perturbed by small variations (noise) in the input data. In adaptive resonance theory (ART) architectures (Carpenter & Grossberg,

1987, 1987), the stability-plasticity dilemma is addressed and handled explicitly by the 2/3rd rule where codes are formed adaptively from the input samples.

In this paper, we show that online adaptive decision trees can also be used for grouping and code formation from the input data samples in an unsupervised manner. In (Basak, 2006), we have claimed and experimentally shown that OADT has an inherent regularizarion capability in the supervised domain such that it reduces the variance by incorporating certain bias in the form of a constant collective activation of the leaf nodes. We exploit the same property of OADT (a constant sum of leaf activations) in the unsupervised domain and experimentally observed the stability of OADT in grouping the data samples in the online mode. We do not incorporate any competitive learning among the leaf nodes explicitly as it is used in many self-organizing networks (Kohonen, 1988; Carpenter & Grossberg, 1987, 1987). Contrary to SOM (Kohonen, 1988), OADT lends itself to a self-organizing behavior within a tree structure; and the decision hyperplanes are organized according to the input data distribution.

We first briefly describe the supervised counterpart of OADT in Section 2. We then show how OADT can be used for the unsupervised learning in the online adaptive mode in Section 3. In Section 4, we demonstrate the effectiveness of OADT with certain examples of artificial and real-life data and then validate its capability for class discovery with real-life datasets. Finally we discuss and conclude the paper in Section 5.

2 Description of Supervised OADT

We describe the architecture and learning of OADT in the supervised mode in this section. In (Basak, 2006), we described OADT as ExOADT (an acronym for Extended OADT) in order to distinguish it from the previous model (Basak, 2004). Here we describe ExOADT as OADT considering OADT as the generic name for this kind of models. ExOADT consists of two major components as illustrated in Figure 1 namely, an adaptive tree and a regression layer. The adaptive tree is a complete binary tree of a certain depth l. Each non-terminal node i has a local decision hyperplane characterized by a vector $(\mathbf{w}_i, \theta_i)$ with $\|\mathbf{w}_i\| = 1$ such that $(\mathbf{w}_i, \theta_i)$ together has n free variables for an n-dimensional input space. The leaf nodes of the adaptive tree is connected to the output nodes through a regression layer characterized by a weight matrix β.

Each nonterminal node of the adaptive tree receives the same input pattern $\mathbf{x} = [x_1, x_2, \cdots, x_n]$, and the activation from its parent node. The root node receives only the input pattern $\mathbf{x}$. Each nonterminal node produces two outputs, one for the left child and the other for the right child. We number the nodes in the breadth-first order such that the left child of node i is indexed as $2i$ and the right one as $2i + 1$. The activation of two child nodes of a non-terminal node i are given as

$$\begin{aligned} u_{2i} &= u_i.f(\mathbf{w}_i^T\mathbf{x} + \theta_i) \\ u_{2i+1} &= u_i.f(-(\mathbf{w}_i^T\mathbf{x} + \theta_i)) \end{aligned} \tag{1}$$

where u_i is the activation of node i, and $f(.)$ represents the local soft partitioning function governed by the parameter vector $(\mathbf{w}_i, \theta_i)$. We choose $f(.)$ as a sigmoidal function. The activation

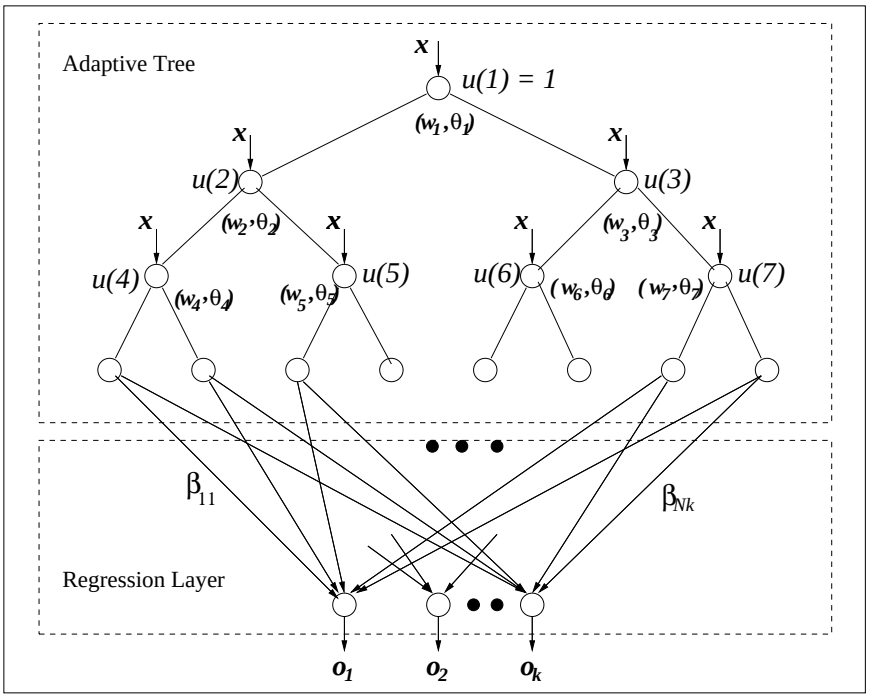


Figure 1: Structure of the supervised OADT for pattern classification and function approximation. It has two components namely, adaptive tree and the regression layer. Each non-terminal node in the adaptive tree receives the input pattern x as input, and also the activation from its parent. The root node activation is always unity. Each non-terminal node stores hyperplane parameterized by (w, θ). The leaf activations are transformed by a regression matrix β as stored in the regression layer to produce the output $o = [o_1, o_2, \cdots, o_k]$ with k output nodes.

of the root node, indexed by 1, is always set to unity. The activation of a leaf node p can therefore be derived as

$$u_p = \prod_{i \in P_p} f\left(lr(i,p)(\mathbf{w}_i.\mathbf{x} + \theta_i)\right). \tag{2}$$

where P_p is the path from the root node to the leaf node p, and $lr(.)$ is an indicator function indicating whether the leaf node p is on the right or left path of node i such that

$$lr(i,p) = \begin{cases} 1 & \text{if } p \text{ is on the left path of } i \\ -1 & \text{if } p \text{ is on the right path of } i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Considering the property of sigmoidal function $f(.)$ that $f(-v) = 1 - f(v)$, we observe the sum of activation of two siblings is always equal to that of their parent. Extending this reasoning across the layers of the tree in a bottom-up order, it is formally shown in (Basak, 2004) that

$$\sum_{j \in \Omega} u_j = 1 \tag{4}$$

provided that the root node activation is always unity. Here Ω represents the set of leaf nodes. The very fact that the sum of leaf node activations of the adaptive tree is always unity irrespective of the input pattern enables OADT of certain inherent regularization capability in the supervised mode. We have chosen $f(.)$ as

$$f(v) = \frac{1}{1 + \exp(-mv)} \tag{5}$$

where m determines the steepness of the function $f(.)$. In (Basak, 2004), we have shown that

$$m \geq \frac{1}{\delta} \log l/\epsilon \tag{6}$$

where δ is a margin between a point (pattern) and the closest decision hyperplane, ϵ is a constant such that the minimum activation of a maximally activated node should be no less that $1 - \epsilon$. For example, if we choose $\epsilon = 0.1$ and $\delta = 0.1$ then $m \geq 10 \log(10l)$.

The leaf node activations are transformed through the regression layer (charaterized by weight matrix β) to produce the output activation such that the activation of the ouput node k is given as

$$o_k = g(\sum_{j \in \Omega} \beta_{jk} u_j) \tag{7}$$

where $g(.)$ is either sigmoidal in nature (not necessarily the same as $f(.)$) for pattern classification or linear in nature for function approximation. The number of output nodes is equal to the number of pattern classes in the case of classification.

For classification, an objective functional in the form

$$E(\mathbf{x}) = \frac{1}{2} \sum_k (y_k(\mathbf{x}) - o_k(\mathbf{x}))^2 \tag{8}$$

was considered (Basak, 2006) to measure the loss where y represents the desired class labels such that $y_k \in \{0, 1\}$ depending on whether the pattern belongs to k-th class or not. For function

approximation, y is replaced by the desired values of the predictor variable. With the objective functional in Equation (8), steepest gradient descent with line search (Friedman, 2001) has been performed to obtain the locally optimal separating hyperplanes in the nonterminal nodes and the regression matrix β.

3 Self Organization in OADT

Here we show how OADT can be used for the purpose of online adaptive grouping of the input data samples by adapting the decision hyperplanes of the nonterminal nodes in a self-organized manner. In the supervised learning, the output of the regression layer represents the respective pattern classes and the final output is obtained by transforming the leaf activations through the regression layer. In order to perform grouping on the input data samples, we deviate from the structure used in the supervised mode. In sequel, we call SOOADT (an acronym for self-organized OADT) to distinguish it from our original OADT used for classification.

3.1 Structure and Activation of SOOADT

Structurally, SOOADT is very similar to OADT (or ExOADT as in (Basak, 2006)) except that the regression layer is replaced by a "code formation layer" characterized by a weight matrix β as shown in Figure 2. The adaptive tree component of SOOADT is exactly the same as that in OADT such that each nonterminal node has two child nodes. As in the supervised OADT, in SOOADT also, each nonterminal node of the adaptive tree receives the same instance of the input vector $\mathbf{x}$. The activation of each nonterminal node is divided into two child nodes as in Equation (1) subject to a hyperplane $(\mathbf{w}, \theta)$ with a constraint $\|\mathbf{w}\| = 1$, and the sigmoidal activation function as in Equation (5). Since the activation of the leaf nodes in SOOADT is identical to that in the OADT, the sum of activation of the leaf nodes of the adaptive tree is always unity provided that the root node activation is unity.

In the SOOADT, the clusters or groups of the data samples are represented by the leaf nodes of the adaptive tree as opposed to the supervised OADT where the pattern classes are represented by the output of the regression layer. Therefore for an SOOADT of depth l, at most 2^l groups can be formed. Later we show that depending on certain parameter of the adapting process, the number of groups can change with the same data distribution. Corresponding to each leaf node j, there exist a code β_j in the code formation layer. Unlike the regression layer in supervised OADT, the code formation layer in SOOADT receives the same input pattern $\mathbf{x}$ (there is no desired output vector) and measures the deviation of the input from the stored code β_j corresponding to a leaf node j. The process of measuring the deviation of a stored code from the input pattern is similar to that in SOM (Kohonen, 1988); however, in SOM, the activation of a neuron depends on the distance of the input samples from the stored code. In SOOADT, the activation of a leaf node does not depend on the deviation of the corresponding stored code in β_j from the input $\mathbf{x}$, rather the activation depends on the parameters stored in the adaptive tree. Thus the code

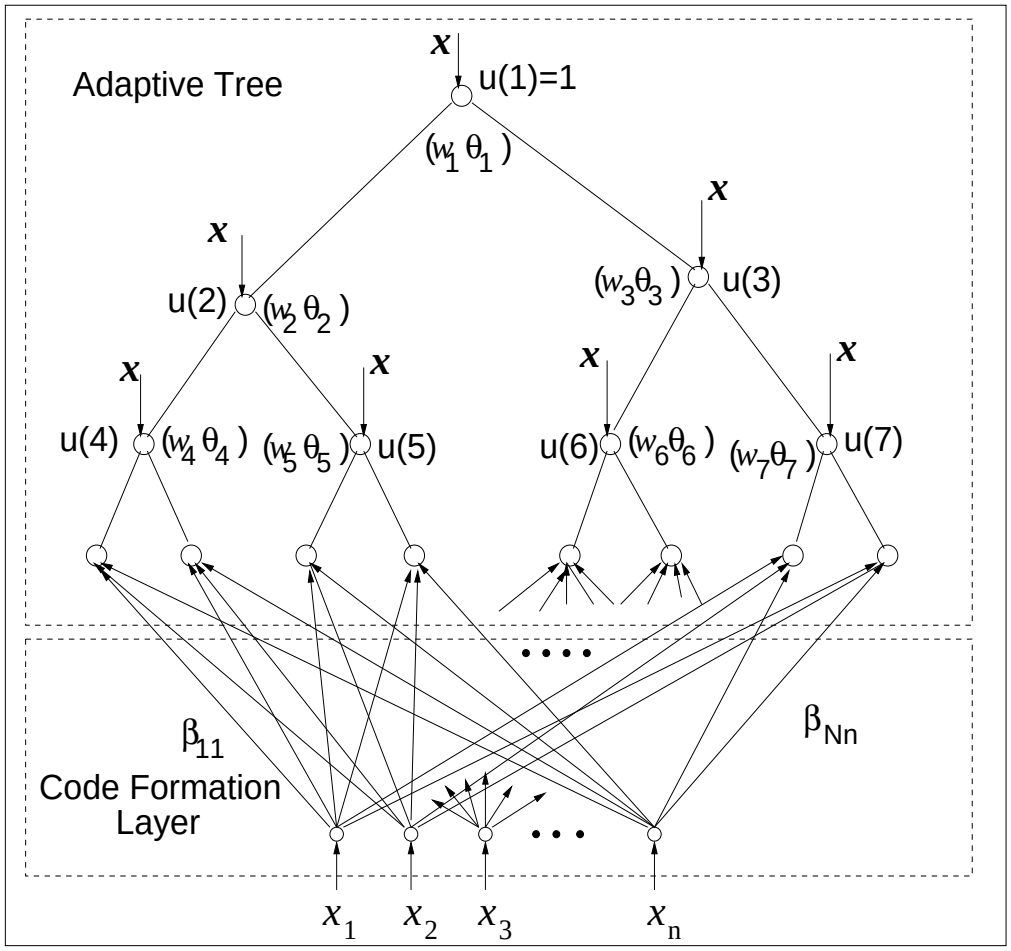


Figure 2: Structure of the self-organized OADT (SOOADT). It has two components namely, adaptive tree and a code formation layer. The adaptive tree is the same as that in the supervised OADT. The code formation layer receives the input pattern x as input and adapts the stored code vectors β during the process of self-organization along with the adaptation of the adaptive tree.

formation layer provides SOOADT the capability of adapting the parameters of the adaptive tree component.

Interestingly, we observe that the leaf node activations can also be interpreted in terms of the degrees of membership of a pattern $\mathbf{x}$ in different clusters as represented by the different leaf nodes. This is conceptually analogous to that in soft c-means (Bezdek, 1981) algorithm where each pattern is attributed by a set of membership values such that each membership value represents the degree of membership to the corresponding cluster. In soft c-means, the number of clusters is always equal to c, and a constraint is explicitly imposed to make the sum of membership values of a pattern to different clusters equal to unity. In SOOADT, on the other hand, the number of groups is not necessarily a constant (its upper bound is 2^l where l is the depth of the tree). The very mechanism of leaf node activation in the adaptive tree of the SOOADT imposes the constraint that the sum of leaf node activations is always unity. We now elaborate the adaptation mechanism in SOOADT.

3.2 Adaptation in SOOADT

In order to formulate the adaptation mechanism in SOOADT, we first define objective functional, and then minimize the objective functional using steepest gradient descent. Note that, in the literature of pattern clustering also, often the clusters are formed by minimizing certain explicit objective functional. On the other hand, in SOM or the ART-like architectures, the codes are formed not by explicitly minimizing any objective functional, rather they are formed by locally adapting the maximally activated neurons. In defining the objective functional of SOOADT, we consider the discrepancy between the input pattern and the existing codes in β and the activation of the leaf nodes together. Formally, the objective functional $E(x)$ for an input pattern $\mathbf{x}$ is given by

$$E(x) = \frac{1}{2} \sum_{j \in \Omega} \|x - \beta_j\|^2 u_j^{1/\alpha} \tag{9}$$

where α is a constant which acts as a control parameter for SOOADT.

We adapt the parameters of the SOOADT (the hyperplanes of the adaptive tree (w, θ) and the stored codes β in the code formation layer together) in order to minimize the objective functional E for a given α. We minimize E by steepest gradient descent such that

$$\begin{aligned} \Delta\beta_j &= -\eta_{opt} \frac{\partial E}{\partial \beta_j} \\ \Delta\mathbf{w_i} &= -\eta_{opt} \frac{\partial E}{\partial \mathbf{w_i}} \\ \Delta\theta_i &= -\eta_{opt} \frac{\partial E}{\partial \theta_i} \end{aligned} \tag{10}$$

where η_{opt} is a certain optimal learning rate parameter decided in the online mode. Since SOOADT is an online adaptive learning tool, we do not make the learning rate dependent on the number of iterations which is usually done in the case of SOM learning [1](Kohonen, 1988).

[1]subject to $\sum_t \eta_t \to \infty$, and $\sum_t \eta_t^2 < \infty$

Evaluating the partials in Equation (10), we obtain

$$\begin{aligned} \Delta\beta_j &= \eta_{opt} u_j^{1/\alpha}(\mathbf{x} - \beta_j) \\ \Delta\mathbf{w_i} &= -\eta_{opt} q_i \mathbf{x} \\ \Delta\theta_i &= -\eta_{opt}\lambda q_i \end{aligned} \tag{11}$$

where q_i is expressed as

$$q_i = \frac{m}{2\alpha}\left[\sum_{j\in\Omega} \|\mathbf{x} - \beta_j\|^2 (u_j)^{1/\alpha}(1 - v_{ij}) lr(i,j)\right] \tag{12}$$

Each nonterminal node i locally computes v_{ij} which is given as

$$v_{ij} = f((\mathbf{w}_i^T\mathbf{x} + \theta_i) lr(i,j)) \tag{13}$$

Since $(\mathbf{w}, \theta)$ at each nonterminal node has n free variables together (n is the input dimensionality) subject to the condition that $\|\mathbf{w}\| = 1$, the updating of $\mathbf{w}$ is Equation (11) is restricted to the normalization condition. The parameters $\mathbf{w}$ is updated such a way that $\Delta\mathbf{w}$ lies in the hyperplance normal to $\mathbf{w}$ which ensures that $\mathbf{w}.\Delta\mathbf{w} = 0$, i.e., the constraint $\|\mathbf{w}\| = 1$ is satisfied for small changes after updating. For small η, it can be shown (Basak, 2004) that the updating rule for $\mathbf{w}$ can be expressed as,

$$\Delta\mathbf{w_i} = -\eta_{opt} q_i (\mathbf{I} - \mathbf{w_i}\mathbf{w_i}^T)\mathbf{x} \tag{14}$$

where $\mathbf{I}$ is the identity matrix. In Equation (11), the parameter λ represents the scaling of the learning rate parameter with respect to θ. We normalize the input variables such that $\mathbf{x}$ is bounded in $[-1, +1]^n$, and we select $\lambda = 1$. Note that if the input is not normalized then we can select a different value for λ.

For every input sample $\mathbf{x}$ at every iteration, we select η_{opt} using the line search method (also used by Friedman in (Friedman, 2001)) such that the change in E to $E + \Delta E$ is linearly extrapolated to zero. Equating the first order approximation of ΔE (line search) to $-E$, we obtain,

$$\eta_{opt} = \frac{E}{\sum_{j\in\Omega}(u_j)^{2/\alpha} + \sum_{i\in N} q_i^2 \left(\lambda + \|\mathbf{x}\|^2 - (\mathbf{w_i}^T\mathbf{x})^2\right)} \tag{15}$$

In Equation (15), we observe that for very compact or point clusters, η can become very large near optima. In order to stabilize the behavior of the adaptive tree, we modify the learning rate as

$$\eta_{opt} = \frac{E}{\gamma + \sum_{j\in\Omega}(u_j)^{2/\alpha} + \sum_{i\in N} q_i^2 \left(\lambda + \|\mathbf{x}\|^2 - (\mathbf{w_i}^T\mathbf{x})^2\right)} \tag{16}$$

with a constant γ. In our model, we chose $\gamma = 1$.

From the learning rules, we observe that the codes represented by β are updated such that they approximate the input pattern subject to the activation of the leaf nodes. This behavior is very similar to that in SOM, where the codes are explicitly modified to approximate the input pattern in a similar manner over a neighborhood. In SOOADT, there is no concept of physical neighborhood function as in SOM, rather the updating of the codes β is affected by the

activation of the leaf nodes. The maximally activated leaf node attracts the corresponding β most towards the input pattern. In the adaptive process, if the input patterns are drawn from a stable distribution then η_{opt} decreases with time (since E decreases) and therefore, the updating rules reveal that the corresponding codes in β are also less affected by the presence of input patterns. This is analogous to SOM where the learning rate parameter is explicitly decreased with the number of iterations subject to $\sum \eta_t \rightarrow \infty$ and $\sum \eta_t^2 < \infty$. In SOOADT on the other hand, η is not explicitly decreased by any such learning schedule. Rather, depending on the input distribution, η_{opt} adjusts itself.

It is also interesting to observe the behavior of the decision hyperplanes in the nonterminal nodes of the adaptive tree. If there is a close match between the input pattern $\mathbf{x}$ and the stored code β corresponding to the maximally activated leaf node, then the decision hyperplanes are not perturbed much since $\|\mathbf{x} - \beta\|$ becomes very small. On the other hand, if there is a mismatch then the movement of the decision hyperplance is decided by the stored function $lr(i, j)$. If the maximally activated leaf node j is on the left path of the nonterminal node i then $lr(i, j) = 1$, i.e., $\mathbf{w_i}$ is moved such a way that $\mathbf{\Delta w_i}$ has an opposite sign of $\mathbf{x}$ subject to being in the normal hyperplane of $\mathbf{w_i}$. Since the activated leaf node is on the left path i.e., $\mathbf{x}$ is on the left half of the decision hyperplane, the decision hyperplane effectively moves opposite to $\mathbf{x}$ to increase the activation of the activated leaf node. The magnitude of the shift depends on the activation value of the leaf, and the discrepancy between the corresponding stored code and the input pattern, Similar reasoning is valid if the maximally activated leaf node j is on the right path of the nonterminal node i. We also observe that as we move towards the root node, the decision hyperplanes are perturbed by the cumulative effect of the entire subtree under the nonterminal node (if j is not under the subtree of i then $lr(i, j) = 0$). We also notice that the updating rule for decision hyperplanes are independent of each other. For example, a leaf node j can be on the left path of a nonterminal node i with $lr(i, j) = 1$, however it can be on the rigth path of its left child. Therefore, the leaf nodes affect different nonterminal nodes differently at different levels depending on the stored function $lr(i, j)$.

3.3 Choice of α

Interstingly, the objective functional in Equation (9) looks similar to that in soft c-means (Bezdek & Castelaz, 1977; Bezdek, 1981) where β corresponds to the cluster centers and u corresponds to the membership values. However, α in soft c-means is chosen in $[0, 1]$ and usually $\alpha < 1$. In the limit $\alpha = 1$, it becomes the hard c-means algorithm. In soft c-means α is chosen less than unity in order to make the cluster centers more dependent on the proximity points, and the clusters are represented by the cluster centers. In SOOADT, on the other hand, leaf node activations do not directly depend on the computed codes in the code formation layer. Rather, the decision hyperplanes stored in the intermediate nodes of the adaptive tree determine the leaf node activations. The code formation layer, parameterized by the β matrix, stores the respective codes corresponding to the groups of samples identified by the adaptive tree, each leaf node representing one group of samples respectively. The decision hyperplanes in the adaptive tree and the codes in the code formation layer are adapted together to minimize the objective

functional (Equation (9)).

In SOOADT, we usually choose $\alpha > 1$. Since each leaf node activation $u \leq 1$, a choice of $\alpha > 1$ makes $u^{1/\alpha} > u$ if $u < 1$, and it remains the same if $u = 1$. From Equation (4), we observe that with any given input $\mathbf{x}$, since $\sum_{j \in \Omega} u_j = 1$. In our parametric setting, as stated in Equation (6) (Basak, 2004), the minimum activation of the maximally activated node is at least $1 - \epsilon$. Therefore, for a choice of $\alpha < 1$, the difference between the activation of the maximally activated leaf and that of the next highest activated leaf increases. As a result, only the maximally activated leaf node affect the adaptation process. As we discussed in the previous section (learning rules) that the decision hyperplanes of the adaptive tree are adjusted such that an activated leaf gets more activation during the adaptation process. Therefore, for a choice of $\alpha < 1$, the SOOADT parameters are adapted in such a way that the parameters for only the maximally activated leaf node are adapted, and other leafs do not play any effective role. In other words, the initial configuration of the network is not adapted much and the SOOADT loses its adaptivity property. For a choice of $\alpha > 1$, on the other hand, the effective difference between the maximum leaf activation and second maximum leaf activation decreases in the objective functional E, and therefore the samples are spread across different groups represented by different leaf nodes. In the limiting condition, as $\alpha \to 0$, $E \to 0$, and the SOOADT does not adapt at all and remains in the initial configuration. On the other hand, for $\alpha \to \infty$, there is no role of individual leaf node activations in E, and the parameters of the adaptive tree are adapted in such a way that all samples are merged into one group. We can also observe the same behavior from the combination of adaptation rules and the learning rate (Equation (12) and (15)), where with the increase in the value of α, the magnitude of the changes in parameter values increase, and this enables SOOADT to become more adaptive. With a choice of $\alpha = 1$, the leaf nodes contribute to the objective functional E with their original activation values. Therefore, in order to make the SOOADT more adaptive, we choose $\alpha > 1$. We observe that for a choice of $\alpha < 1$, fragmented regions are formed and the SOOADT is not able to merge them.

We explain the effect of α empirically with a very simple example. Let there be only two different samples $\mathbf{x_1}$ and $\mathbf{x_2}$, and the SOOADT has only two leaf nodes. Without loss of generality, let us represent the activation of leaf nodes as u_1 and u_2. Let the distance between the two samples be $d = \|x_1 - x_2\|$. Ideally, there are two possible cases.

Case I: The sample $\mathbf{x_1}$ forms one cluster and $\mathbf{x_2}$ forms the second one such that codes formed are $\beta_1 = x_1$ and $\beta_2 = x_2$. The SOOADT forms a separating hyperplane exactly normal to the vector $x_1 - x_2$ and equidistant from the samples. In such case, when x_1 is presented to SOOADT then $u_1(x_1) = 1/(1 + \exp(-md/2))$ and $u_2(x_1) = 1/(1 + \exp(md/2))$. In such a scenario,

$$E(x_1) = \|x_1 - \beta_1\|^2 u_1^{1/\alpha} + \|x_1 - \beta_2\|^2 u_2^{1/\alpha} \tag{17}$$

i.e.,

$$E(x_1) = \frac{d^2}{(1 + \exp(md/2))^{(1/\alpha)}} \tag{18}$$

We get exactly same value for $E(x_2)$ due to symmetry and therefore the total loss is

$$E_1 = E(x_1) + E(x_2) = \frac{2d^2}{(1 + \exp(md/2))^{(1/\alpha)}} \tag{19}$$

Case II: Both the samples $\mathbf{x_1}$ and $\mathbf{x_2}$ form one cluster and the code β is exactly placed in the middle such that $\beta = (x_1 + x_2)/2$. Since both samples are in the same cluster, there is no separating hyperplane between them (ideally it is at an infinite distance from x_1 and x_2), and only one leaf node is activated such that $u_1 = 1$ and $u_2 = 0$ for both x_1 and x_2. In such a scenraio,

$$E(x_1) = \|x_1 - \beta\|^2 u_1^{1/\alpha} + \|x_1 - \beta\|^2 u_2^{1/\alpha} \tag{20}$$

i.e.,

$$E(x_1) = d^2/4 \tag{21}$$

Since we obtain exactly the same value for $E(x_2)$, the total loss is

$$E_2 = E(x_1) + E(x_2) = \frac{d^2}{2} \tag{22}$$

From Equations (19) and and (22), we obtain

$$\left(\frac{E_1}{E_2}\right)^{\alpha} = \frac{4^{\alpha}}{1 + \exp(md/2)} \tag{23}$$

From Equation (23), we observe that there exists an

$$\alpha_0 = \frac{1}{2}\log_2(1 + \exp(md/2)) \tag{24}$$

such that for $\alpha \geq \alpha_0$, $E_1 \geq E_2$. In other words, since we minimize the objective functional, the scenario in *case II* is prefered over that in *case I* if we choose $\alpha > \alpha_0$, i.e., both samples are clustered in the same group. On the other hand for a choice of $\alpha < \alpha_0$, the samples are preferred to represent two different groups (since in that case $E_1 < E_2$).

As discussed in Section 2 (Equation (6)), with a setting $d = \delta$, we obtain

$$\frac{1}{1 + \exp(-md)} = 1 - \epsilon \tag{25}$$

With algebraic manipulation, we obtain

$$\exp(md/2) = \sqrt{1/\epsilon - 1} \tag{26}$$

Therefore from Equations (24) and (26), we have

$$\alpha_0 = \frac{1}{2}\log_2(1 + \sqrt{1/\epsilon - 1}) \tag{27}$$

As in Section 2 (Equation (6)), letting $\epsilon = 0.1$, we obtain $\alpha_0 = 1$.

The same reasoning is valid for more than two samples with multiple levels in the adaptive tree. In the case of multiple levels, we assume that two closest points are separated only at the last level of the tree. Therefore, for a choice of $\alpha < 1$, the adaptive tree can partition single small clusters into multiple ones. In short, for a choice of large α, the number of groups formed by the adaptive tree decreases and large chunks of data are grouped together. On the other hand, if α is decreased then the number of groups increases. In soft c-means, on the contrary, the number of

clusters is always constant and equal to c. The parameter α is used to control the radius of each soft cluster during the computation of the centroid of each cluster; and the membership values of the samples to different clusters depend on the computed cluster centers. Therefore, although the objective functional in Equation (9) looks very similar to that in soft c-means, conceptually they are totally different.

In the code forming self-organizing networks (Carpenter & Grossberg, 1987, 1987; Grossberg, 1987), the adaptation process is mostly guided by the stability-plasticity dilemma. If the network becomes very plastic and adapts to new samples often then the existing codes becomes unstable and new codes are formed. In such netwroks, the activations of the neurons are explicitly determined by the stored codes. In the self-organizing map (SOM) network (Kohonen, 1988) also, the activations of the neurons explicitly depend on the stored codes, and the similar codes are formed over a neighboorhood region. In the SOOADT, on the other hand, the leaf node activations depend on the stored hyperplanes in the intermediate nodes of the adaptive tree. If the SOOADT loses plasticity for a small value of α then the codes as well as the hyperplanes are not adapted. On the other hand, for a large value of α, the regions formed by the leaf nodes are merged together to form bigger clusters in the input space. Since we minimize the objective functional in the SOOADT with a given structure such that the sum of the leaf node activations is always unity, the SOOADT inherently exhibits stability property locally as the objective functional convereges to a local minima.

4 Experimental Results

We implemented the SOOADT in the MATLAB 7.0 environment. We experimented the performence of SOOADT with both synthetic and real-life data sets. We first describe the parameter setting of SOOADT and then illustrate the behavior of SOOADT on certain toy examples in two-dimensional space. We also illustrate the behavior of SOOADT on certain real-life data projected onto two-dimensional space. We then validate the performance of SOOADT in class discovery on real-life data sets. We use the same modality in obtaining the classification performance as it is often used in SOM. We assign each leaf node a class label depending on the majority of the samples and then compute the overall classification score on the training data set. We then also demonstrate the performance of SOOADT with one example for classifying leukemia samples from the gene expression data in the unsupervised mode.

4.1 Protocol

We trained SOOADT in the online mode. The entire batch of samples is repeatedly presented in the online mode to SOOADT, and we call the number of times the batch of samples presented to an SOOADT as the number of epochs. If the data density is high or the dataset has a set of compact clusters then we observe that the SOOADT takes less number of epochs to converge; and for relatively low data density or in the presence of sparse clusters, SOOADT takes a larger

number of epochs. On an average, we observed that SOOADT converges near its local optimal solution within 50 epochs and sometimes even less, although the required number of epochs increases with the depth of the tree. Subsequently, we report all results for 200 epochs.

We normalize all input patterns such that each component of $\mathbf{x}$ lies in the range $[-1, +1]$. We normalize each component of $\mathbf{x}$ (say, x_i) as

$$\hat{x_i} = \frac{2x_i - (x_i^{max} + x_i^{min})}{x_i^{max} - x_i^{min}} \tag{28}$$

where x_i^{max} and x_i^{min} are the maximum and minimum values of x_i over all observations respectively. Note that, in this normalization, we do not take into account of the outliers. Data outliers can badly influence the variables in such normalization. Instead of linear scaling, use of certain non-linear scaling or certain more robust scaling such as that based on inter-quartile range may further improve the performance of our model. However, in this paper, we preserve the exact distribution of the samples and experiment with the capability of SOOADT in extracting the groups of samples in the online adaptive mode.

The performance of SOOADT depends on the slope (parameter m) of the sigmoidal function (Equation (5)). In general, we observed that for a given control parameter α, the number of decision hyperplanes increases with the decrease in the slope (this is also validated in Equation (24) where α_0 decreases with the decrease in m). This is due to the fact that if we decrease the slope then more than one leaf node gets significant activation to attract the codes (β) in the code formation layer and thereby the number of clusters increases. On the other hand, with the increase in the slope, only a few (in most of the cases only one) leaf nodes are significantly activated to influence the decision hyperplanes and therefore the number of clusters decreases. In general, for a small m, a group of leaf nodes together represents a compact cluster. As stated in Equation (6), for a given depth l of the adaptive tree, we can fix the slope as $m \geq \frac{1}{\delta}\log(l/\epsilon)$, where δ is a margin between a point (pattern) and the closest decision hyperplane, ϵ is a constant such that the minimum activation of a maximally activated node should be no less that $1 - \epsilon$. Experimentally, we fixed $\delta = 0.2$ for an $\epsilon = 0.1$. In other words, we fix the slope as $m = 5\log(10l)$ for given depth l of the tree. Note that, the value of δ that we have chosen is valid for the normalized input space that we use. If the data is not normalized and the input range is higher then a larger value of δ can also be selected. We also observed the behavior of SOOADT with respect to the control parameter α. As we increase α, the number of clusters extracted by SOOADT decreases and vice-versa. In the next section, we illustrate certain behavior of SOOADT for different choices of α.

In the adaptation rule (Equation (11)) as stated before, we always select λ as unity since we operate in the normalized input space. We also select the constant $\gamma = 1$ in computing the learning rate irrespective of any other parameter. Note that in the supervised version of the OADT also, we use the same value for γ in obtaining the learning rate. We initialize β to zero. We also initialize θ at each nonterminal node to zero. We initialize the weight vectors $\mathbf{w}$ randomly with a Gaussian distribution and then normalize each weight vector such that $\|\mathbf{w}\| = 1$. We experiment with SOOADT for different depths of the tree. We first observe the performance of SOOADT on toy examples for different values of the control parameter α as we illustrate in

the next section. We then validate the performance of SOOADT on real-life data sets. We then show the capability of the SOOADT in performing unsupervised classification for class discovery on real-life data sets.

4.2 Results

4.2.1 Synthetic Data

First we demonstrate the capability of SOOADT on certain sysnthetic datasets. We have designed the synthetic datasets in such a way that the clusters are visually apparent yet sometimes it is difficult to obtain good clustering results using the usual c-means type of algorithms. Our objective here is to show that SOOADT self-organizes itself in the online adaptive mode such that it not only forms a hierarchy but also is able to identify different regions in the datasets.

Example I: In this example, there are eight clusters forming an octagon. Figure 3 illustrates the output of SOOADT with depths of $l = 2,3,4,5$, and 6 respectively. For SOOADT of depth 2, we have only 4 leaf nodes such that SOOADT can identify at most 4 different groups. An SOOADT of depth 3 having eight leaf nodes can identify eight clusters exactly. As we increase the depth of SOOADT, we observe that it extracts only eight groups in the data and does not split a group into more than one cluster. We illustrate the results for an SOOADT with the parameter setting, $m = 5\log(l/\epsilon)$ where l is the depth of the tree. We fix $\alpha = 1.2$ which is slightly greater than unity. Figure 3 illustrates the results obtained for 200 epochs, although we observe that SOOADT of depths 2, 3 and 4 converge in less than 50 epochs.

With this same example, we illustrate the effect of reducing the parameter m (the slope of the transfer function associated with each non-terminal node of the adaptive tree) in Figure 4. We use $m = m_0\log(l/\epsilon)$, and select $m_0 = 1,3,5$, and 10 respectively. The result for $m_0 = 5$ is already provided in Figure 3. For a comparison we include the result for $m_0 = 5$ in Figure 4 also. We illustrate the results for a depth $l = 5$, $\alpha = 1.2$ and 200 epochs. We observe that for relatively lower slope values, SOOADT identifies a larger number of groups in the data. This is due to the fact that for relatively smaller m, the activation is ditributed among the leaf nodes such that no leaf node is fully activated, and at cluster boundaries more than one leaf node interact with each other. This forces SOOADT to allocate relatively smaller regions to each leaf node. The scenario becomes just the opposite as we increase m and we observe that only 6 groups are identified for $m_0 = 10$. This phenomenon is also evident from Equation (24) where the value of α_0 increases with m. Therefore, with a fixed α_0, larger clusters are extracted from the data with a large value of m and vice-versa.

As we stated the effect of α in grouping the data by SOOADT (Section 3.3), we illustrate the same on this example in Figure 5. We illustrate the results for $\alpha = 1,1.2,1.4$, and 1.6 respectively for an SOOADT for depth $l = 5$ with $m_0 = 5$ for 200 epochs. We observe that for an $\alpha = 1.0$, relatively large number of regions are identified by SOOADT. This is due to the fact that initially the regions are created and due to smaller value of α they are not marged with other regions. As

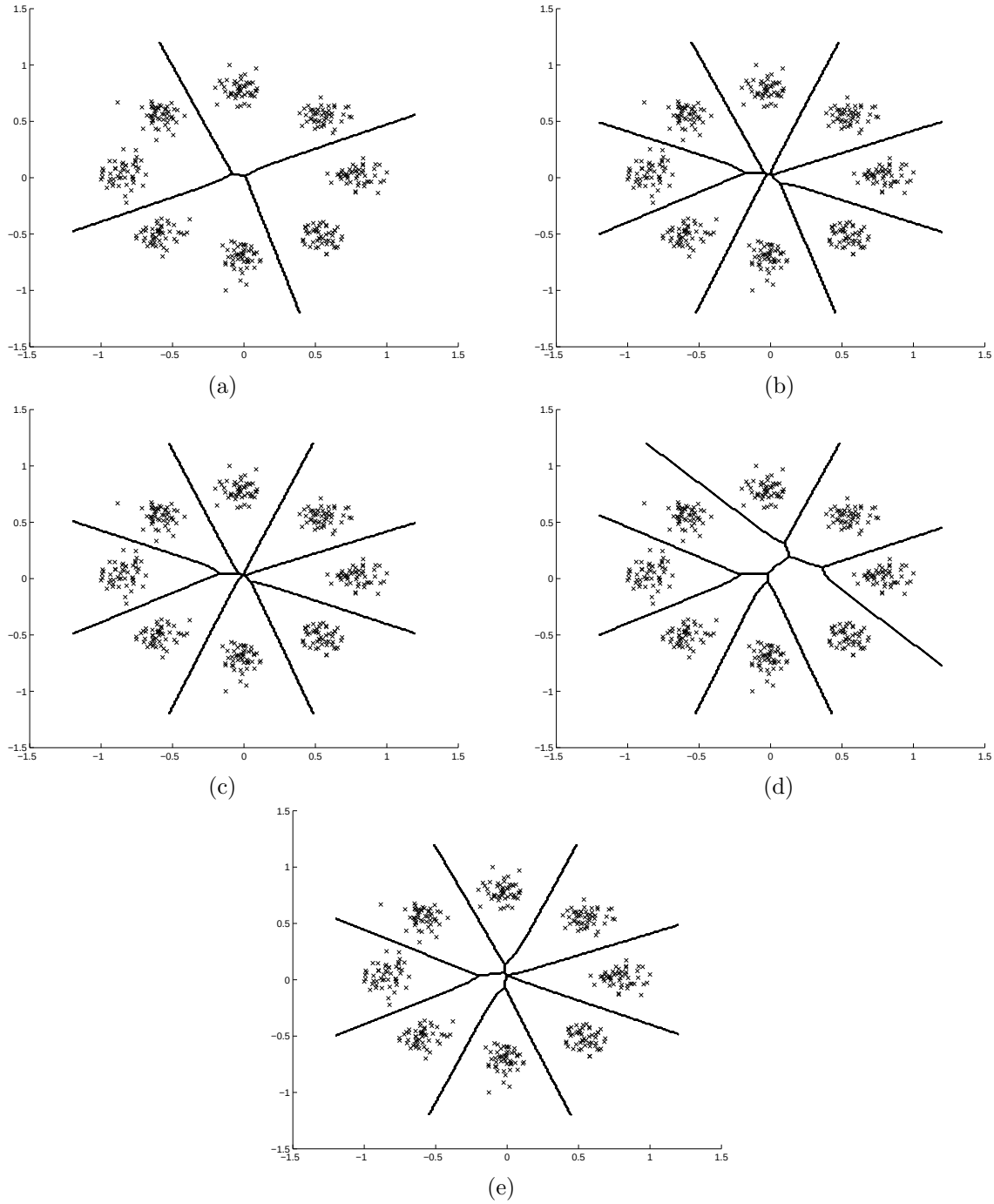


Figure 3: The decision hyperplanes constructed by the SOOADT with a constant $\alpha = 1.2$ and $m_0 = 5$. The performance for depths equal to 2, 3, 4, 5, and 6 are illustrated in (a),(b), (c), (d), (e), and (f) respectively.

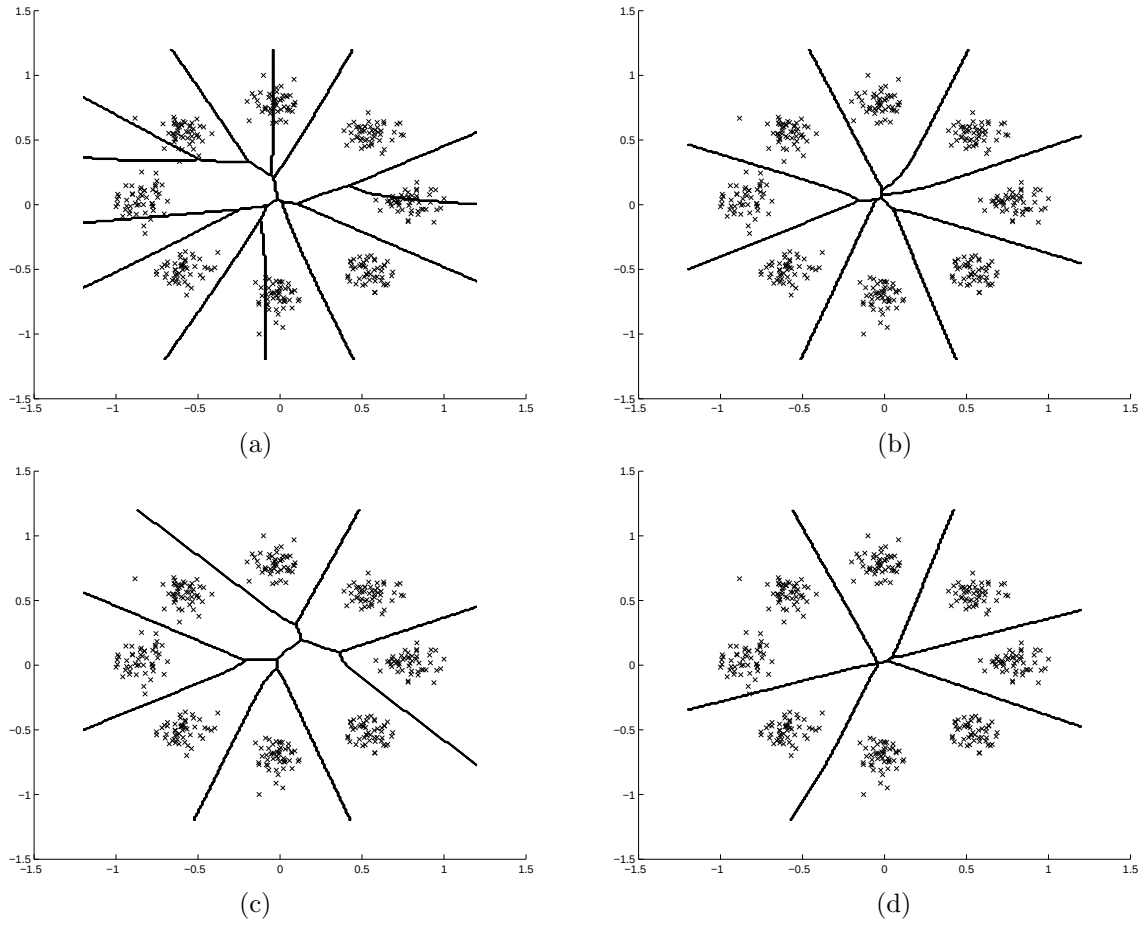


Figure 4: The decision hyperplanes constructed by the SOOADT with a constant $\alpha = 1.2$ and depth equal to 5. The parameter m_0 is varied. (a), (b), (c), and (d) illutrate the behavior of the SOOADT for $m_0 = 1$, $m_0 = 3$, $m_0 = 5$, and $m_0 = 10$ respectively.

α is increased to 1.6, we observe that different groups of data are merged into one segment by SOOADT, and seven different groups are identified.

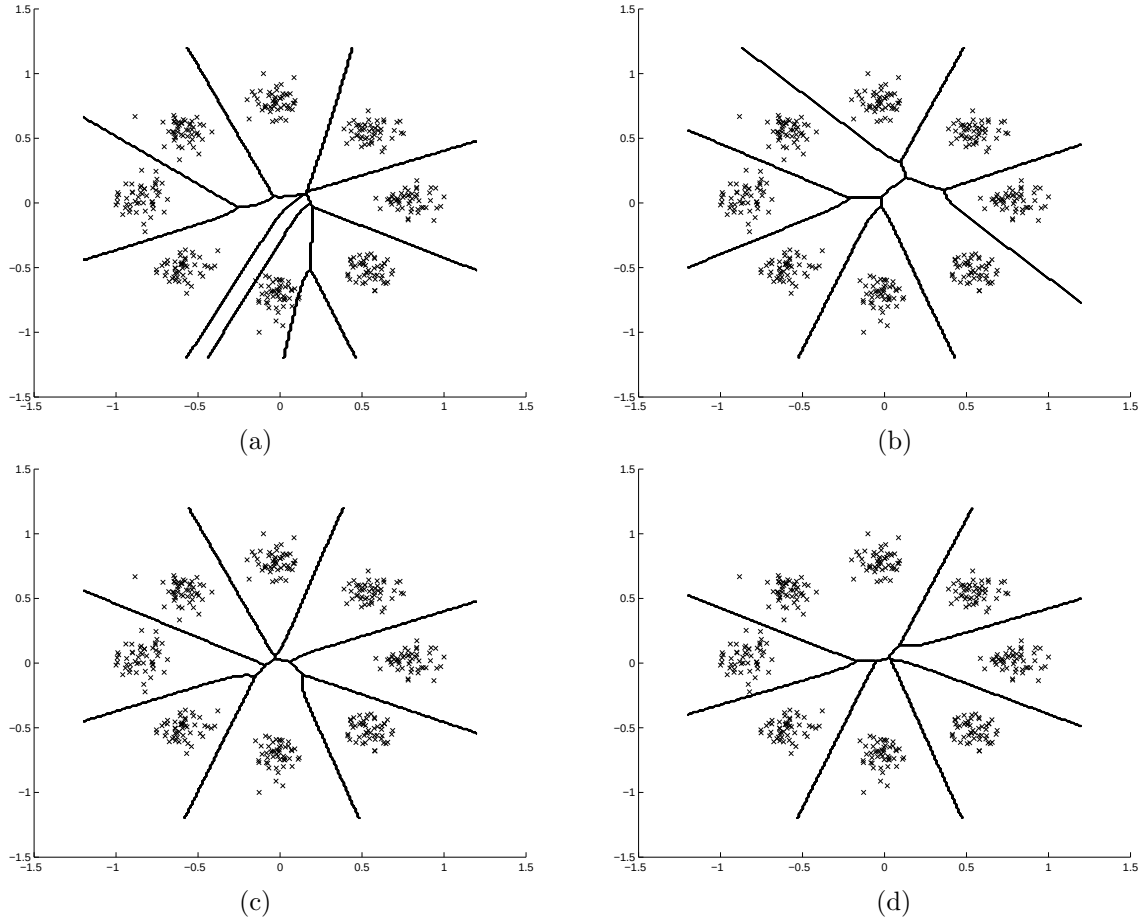


Figure 5: The decision hyperplanes constructed by the SOOADT with a constant $m_0 = 5$ and depth equal to 5. The control parameter α is varied. (a), (b), (c), and (d) illustrate the behavior of the SOOADT for $\alpha = 1$, $\alpha = 1.2$, $\alpha = 1.4$, and $\alpha = 1.6$ respectively.

Exmaple II: In the previous example, we illustrated that SOOADT is capable of identifying the groups for equally spaced data clusters. In this example, we show slightly more complex example. Note that it is not necessary that SOOADT allocates only a single leaf node to one cluster or group of data, rather SOOADT self-organizes itself in the online adaptive mode such that one compact group of data is allocated to a set of leaf nodes. The objective here is that data points from different clusters should be allocated to different leaf nodes. Similar to the previous example, we observe that with the increase in α, SOOADT gradually allocates larger chunks of data to the leaf nodes. We illustrate the behavior of SOOADT in Figure 6. We observe that in Figure 6 (c),(f),(h), the three clusters are perfectly separated with only three leaf nodes. In other sub-figures, we observe that each group of samples is identified by a set of leaf nodes together.

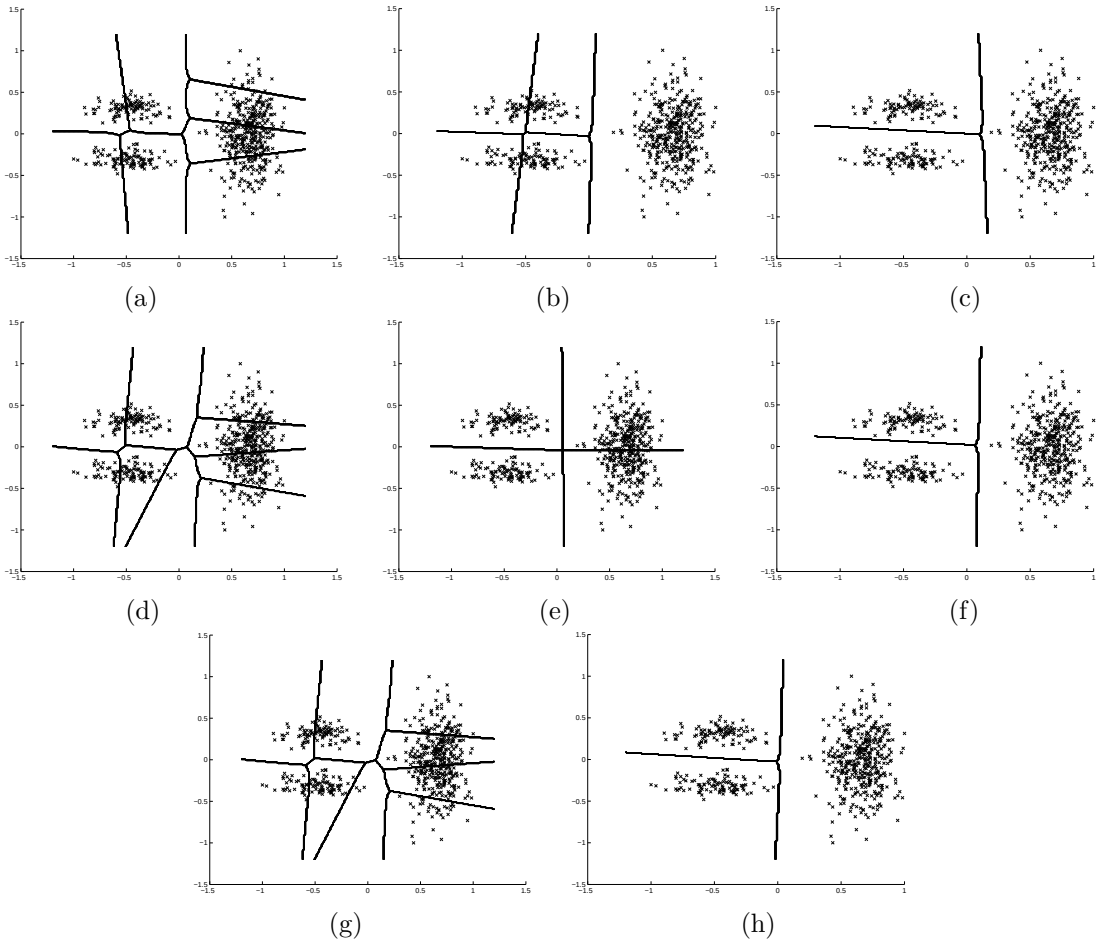


Figure 6: The decision hyperplanes constructed by the SOOADT for different depths with different control parameters with a constant $m_0 = 5$. The constructed hyperplanes are shown for (a)depth equal to 3 and $\alpha = 1.2$, (b) depth equal to 3 and $\alpha = 1.5$, (c) depth equal to 3 and $\alpha = 3.0$, (d) depth equal to 4 and $\alpha = 1.2$, (e) depth equal to 4 and $\alpha = 3.0$, (f) depth equal to 4 and $\alpha = 4.0$, (g) depth equal to 5 and $\alpha = 1.2$, (h) depth equal to 5 and $\alpha = 4.0$

Example III: In this example, we chose data disctribution similar to that in the first example except that we include a large cluster of data in the central region. Interestingly we observe that SOOADT is able to create region for the central cluster with more than one leaf node. We illustrate the behavior of SOOADT in Figure 7. Here we observe that the central cluster (which is bigger in size) is identified by a group of leaf nodes.

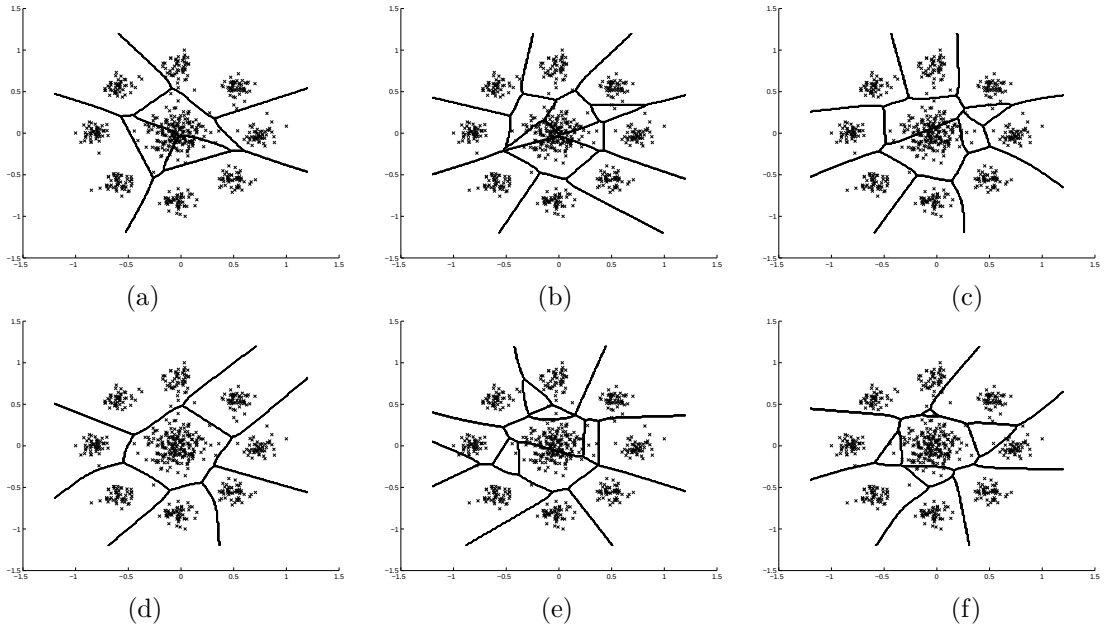
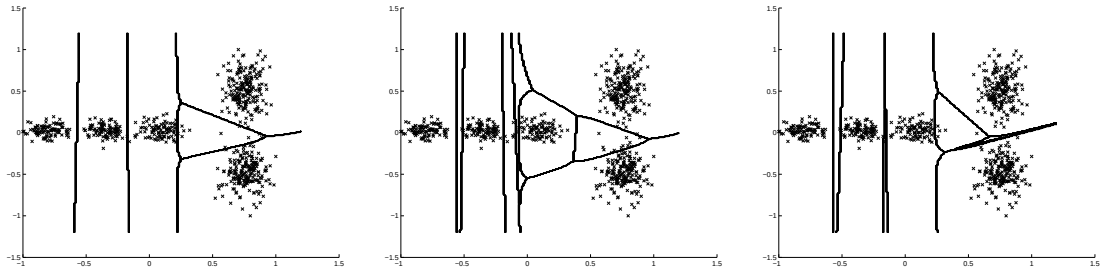


Figure 7: The decision hyperplanes constructed by the SOOADT for different depths with different control parameters with a constant $m_0 = 5$. The constructed hyperplanes are shown for (a) depth equal to 3 and $\alpha = 1.2$, (b) depth equal to 4 and $\alpha = 1.2$, (c) depth equal to 4 and $\alpha = 1.5$, (d) depth equal to 4 and $\alpha = 2.0$, (e) depth equal to 5 and $\alpha = 2.0$, and (f) depth equal to 5 and $\alpha = 3.0$.

Example IV: Here we illustrate (Figure 8) the behavior of SOOADT for three different depths with a constant $\alpha = 3.0$. The overall behavior for other values of α is similar as in other examples. In this example, we observe that as we increase the depth of the tree, the clusters are identified, however, some extra fragmented regions are also formed. This is due to the fact that initially these regions are allocated to certain leaf nodes, and with the present setting of α, the SOOADT is not able to merge them for higher depths. If we increase α then these regions are merged, however, more than one cluster is also merged into one region (one leaf node).

4.2.2 Real-life Data

Here we illustrate the behavior of SOOADT for two real-life datasets namely, Ripley's "crabs" dataset (Ripley, 1996) and Fisher's "iris" dataset (Fisher, 1936).



(a) (b) (c)

Figure 8: The decision hyperplanes constructed by the SOOADT with a constant $m_0 = 5$ and $\alpha = 3.0$ for different depths (a) equal to 3, (b) equal to 4, and (c) equal to 5.

Crabs Data: In crabs dataset, there are total 200 samples, each sample having five attributes. The crabs are from two different species, each species has male and female categories. Thus there are in total four different classes and the data set contains 50 samples from each class. The samples from different classes get highly overlapped when the data is projected onto two-simensional space spanned by the first and second principal components. Interestingly, the four classes are nicely observable when the data is displayed in the space spanned by the second and third principal components. We therefore obtain the second and third principal components and project the dataset onto these two principal components to obtain the two dimensional projected data. We then normalize such that each projected sample is bounded in [-1,1]. Figure 9(a)-(d) illustrates the behavior of SOOADT for depths equal to two to five with a constant $\alpha = 1.2$. Figure 9(e)-(f) illustrates the behavior for depths equal to four and five with $\alpha = 2.0$. In Figure 9(a), we observe that the SOOADT is not able to separate the data points from different classes completely with only four regions. However, if we consider only species information, then the two different species are well sepaparated by the SOOADT with a depth of 2. In Figure 9(c) only, we observe that the SOOADT created a region where samples from two different classes (same species with different sex) are mixed up. In all other situations, the SOOADT is able to separate the four different classes by allocating more than one leaf node to each class.

Iris Data: We next illustrate the behavior of SOOADT for 'iris' dataset (the original dataset was reported in (Fisher, 1936), and we obtain the data from (Blake & Merz, 1998)). In 'iris' data, there are three different types of iris flowers, each category having four different features namely sepal length, sepal width, petal length, and petal width. There are 50 samples from each class resulting in a total of 150 samples in the four dimensional space. The three different classes can be nicely identified when the data set is displayed in a two-dimensional space spanned by two derived features namely sepal area (sepal length $*$ sepal width) and petal area (petal length $*$ petal width). These two dimensions together have high discriminating capability. We derive these two features for each sample and then normalize each sample in [-1,+1]. Figure 10 illustrates the behavior of SOOADT for this dataset with these two normalized derived features for depths equal to two to four with a constant $\alpha = 1.2$. We observe that the SOOADT with depth equal to 2, is able to separate one class completely. However, it creates one region (corresponding to one leaf node) where samples from two different classes are mixed up. The purity of the regions improves a lot when we move to an SOOADT with a depth equal to 3, and we observe that an

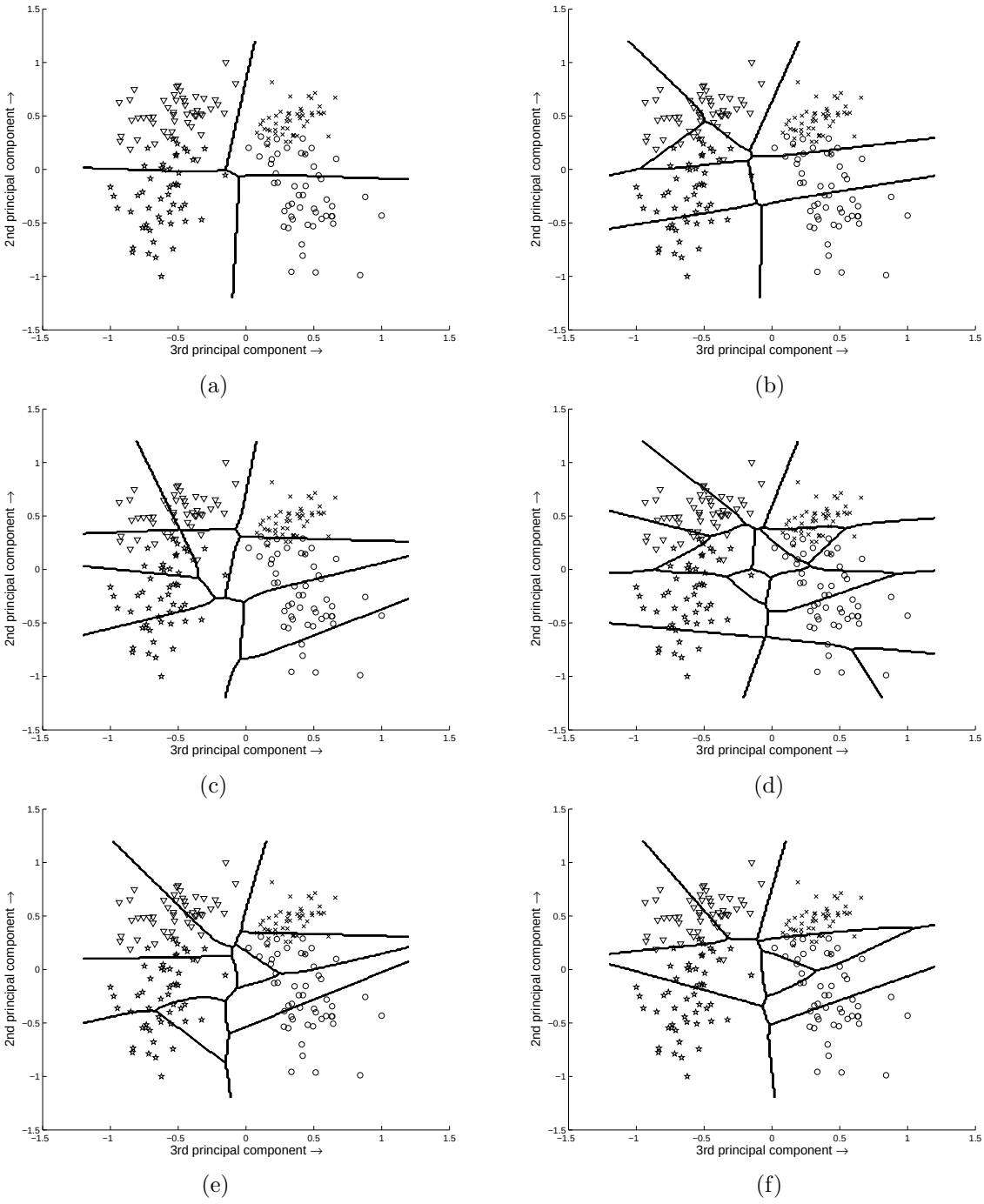


Figure 9: The decision hyperplanes constructed by the SOOADT on the "crabs" dataset projected onto a two-dimensional plane spanned by the second and third principal components. The four different classes in the dataset are marked with different notations. Figure illustrates the hyperplanes with a constant $m_0 = 5$ for (a) depth equal to 2 and $\alpha = 1.2$, (b) depth equal to 3 and $\alpha = 1.2$, (c) depth equal to 4 and $\alpha = 1.2$, (d) depth equal to 5 and $\alpha = 1.2$, (e) depth equal to 4 and $\alpha = 2.0$, (f) depth equal to 5 and $\alpha = 2.0$.

SOOADT with a depth equal to 4 is able to perfectly separate the three different classes from the data.

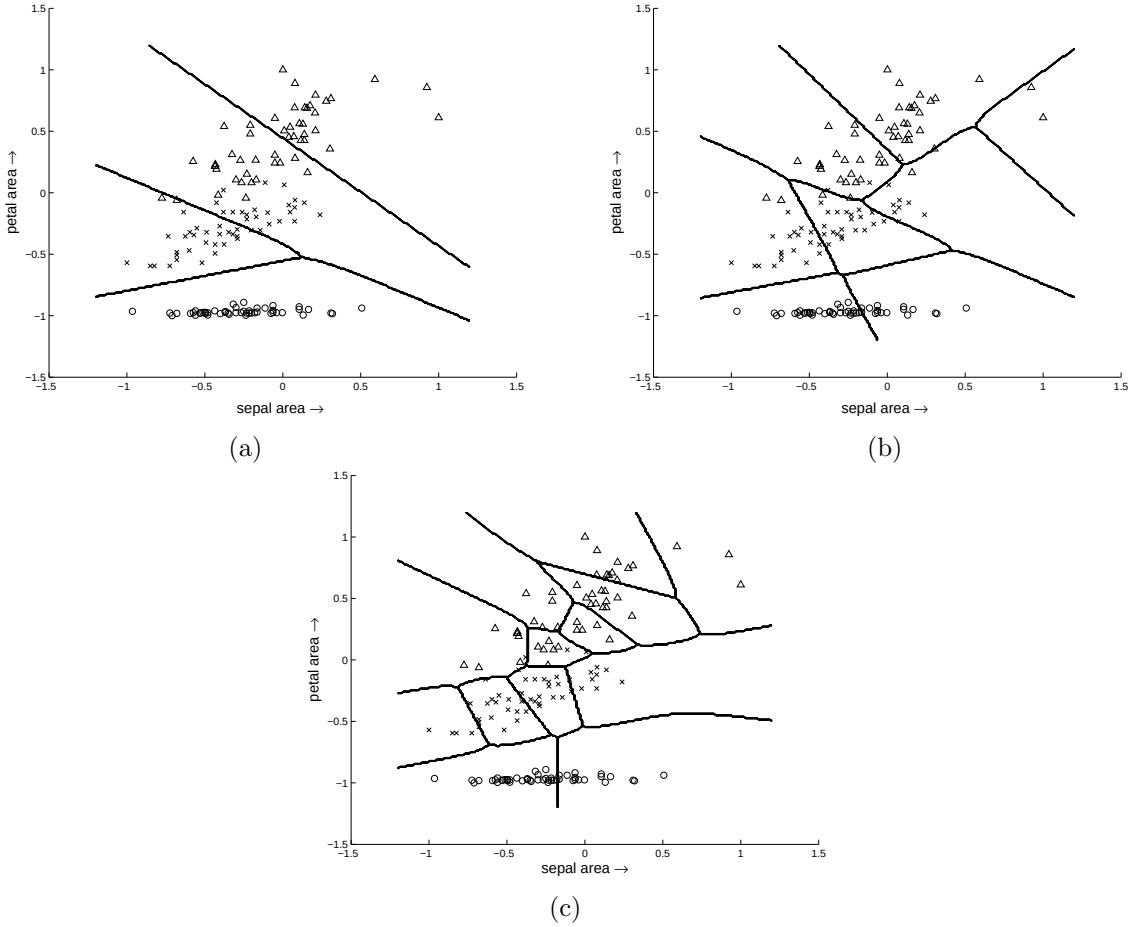


Figure 10: The decision hyperplanes constructed by the SOOADT on the "iris" dataset projected onto a two-dimensional plane spanned by the sepal area (product of first and second attribute values) and the petal area (product of third and fourth attribute values). Figure illustrates the hyperplanes with a constant $m_0 = 5$ and $\alpha = 1.2$ for (a) depth equal to 2, (b) depth equal to 3, and (c) depth equal to 4.

4.2.3 Unsupervised Classification on Real-life Data

Here we demonstrate the effectiveness of SOOADT in performing the class discovery on certain real-life data sets by means of unsupervised classification. In order to obtain the classification performance, we consider the samples allocated to each leaf node. For each leaf node, we obtain the classlabel of the majority of the samples allocated to that leaf node, and assign the corresponding class label to that leaf node. Thus we assign a specific class label to each leaf node of SOOADT depending on the groups of samples allocated to that leaf node. In crabs dataset,

we use two class labels (species only) instead of the four labels. Apart from the iris and crabs dataset, we also use three other data sets namely pima-indians-diabetes (originally reported in (Smith, Everhart, Dickson, Knowler, & Johannes, 1988), we call it 'pima'), Wisconsin Diagnostic Breast Cancer (originally reported in (Street, Wolberg, & Mangasarian, 1993), in short called 'WDBC'), and Wisconsin Prognostic Breast Cancer (originally reported in (Street, Mangasarian, & Wolberg, 1995), in short called 'WPBC'). We obtained these three data sets from (Blake & Merz, 1998). All three data sets namely Pima, WDBC, and WPBC have two different class labels. Pima data set contains eight attributes and 768 samples, WDBC has 569 samples each with 30 attributes, and WPBC consists of 198 samples with 32 attributes. We report the classification accuracies obtained by SOOADT in Table 1. As a comparison, we provide the results obtained by a supervised decision tree, C4.5, (Quinlan, 1993, 1996; Witten & Frank, 2000) on these data sets. We report the 10-fold cross-validation (CV) score for C4.5. We compare the cross-validation score since SOOADT operates completely in the unsupervised mode and the purpose is to reveal its capability in class discovery. We observe that SOOADT is able to identify groups such that the class information can be extracted from its leaf nodes.

Dataset	depth=3	depth=4	depth=5	depth=6	C4.5 (depth)
Pima	68.62	68.62	69.40	70.31	74.09 (9)
WPBC	76.77	76.77	76.77	76.77	76.77 (6)
Iris	92.00	96.00	96.67	97.33	95.33 (4)
Crabs	65.00	66.50	71.50	74.00	90.5 (7)
WDBC	88.05	89.28	90.33	90.16	92.44 (6)

Table 1: Accuracies on real-life datasets for unsupervised classification by the SOOADT with a constant $m_0 = 5$, $\alpha = 1.2$ for different depths from 3 to 6. A cross-validation performance of the supervised C4.5 decision tree is shown with the corresponding depths of the tree.

4.2.4 Unsupervised Classification of Leukemia Samples in Gene Expression Data

We used SOOADT in performing unsupervised classification of acute leukemia samples from the gene expression using DNA microarray as used in Golub et al (Golub et al., 1999). In the data set, there are 72 samples consisting of two different types of leukemia namely, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). For each sample, gene expression was monitored using Affymetrix expression arrays. Golub et al. used self-organizing map to study the gene expression data and performed the unsupervised classification of the cancerous genes. The same data has been used in (Lange, Roth, Braun, & Buhmann, 2004) where appropriate model order has been obtained based on stability-based validation of the clustering solutions. The original data consists of 72 different samples (47 ALL and 25 AML samples), each expressed by 6817 genes. We preprocess the data using the same technique as reported in (Lange et al., 2004). First we restrict all expression levels in $[100, 16000]$ by assigning boundary values to the gene expressions outside this range. We then retain only those genes having quotient of maximum and minimum expression levels across the samples greater than 5, and the difference between maximum and minimum expression levels across the samples greater than 500. The

resulting data set has 72 samples each with 3571 genes. We then $\log_{10}$ transform the data and normalize each sample to have zero mean and unit variance across the genes. We then use top 100 genes which have highest variance across the samples. Thus the final data set consists of 72 samples each being expressed by 100 genes. We then further normalize each sample such that each variable lies in the range $[-1, +1]$. With this 100 dimensional data set, we use SOOADT to perform unsupervised classification as explained in the previous section. For an SOOADT of depth 3, we achieve 80.56% accuracy, whereas for depths of 4 and 5 we consistently obtain an accuracy of 86.11% (62 samples were correctly classified). This is exactly the same accuracy that has been achieved in (Lange et al., 2004) using the ALL-AML labeling. This shows that in this specific data set, if the true labeling were unknown then SOOADT is able to group the data properly so that different cancer classes could be discovered. With an SOOADT of depth 6 we obtain 84.7% accuracy (61 samples are labeled correctly). The reduction in the accuracy happens due to the fact that SOOADT of a larger depth creates too many fragmented regions and therefore in some regions very few samples are contained. This reduces the classification accuracy since we decide the class labels based on the majority of labels in each leaf. Overall we observe that SOOADT is able to identify the groups properly so that the information about difference cancerous classes can be extracted from the leaf nodes.

5 Discussion and Conclusions

We proposed a model namely SOOADT which is essentially an extention of the online adaptive decision trees (Basak, 2004, 2006) for performing unsupervised grouping on the input data. The SOOADT is able to self-organize itself within a tree structure towards extracting the different groups from the data. We have shown the effectiveness of the SOOADT over synthetic and real-life data sets. We observed that in real-life data sets, the SOOADT is able to identify the different classes in an unsupervised mode either with a single leaf node or with more than one leaf node collectively. In syntheric data sets also we observed the capability of the SOOADT in identifying different groups of data either with a single leaf or more than one leaf collectively. We have also reported the unsupervised classification scores on certain real-life data sets and compared them with the cross-validation score of a supervised decision tree namely, C4.5. We also have demonstrated the capability of the SOOADT in class discovery for acute leukemia samples using gene expression data.

The SOOADT is constructed by augmenting a code formation layer with the basic model of OADT. Interestingly, the code formation layer forms the codes although the activation of leaf nodes do not depend on the formed codes. This is a basic difference with most of the existing self-organizing networks in the literature. In the existing self-organizing networks, usually the firing activity of the neurons depend on the stored codes. On the other hand, in SOOADT the leaf activation depends on the stored hyperplanes in the tree and these hyperplanes are guided by the stored codes during the adaptation process. We do not incorporate any explicit competitive meachnism between the activated leaf nodes. One of the leaf nodes automatically gets maximally activated since the sum of leaf node activations is always constant. We derive the entire process

of self-organization by minimizing an objective functional. The parameter α in the objective functional controls the entire process of learning. In the limits when $\alpha \to \infty$ then all input data are considered to be in one cluster and the parameters in the adaptive tree are adapted accordingly. On the other hand for $\alpha = 0$, there is no adaptive property in the SOOADT and the hyperplanes as well as the codes get fixed. In our experimental setting, the SOOADT exhibits balanced adaptive property with a choice of α slightly greater than unity. For a choice of $\alpha \leq 1$, many fragmented groups can form. We have also shown the behavior of the SOOADT for various different choices of α. The objective functional in the SOOADT looks similar in form to that in the soft c-means (Bezdek & Castelaz, 1977; Bezdek, 1981). However, in soft c-means, the number of groups, c, is fixed and the performance of the algorithm depends on the control parameter α. As the parameter $\alpha \leq 1$ approaches unity, soft c-means behaves like the hard c-means algorithm. The parameter α in soft c-means controls the radius of the hypersphere for computation of the cluster centers. On the other hand in SOOADT, the parameter α controls the behavior of the adaptive tree; with a very small α, the SOOADT loses plasticity and the codes are not adapted whereas with a large α many codes are merged. Unlike soft c-means, in SOOADT there is no concept of choosing number of groups in the data beforehand, rather it depends on the control parameter α and the depth of the adaptive tree. We empirically have shown that the control parameter α in the SOOADT can be chosen slightly greater than unity. The optimal value of the parameter depends on the data distribution and the depth of the tree. A more rigorous analysis is required towards formally establishing the behavior of the SOOADT to obtain the best convergence depending on the control parameter, depth of the tree and the data distribution. This constitutes a scope of future study in the proposed SOOADT model.

We used steepest gradient descent using line search meachnism (Friedman, 2001) to adapt the tree. Since the learning rate for every instance of every data sample is derived based on the input and the network parameters, effectively the learning rate is never frozen and the SOOADT can adapt to any data distribution. In the original self-organizing map (Kohonen, 1988), on the other hand, the learning rate is decreased subject to $\sum_t \eta_t \to \infty$ and $\sum_t \eta_t^2 < \infty$. The performance and the adaptation rate of the SOOADT can possibly be further enhanced by using smarter learning algorithms. In this context, we mention that we also implemented SOOADT using natural gradient descent algorithm (Amari, 1998). In the natural gradient descent algorithm, the inverse Fisher information matrix is estimated and the learning gradient is multiplied by the inverse Fisher information matrix. In (Amari, Park, & Fukumizu, 2000), the inverse Fisher information matrix has been estimated with an approximation in the online mode (usually the exact computation of inverse Fisher information matrix can be performed only with the entire batch of samples). In this online approximation of the inverse Fisher matrix computation, one more parameter similar to that of the learning rate, has been used and is decreased with the number of iterations. With such kind of online approximation of the inverse Fisher matrix computation, we lose the adaptivity in the sense that the learning rate parameter again gets frozen after observing a certain number of sample instances. In order to make it adaptive, we used a fixed parameter value (0.1) to estimate the approximation of the inverse Fisher information matrix, and at every sample iteration we normalize the inverse matrix such that its determinant is unity. We choose the learning rate (a scalar quantity) using the same line search and modify the learning gradient with the approximated inverse Fisher information matrix. Using this modified

form of natural gradient descent implementation, we observe that the the SOOADT indeed takes less number of iterations to produce similar results, however since the computation of normalized inverse Fisher information matrix at each iteration becomes costlier, the effective speed-up in adaptation becomes marginal and in some cases it is rather worse. It is not necessarily any drawback of the natural gradient descent algorithm, rather our implementaton of the modified natural gradient descent did not show significant improvement. However, we did the modification in order to preserve the adaptivity property of the SOOADT. Nonetheless, the behavior of the SOOADT can possibly be further improved with better learning algorithms as a scope for further study.

In short, we have shown that OADT can be extended to exhibit self-organizing property in the online adaptive mode towards obtaining different coherent groups from the input data distribution. This property can be suitably exploited for effective class discovery from the real-life data. The proposed model, namely SOOADT, does not incorporate any competitive learning meachanism unlike the existing self-organizing mechanisms in the neural network domain. Rather the SOOADT with a given depth adapts the stored decision hyperplanes and the codes together towards extracting different groups from the input distribution and the process is guided by a certain control parameter α of the network.

References

Amari, S. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Trans. Computer*, *C-21*, 1197-1206.

Amari, S. (1977). Neural theory of association and concept formation. *Biological Cybernetics*, *26*, 175-185.

Amari, S. I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251-276.

Amari, S. I., Park, H., & Fukumizu, K. (2000). Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, *12*(6), 1399-1409.

Basak, J. (2004). Online adaptive decision trees. *Neural Computation*, *16*, 1959-1981.

Basak, J. (2006). Online adaptive decision trees: Pattern classification and function approximation. *Neural Computation*, *18*, 2062-2101.

Basak, J., & Krishnapuram, R. (2005). Interpretable hierarchical clustering by constructing an unsupervised decision tree. *IEEE Transactions Knowledge and Data Engineering*, *17*, 121-132.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms.* New York: Plenum Press.

Bezdek, J. C., & Castelaz, P. (1977). Prototype classification and feature selection with fuzzy sets. *IEEE Trans. on Systems, Man and Cybernetics*, *7*, 87-92.

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases.* University of California, Irvine, Dept of Information and Computer Sciences. Also available at:*http://www.ics.uci.edu/simmlearn/MLRepository.html*.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1983). *Classification and regression trees.* New York: Chapman & Hall.

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning*, *19*, 45-77.

Carpenter, G., & Grossberg, S. (1987). Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, *26*, 4919-4930.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, *37*, 34-115.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification (2nd edition).* New York: Wiley.

Durkin, J. (1992). Induction via ID3. *AI Expert*, *7*, 48-53.

Fayyad, U. M., & Irani, K. B. (1992). On the handling of continuous-values attributes in decision tree generation. *Machine Learning*, *8*, 87-102.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*, 179-188.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*, 1-141.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals Statistics*, *29*, 1189-1232.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caliguri, M. A., Bloomfield, C. D., & Lander, E. S. (1999).

Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531-537.

Grossberg, S. (1987). Competitive learning : from interative activation to adaptive resonance. *Cognitive Science*, *11*, 23-63.

Haykin, S. (1999). *Neural networks: A comprehensive foundation.* Upper Saddle River, NJ: Prentice Hall.

Held, M., & Buhmann, J. M. (1998). Unsupervised on-line learning of decision trees for hierarchical data analysis. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Neural information processing systems nips' 1997* (Vol. 10). USA: MIT Press.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data.* Upper Saddle River, NJ: Prentice-Hall Inc.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*, 264-323.

Kohonen, T. (1988). *Self-organization and associative memory.* Berlin: Springer-Verlag.

Lange, T., Roth, V., Braun, M. L., & Buhmann, J. M. (2004). Stability-based validation of clustering solutions. *Neural Computation*, *16*, 1299-1323.

Linsker, R. (1986). From basic network principles to neural architecture : Emergence of spatial opponent cells. *Proceedings National Academy of Sciences, USA*, *83*, 7508-7512.

Liu, B., Xia, Y., & Yu, P. (2000). Clustering through decision tree construction. In *Proceedings of the ninth international conference on information and knowledge management* (p. 20-29). McLean, VA, USA: ACM.

Malsburg, C. Von-der. (1973). Self-organization of orientation selective cells in striate cortex. *Kybernetics*, *14*, 85-100.

Mao, J., & Jain, A. K. (1996). A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions Neural Networks*, *7*, 16-29.

Quinlan, J. R. (1993). *Programs for machine learning.* San Fransisco, CA, USA: Morgan Kaufmann.

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence*, *4*, 77-90.

Ripley, B. D. (1996). *Pattern recognition and neural networks.* Cambridge, UK: Cambridge University Press.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988). Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the twelfth annual symposium on computer application in medical care* (p. 261-265). Washington D.C., NY, USA: IEEE Computer Society Press.

Street, W. N., Mangasarian, O. L., & Wolberg, W. H. (1995). An inductive learning approach to prognostic prediction. In A. Prieditis & S. Russell (Eds.), *Proceedings of the twelfth international conference on machine learning* (p. 522-530). San Francisco: Morgan Kaufmann.

Street, W. N., Wolberg, W. H., & Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. In *Proceddings is&t/spie international symposium on electronic imaging: Science and technology* (Vol. 1905, p. 861-870). San Jose, CA.

Witten, I. H., & Frank, E. (2000). *Data mining : Practical machine learning tools and techniques with Java implementations (chapter 8).* USA: Morgan-Kauffmann.