Appl Intell (2017) 47:96–113
DOI 10.1007/s10489-016-0886-8

Tree-based localized fuzzy twin support vector clustering with square loss function

Reshma Rastogi[1] · **Pooja Saigal**[1]

Published online: 13 February 2017

Abstract Twin support vector machine (TWSVM) is an efficient supervised learning algorithm, proposed for the classification problems. Motivated by its success, we propose Tree-based localized fuzzy twin support vector clustering (Tree-TWSVC). Tree-TWSVC is a novel clustering algorithm that builds the cluster model as a binary tree, where each node comprises of proposed TWSVM-based classifier, termed as localized fuzzy TWSVM (LF-TWSVM). The proposed clustering algorithm Tree-TWSVC has efficient learning time, achieved due to the tree structure and the formulation that leads to solving a series of systems of linear equations. Tree-TWSVC delivers good clustering accuracy because of the square loss function and it uses nearest neighbour graph based initialization method. The proposed algorithm restricts the cluster hyperplane from extending indefinitely by using cluster prototype, which further improves its accuracy. It can efficiently handle large datasets and outperforms other TWSVM-based clustering methods. In this work, we propose two implementations of Tree-TWSVC: Binary Tree-TWSVC and One-against-all Tree-TWSVC. To prove the efficacy of the proposed method, experiments are performed on a number of benchmark UCI datasets. We have also given the application of Tree-TWSVC as an image segmentation tool.

✉ Reshma Rastogi
reshma.khemchandani@sau.ac.in

Pooja Saigal
pooja.saigal@students.sau.ac.in

[1] Department of Computer Science, South Asian University, New Delhi, India

Keywords Localized fuzzy twin support vector machine · Unsupervised learning · Fuzzy tree-based clustering · Square loss function · Image segmentation through clustering

1 Introduction

Machine learning approaches can be broadly classified as supervised and unsupervised learning, based on the availability of data labels or output. Supervised learning uses labelled training samples, while unsupervised learning is 'learning without label information'. Maximum margin classifiers like support vector machines (SVM) [1] have been successfully used for supervised classification. An improvement over SVM was proposed by Jayadeva et al. termed as TWSVM [2, 3]. TWSVM is a supervised learning method that classifies data by generating two nonparallel hyperplanes which are proximal to their respective classes and at least unit distance away from the patterns of other class. TWSVM solves a pair of quadratic programming problems (QPPs) and is based on empirical risk minimization principle; it is proved to be almost four times faster than SVM [2]. Many variations of TWSVM have been proposed, which include Least Squares TWSVM (LS-TWSVM) [4], fuzzy version of LS-TWSVM [5] and robust energy-based LS-TWSVM [6]. LS-TWSVM solves a system of linear equations and hence, it is faster than TWSVM. Shao et al. proposed least square twin parametric-margin SVM [7]. Recently, Khemchandani et al. proposed multi-category Laplacian TWSVM [8].

In many practical problems, the cost of obtaining the labels is very high as compared to the cost of generating the data, for example speech recognition, web page

classification, video surveillance, etc. In this scenario, unsupervised learning approaches can be exploited, but they are more challenging as the data labels are not available. Clustering is an unsupervised learning task, which aims at partitioning data into a number of clusters [9–11]. Patterns that belong to the same cluster should have an affinity with each other and must be distinct from the patterns in other clusters. Clustering has its application in various domains of data analysis which include medical science, finance, pattern recognition and image analysis [12–14].

Clustering algorithms aim at partitioning the data into a number of clusters [9] based on similar features by using algorithms like K-means clustering [10] and hierarchical clustering [11]. The K-means algorithm [10] is a centroid-based clustering method that represents each cluster by its mean vector. Hierarchical clustering [11] is based on distance connectivity. Clusters can be identified by using statistical distributions, such as multivariate normal distributions used in expectation-maximization algorithm [15]. Al-Harbi et al. proposed adaptive K-means clustering [16]. Plane-based clustering methods have been proposed such as K-plane clustering (KPC) [17] and local k-proximal plane clustering (LkPPC) [18] by Bradley et al. and Yang et al. respectively. For a K-cluster problem, theses algorithms identify K clusters by generating K planes.

Following the success of margin-based classifiers in supervised learning, researchers have been trying to extend them to unsupervised learning. Recently, Xu et al. [19] proposed maximum margin clustering (MMC) which performs clustering in SVM framework and finds a maximum margin separating hyperplane between clusters. Valizadegan et al. proposed a variation of MMC termed as generalized MMC [20]. Large margin supervised learning methods are usually formulated as QPPs, which suggests that they could be computationally more expensive. Since the labels are missing in unsupervised problems, the resulting optimization problem would be hard and non-convex that considers all the combinations over possible discrete class labels. MMC based methods [20] resort to relaxing the non-convex clustering problem as a semidefinite program (SDP) [21]. MMC can not be used for very large datasets because SDP is computationally expensive [22]. Zhang et al. [23] proposed a feasible variation for MMC and implemented MMC as an iterative support vector machine (iterSVM).

Recently, Wang et al. proposed TWSVM for clustering (TWSVC) [24] that uses information from both within the cluster and between clusters. It requires that the plane of one cluster should be a unit distance away from patterns of other clusters on both the sides. TWSVC is an iterative clustering algorithm that solves K QPPs for a K-cluster problem using one-against-all approach (OAA) [25]. Due to the nature of loss function used by TWSVC (as discussed in Appendix A), the labels flip rarely in successive iterations i.e. change their value from 0 to 1 or vice-versa. Hence, the accuracy of TWSVC is limited by the performance of initialization algorithm and the labels seldom change in the iterations. Recently, Khemchandani et al. [26] proposed fuzzy least squares TWSVC (F-LS-TWSVC) that uses fuzzy membership to create clusters, which are further obtained by solving systems of linear equations only.

In this paper, we propose Tree-TWSVC which is motivated by MMC [19] and TWSVC [24]. Tree-TWSVC can overcome the limitations of TWSVC and has the following characteristics:

- Unlike MMC that solves expensive SDP problems, the proposed clustering algorithm Tree-TWSVC formulates convex optimization problems which are solved as a system of linear equations. Also, MMC identifies only two clusters whereas Tree-TWSVC identifies multiple clusters by building a tree with LF-TWSVM classifiers at each level.
- Tree-TWSVC recursively divides the data to form the tree structure and iteratively generates the hyperplanes for the partitioned data, until the convergence criterion is met. Due to its tree structure, Tree-TWSVC is much faster than the classical approaches like OAA (used in TWSVC or F-LS-TWSVC), for handling multi-cluster data. It can handle very large-sized datasets with comparable or better clustering results than other TWSVM-based clustering methods.
- At each node of the cluster tree, LF-TWSVM creates two clusters so that the data points of one cluster are proximal to their cluster plane and prototype, and the data points of another cluster should be a unit distance away from this cluster plane. The cluster prototype prevents the plane from extending indefinitely and keeps the plane aligned *locally* to its cluster.
- Tree-TWSVC avoids the approximation through Taylor's series expansion, as required by TWSVC and F-LS-TWSVC due to constraints with mod (or absolute) function and hence Tree-TWSVC gives more accurate results.
- Tree-TWSVC determines a fuzzy membership matrix for data samples which associates a membership value of each data sample to all the given clusters. The initial fuzzy membership matrix is obtained using localized fuzzy nearest neighbour graph (LF-NNG).
- We have used square loss function which is symmetric and allows the output to change (i.e. flip from +1 to -1 or vice-versa) if required, in successive iterations. By using the square loss function, the optimization problem is solved as a system of linear equations; whereas TWSVC solves QPPs to generate the hyperplanes.

The paper is organized as follows: Section 2 gives a brief introduction of MMC, TWSVC and F-LS-TWSVC;

and introduces the notations and symbols used in the paper. Section 3 presents the proposed classifier LF-TWSVM and the clustering algorithm Tree-TWSVC, which is followed by experimental results in Section 4. The paper is concluded in Section 5.

2 Background

Supervised learning algorithms have been used to solve clustering problems such as MMC [19], TWSVC [24] and F-LS-TWSVC [26]. These clustering approaches are briefly explained in the following section.

2.1 Maximum margin clustering (MMC)

Motivated by the success of maximum margin methods in supervised learning, Xu et al. proposed maximum margin clustering (MMC) that aims at extending maximum margin methods to unsupervised learning [19]. Since its optimization problem is non-convex, MMC relaxes the optimization problem as semidefinite programs (SDP).

For the training set $(x_i)_{i=1}^m$, where x_i is the input in n-dimension space and $y = \{y_1, ...y_m\}$ are unknown cluster labels. The primal problem for MMC is given as

$$\begin{aligned} \underset{y}{Min}\ \underset{w,b,\xi}{Min} \quad & \|w\|_2^2 + 2C\xi^T e \\ \text{subject to} \quad & y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad y_i \in \{+1, -1\}, \quad i = 1, ..., m \\ & -l \leq e^T y \leq l, \end{aligned} \tag{1}$$

where ϕ is the mapping induced by the kernel function and ξ is a vector of error variables. $\|.\|_2^2$ represents L_2-norm. e is a vector of ones of appropriate dimension and (w, b) are the parameters of the hyperplane that separates the two clusters. The parameter C is a trade-off factor and $l \geq 0$ is a user-defined constant that controls class imbalance. Since the constraint $y_i \in \{+1, -1\} \Leftrightarrow y_i^2 - 1 = 0$ is non-convex, (1) is a non-convex optimization problem. As discussed in [19], MMC relaxes the non-convex optimization problem and solves it as SDP. SDP is convex but computationally very expensive and can handle small data sets only. Zhang et al. proposed an iterative SVM approach to solve the MMC problem (1) based on alternating optimization [23].

2.2 Twin support vector machine for clustering (TWSVC)

TWSVC [24] is a plane-based clustering method which uses TWSVM classifier and follows OAA approach to determine K cluster center planes for a K-cluster problem. Since TWSVC considers all the data points (in OAA manner) for finding the cluster planes, it requires that each plane should be close to its own cluster and away from other clusters' data points on both the sides. For a K-cluster problem, let there be m data points $X = (x_1, x_2, ..., x_m)^T$ where $x_i \in \mathbb{R}^n$, with their corresponding labels in $\{1, 2, ..., K\}$; X is $m \times n$ matrix. Let the data for i^{th} cluster be represented by X_i and the data points of all clusters other than i^{th} cluster are given by $\widehat{X}_i$. For the K-cluster problem, TWSVC seeks K cluster center planes, which are given as

$$x^T w_i + b_i = 0, \qquad i = 1, 2, ..., K. \tag{2}$$

The planes are proximal to the data points of their own cluster. TWSVC uses initialization algorithm to get the initial cluster labels for data points and determines the initial cluster planes. The algorithm alternatively finds the labels of data points and cluster center planes until some termination condition is satisfied [24]. The cluster planes are obtained by considering the following problem, with initial cluster plane parameters $[w_i^0,\ b_i^0]$,

TWSVC:

$$\begin{aligned} \min_{w_i^{j+1}, b_i^{j+1}, \xi_i^{j+1}} \quad & \frac{1}{2}\|X_i w_i^{j+1} + eb_i^{j+1}\|_2^2 + ce^T\xi_i^{j+1} \\ \text{subject to} \quad & T(|\widehat{X}_i w_i^{j+1} + eb_i^{j+1}|) + \xi_i^{j+1} \geq e, \quad \xi_i^{j+1} \geq 0, \end{aligned} \tag{3}$$

where $i = 1, 2, ...K$ is index for the cluster and $j = 0, 1, 2, ...$ is the index of successive problem. $T(\cdot)$ denotes the first-order Taylor's series expansion and parameter c is the weight associated with the error vector. The optimization problem in (3) determines the i^{th} cluster center plane, which is required to be as close as possible to the i^{th} cluster X_i and far away from the other clusters' data points $\widehat{X}_i$ on both the sides. The problem also minimizes the error vector ξ_i which measures the error due to wrong assignment of cluster labels. By introducing the sub-gradient of $|\widehat{X}_i w_i^{j+1} + eb_i^{j+1}|$, (3) becomes

$$\begin{aligned} \min_{w_i^{j+1}, b_i^{j+1}, \xi_i^{j+1}} \quad & \frac{1}{2}\|X_i w_i^{j+1} + eb_i^{j+1}\|_2^2 + ce^T\xi_i^{j+1} \\ \text{subject to} \quad & diag(sign(\widehat{X}_i w_i^j + eb_i^j))(\widehat{X}_i w_i^{j+1} + eb_i^{j+1}) \geq e, \\ & \xi_i^{j+1} \geq 0, \end{aligned} \tag{4}$$

The solution of the above problem can be obtained by solving its dual problem [27] and is given by

$$\begin{aligned} \max_{\alpha} \quad & e^T\alpha - \frac{1}{2}\alpha^T G(H^T H)^{-1}G^T\alpha \\ \text{subject to} \quad & 0 \leq \alpha \leq ce, \end{aligned} \tag{5}$$

where $G = diag(sign(\widehat{X}_i w_i^j + b_i^j e))[\widehat{X}_i \quad e]$, $H = [X_i \quad e]$, and $\alpha \in \mathbb{R}^{m-m_i}$ is the Lagrangian multiplier vector. The problem in (5) is solved iteratively by concave-convex procedure (CCCP) [28], until the change in successive iterations is insignificant. The problem is extended to manifold clustering [24] by using kernel [29]. TWSVC uses an initialization algorithm [24] which is based on the nearest neighbour graph (NNG) and provides more stability to the algorithm.

2.2.1 Fuzzy least squares twin support vector clustering

Working on the lines of TWSVC [24] and least square SVM [30], Khemchandani et al. proposed F-LS-TWSVC [26] which is fuzzy based clustering method and its solution is obtained by solving a series of system of linear equations. Similar to TWSVC, F-LS-TWSVC uses OAA strategy to determine the K-cluster planes. It uses fuzzy nearest neighbour graph (FNNG) for label initialization and generates fuzzy membership matrix S of size $m \times K$. The i^{th} column of S gives the membership value of all data points in i^{th} cluster. The K clusters are obtained by solving following optimization problem:

$$\begin{aligned} \min_{w_i,\, b_i,\, q_i, X_i} \quad & \frac{1}{2}\|((S_i X_i)w_i + b_i e)\|_2^2 + \frac{\nu}{2}(\|w_i\|_2^2 + b_i^2) + \frac{C}{2}\|q_i\|_2^2 \\ \text{subject to} \quad & |((\overline{S_i} X_i)w_i + b_i e)| + q_i = e, \end{aligned} \tag{6}$$

where $i = 1, ..., K$. The diagonal matrices S_i and $\overline{S_i}$ indicate the fuzzy membership value of data points belonging to and not belonging to i^{th} cluster respectively [26]. This optimization problem can be solved using the concave-convex procedure(CCCP) similar to TWSVC. However, instead of solving QPPs as in TWSVC, the solution of F-LS-TWSVC is obtained by solving a series of system of linear equations.

2.2.2 Approximation used by TWSVC and F-LS-TWSVC

The shortcoming of both TWSVC and F-LS-TWSVC is that their primal formulations involve constraints with mod (|.|) function. To eliminate the mod function, they resort to Taylor's series expansion which gives an approximation of the original constraint. This condition is avoided in the proposed clustering algorithm by introducing the tree-based approach.

3 Tree-based localized fuzzy twin support vector clustering (Tree-TWSVC)

Taking motivation from MMC [19] and TWSVC [24], we propose Tree-TWSVC, which is an iterative tree-based clustering procedure that employs fuzzy membership matrix to create clusters using LF-TWSVM. The proposed algorithm can efficiently handle large multi-cluster datasets. For a K-cluster problem, it initially generates a fuzzy membership matrix for two clusters by using LF-NNG initialization algorithm (discussed in Section 3.2.4). Based on higher membership values, the data is partitioned into two clusters. Since membership values are based on the proximity of data points, the patterns of one cluster are similar to each other and distinct from the other cluster's patterns. Hence, Tree-TWSVC considers the inter-cluster and intra-cluster relationships. Each of the two clusters thus obtained can be recursively divided until K clusters are obtained. With each partition, the size of data is reduced which makes the procedure more time-efficient.

The proposed algorithm Tree-TWSVC starts with initial labels $(+1, -1)$, as generated by LF-NNG. By using the initial labels, the data X with m points is divided into two clusters, A and B, of size m_1 and m_2 respectively (where $m = m_1 + m_2$) as shown in Fig. 1. The group A can be further partitioned into A_1 and A_2, but Tree-TWSVC does not consider data points of B at this stage. This is because, in the first partition of dataset, the data points of A are separated from B, by considering inter-cluster relationship. In the second partition, the algorithm concentrates on the data points of A only and is able to generate more stable results in lesser time. The proposed algorithm is more efficient than plane-based clustering like TWSVC [24] and F-LS-TWSVC [26] that use classical OAA multi-category approach and the same is established by the results of numerical experiments in Section 4.

TWSVM [2] is initially proposed for classification problems and uses L_1 norm error function. When TWSVM-like formulation is used in clustering framework, as done in TWSVC [24], this could lead to premature convergence as the error function does not facilitate flipping of cluster labels, if required. The procedure gets stuck in a poor local

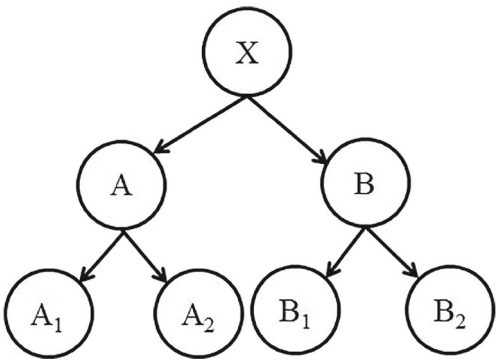


Fig. 1 Illustration of tree of clusters. Data is partitioned into two clusters at every internal node by iteratively generating cluster hyperplanes using LF-TWSVM

optimum and there is little or no change in the initial and final labels. This happens because the loss function is not symmetric and fails to change the labels in successive iterations (Please see Appendix A). To overcome this issue, we propose a new classifier LF-TWSVM, that efficiently handles the problem of premature convergence and is used to build the cluster model of Tree-TWSVC.

3.1 Proposed classifier: localized fuzzy TWSVM (LF-TWSVM)

In this work, we propose a novel classifier, termed as LF-TWSVM which we further use in an unsupervised framework. Unlike TWSVM, LF-TWSVM uses square loss function and a prototype. The prototype prevents the hyperplane from extending indefinitely and keeps it aligned *locally* to the data points.

3.1.1 LF-TWSVM: Linear version

Let the dataset X consist of m points in n-dimensional space. The data is divided into two clusters and hyperplanes are generated, which are given by

$$x^T w_1 + b_1 = 0, \qquad x^T w_2 + b_2 = 0. \tag{7}$$

LF-TWSVM employs the fuzzy membership matrix $F \in \mathbb{R}^{m\times 2}$ generated by LF-NNG and based on higher membership value, it partitions the data X into two clusters A (positive cluster) and B (negative cluster), of size m_1 and m_2 respectively. The planes for the two clusters A and B are obtained by solving the following problems:

$$\begin{aligned} \min_{w_1,b_1,\xi_2,A,v_1} \quad & \frac{1}{2}\|S_{AA}Aw_1 + e_1b_1\|_2^2 + \frac{c_1}{2}\|\xi_2\|_2^2 \\ & + \frac{c_2}{2}\|S_{AA}A - e_1v_1\|_2^2 + \frac{c_3}{2}(\|w_1\|_2^2 + b_1^2) \\ \text{subject to} \quad & -(S_{BA}Bw_1 + e_2b_1) + \xi_2 = e_2, \end{aligned} \tag{8}$$

$$\begin{aligned} \min_{w_2,b_2,\xi_1,B,v_2} \quad & \frac{1}{2}\|S_{BB}Bw_2 + e_2b_2\|_2^2 + \frac{c_1}{2}\|\xi_1\|_2^2 \\ & + \frac{c_2}{2}\|S_{BB}B - e_2v_2\|_2^2 + \frac{c_3}{2}(\|w_1\|_2^2 + b_2^2) \\ \text{subject to} \quad & (S_{AB}Aw_2 + e_1b_2) + \xi_1 = e_1. \end{aligned} \tag{9}$$

The diagonal matrices S_{AA} (size $(m_1 \times m_1)$) and S_{BA} (size $(m_2 \times m_2)$) define the membership value of data points of A and B respectively, in positive cluster, taken from matrix F. Similarly, the other two diagonal matrices S_{AB} and S_{BB} are defined for the negative cluster. The primal problems in (8) and (9) are motivated from TWSVM [2] and are modified on the lines of LS-TWSVM [4]. Thus, the inequality constraints are replaced with equality constraints and L_2-norm of error variables ξ_1 and ξ_2 is used; c_1 is the associated weight and e_1 , e_2 are vectors of one's of appropriate dimensions. The constraints of LF-TWSVM (8) and (9) do not require mod (|.|) function as required in the constraints of TWSVC (3). TWSVC determines the cluster planes using OAA multi-category approach and considers all the data points while finding the cluster planes. The data points of other clusters may lie on both sides of the cluster plane and hence, the constraints with mod function are required. For Tree-TWSVC, the data is divided into two clusters at each node, therefore one cluster would lie on only one side of another cluster. Hence, the constraints of Tree-TWSVC are written without mod function.

The first term in the objective function of (8) and (9) is the sum of squared distances of the hyperplane to the data points of its own cluster. Thus, minimizing this term tends to keep the hyperplane closer to the data points of one cluster (say cluster A) and the constraints require the hyperplane to be at unit distance from the points of another cluster (say cluster B). The error vectors ξ_1 and ξ_2 are used to measure the error if the hyperplane is not a unit distance away from data points of another cluster. The second term of the objective function minimizes the squared sum of error variables ξ_1 and ξ_2. The variable v_i $(i = 1, 2)$ is the prototype [18] of the i^{th} cluster and prevents the cluster plane from extending infinitely and controls its localization, proximal to the cluster. The parameter c_2 is weight associated with the proximal term. LF-TWSVM takes into consideration the principle of structural risk minimization (SRM) [31] by introducing the term $(w_i^T w_i + b_i^2, \quad i = 1, 2)$, in the objective function and thus improves the generalization ability. It also takes care of the possible ill-conditioning that might arise during matrix inversion. The parameter c_3 is chosen to be a very small value.

The error function of (8) and (9) is different from that of TWSVC (4) and has been modified for two major reasons. First, to allow flipping of labels during subsequent iterations, which is otherwise limited due to hinge loss function. This is required to minimize the total error. The second reason for using square loss function is that it leads to solving a system of linear equations instead of a QPP. After substituting the equality constraints in the objective function of (8) and (9), the problems become:

LF-TWSVM1:

$$\begin{aligned} \min_{w_1,b_1,A,v_1} \quad P_1 = {} & \frac{1}{2}\|S_{AA}Aw_1 + e_1b_1\|_2^2 + \frac{c_1}{2}\|S_{BA}Bw_1 + e_2b_1 + e_2\|_2^2 \\ & + \frac{c_2}{2}\|S_{AA}A - e_1v_1\|_2^2 + \frac{c_3}{2}((\|w_1\|_2^2 + b_1^2), \end{aligned} \tag{10}$$

LF-TWSVM2:

$$\begin{aligned} \min_{w_2,b_2,B,v_2} \quad P_2 = {} & \frac{1}{2}\|S_{BB}Bw_2 + e_2b_2\|_2^2 + \frac{c_1}{2}\|-(S_{AB}Aw_2 + e_1b_2) + e_1\|_2^2 \\ & + \frac{c_2}{2}\|S_{BB}B - e_2v_2\|_2^2 + \frac{c_3}{2}((\|w_2\|_2^2 + b_2^2). \end{aligned} \tag{11}$$

To get the solution of (10), set the gradient of P_1 with respect to w_1, b_1 and v_1 equal to zero. We get

$$\frac{\partial P_1}{\partial w_1} = 0 \Rightarrow (S_{AA}A)^T(S_{AA}Aw_1 + e_1b_1) + c_3w_1 + c_1(S_{BA}B)^T(S_{BA}Bw_1 + e_2b_1 + e_2) = 0e_1, \quad (12)$$

$$\frac{\partial P_1}{\partial b_1} = 0 \Rightarrow e_1^T(S_{AA}Aw_1 + e_1b_1) + c_3b_1 + c_1e_2^T(S_{BA}Bw_1 + e_2b_1 + e_2) = 0, \quad (13)$$

$$\frac{\partial P_1}{\partial v_1} = 0 \Rightarrow -c_2e_1^T(S_{AA}A - e_1v_1) = 0. \quad (14)$$

Let $E = [S_{AA}A \;\; e_1]$, $F = [S_{BA}B \;\; e_2]$ and $z_1 = [w_1 \;\; b_1]^T$, by combining (12) and (13) we obtain

$$E^TEz_1 + c_3z_1 + c_1F^TFz_1 + c_1F^Te_1 = 0e_1, \Rightarrow z_1 = -c_1(c_1F^TF + E^TE + c_3I)^{-1}F^Te_1. \quad (15)$$

Here, I is an identity matrix of appropriate dimensions. From (14),

$$v_1 = (e_1^TS_{AA}A)/(e_1^Te_1). \quad (16)$$

The second problem i.e. LF-TWSVM2 can be solved in similar manner. From (11), we get

$$G^TGz_2 + c_3z_2 + c_1H^THz_2 - c_1H^Te_2 = 0e_2, \Rightarrow z_2 = c_1(c_1H^TH + G^TG + c_3I)^{-1}H^Te_2, \quad (17)$$

where $G = [S_{BB}B \;\; e_2]$, $H = [S_{AB}A \;\; e_1]$ and $z_2 = [w_2 \;\; b_2]^T$. The prototype variable v_2 is obtained as

$$v_2 = (e_2^TS_{BB}B)/(e_2^Te_2). \quad (18)$$

The augmented vectors z_1 and z_2 can be obtained from (15) and (17) respectively and are used to generate the hyperplanes, as given in (7). The prototypes v_i for the two clusters can be calculated by using (16) and (18) respectively. A pattern $x \in \mathbb{R}^n$ is assigned to cluster i $(i = 1, 2)$, depending on which of the two hyperplanes given by (7) it lies closer to, i.e.

$$y = \underset{i}{argmin} \;\; (\|w_i^Tx + b_i\|_2^2 + c_2\|x - v_i\|_2^2). \quad (19)$$

It finds the distance of point x from the plane $x^Tw_i + b_i = 0$, where $i = 1, 2$ and also considers distance from the corresponding prototype. The predicted label for pattern x is given by y.

3.1.2 LF-TWSVM: Kernel version

The results can be extended to non-linear version by considering the kernel-generated surfaces and are given as

$$Ker(x^T, C^T)u_1 + b_1 = 0, \qquad Ker(x^T, C^T)u_2 + b_2 = 0, \quad (20)$$

where $C^T = [A \quad B]^T$ and Ker is an appropriately chosen positive definite kernel. The primal QPP of the non-linear LF-TWSVM corresponding to the first surface of (20) is given as

KLF-TWSVM1:

$$\min_{u_1,b_1,K_A,V_1} \; Q_1 = \frac{1}{2}\|K_Au_1 + e_1b_1\|_2^2 + \frac{c_1}{2}\|K_Bu_1 + e_2b_1 + e_2\|_2^2 + \frac{c_2}{2}\|K_A - e_1V_1\|_2^2 + \frac{c_3}{2}((\|u_1\|_2^2 + b_1^2), \quad (21)$$

where $K_A = S_{AA}Ker(A, C^T)$, $K_B = S_{BA}Ker(B, C^T)$. The solution for the problem (21) is obtained in similar manner as the linear case. The augmented vector $r_1 = [u_1 \;\; b_1]^T$ is given as

$$r_1 = -c_1(c_1K_F^TK_F + K_E^TK_E + c_3I)^{-1}K_F^Te_1. \quad (22)$$

Here, $K_E = [K_A \;\; e_1]$ and $K_F = [K_B \;\; e_2]$. The identity matrix I is of appropriate dimensions and the prototype V_1 is determined as

$$V_1 = (e_1^TK_A)/(e_1^Te_1). \quad (23)$$

The second plane can be retrieved in a similar manner from (24).

KLF-TWSVM2:

$$\min_{u_2,b_2,K_B,V_2} \; Q_2 = \frac{1}{2}\|K_Bu_2 + e_2b_2\|_2^2 + \frac{c_1}{2}\|-(K_Aw_2 + e_1b_2) + e_1\|_2^2 + \frac{c_2}{2}\|K_B - e_2V_2\|_2^2 + \frac{c_3}{2}((\|u_2\|_2^2 + b_2^2). \quad (24)$$

Here $K_A = S_{AB}Ker(A, C^T)$, $K_B = S_{BB}Ker(B, C^T)$. Once we obtain the surfaces (20), a new pattern $x \in \mathbb{R}^n$ is assigned to class 1 or class -1 in a manner similar to the linear case.

3.2 Clustering algorithms: Tree-TWSVC

Tree-TWSVC is a multi-category clustering algorithm that creates a binary tree of clusters by partitioning the data at multiple levels until the desired number of clusters are obtained. Tree-TWSVC uses an iterative approach to generate two cluster center planes, using LF-TWSVM at each node of the tree and updates the hyperplane parameters in each iteration by aligning the cluster plane along the data. Thus, it minimizes the empirical risk. Tree-TWSVC also minimizes structural risk due to the regularization term added to its formulation. In this work, we propose two implementations for Tree-TWSVC, namely BTree-TWSVC and OAA-Tree-TWSVC.

3.2.1 Binary tree-based localized fuzzy twin support vector clustering (BTree-TWSVC)

BTree-TWSVC is an unsupervised learning procedure that creates K clusters from m-data points. The algorithm takes

two inputs: $X \in \mathbb{R}^{m \times n}$ and K, where X represents m data points in n-dimension feature space and K is the number of clusters.

Algorithm 1 BTree-TWSVC

Input: The dataset X; the number of clusters K.
Output: Hyperplane parameters for internal and leaf nodes of the tree
Process:
1. Select the value for parameters - c_1, c_2, c_3, tol, kernel type and kernel parameter (only for non-linear case).
2. Determine the initial labels, Y_K, for the data points using K-means clustering [10] for K clusters.
3. Use LF-NNG to get the fuzzy membership matrix of all data points for two clusters, $F_2 \in \mathbb{R}^{m \times 2}$. Based on higher membership value, assign labels $(+1, \ -1)$ to each data sample and partition X into two clusters A and B. Also determine the diagonal matrices S_{AA}, S_{AB}, S_{BA}, S_{BB} from F_2. Here, S_{AA} and S_{BA} define the membership value of data points of A and B for positive cluster and S_{AB} and S_{BB} can be defined analogously.
4. Find the initial planes $[w_1, \ b_1]$ and $[w_2, \ b_2]$ for the two clusters by solving the (15) and (17) respectively. Get cluster prototypes v_1, v_2 from (16) and (18)
5. Repeat

a. Determine the distance of each data point from the two clusters and update the membership values as

$$F_{i,j}^{new} = \frac{1}{d_{i,j}}, \tag{25}$$

where $i = 1, ..., m$ and $j = 1, 2$. Here, $d_{i,j}$ represents the distance of i^{th} data point from j^{th} cluster plane and is given by

$$d_{i,j} = \|w_j^T x_i + b_j\|_2^2 + c_2\|x_i - v_j\|_2^2, \tag{26}$$

where $\|.\|_2^2$ is the L_2 $norm$.

b. Create two modified clusters as A^{new} and B^{new} based on F^{new}.
c. Update the hyperplanes $[w_1^{new}, \ b_1^{new}]$, $[w_2^{new}, \ b_2^{new}]$ and prototypes $v_1^{new}, \ v_2^{new}$ for the new clusters A^{new} and B^{new} respectively, by solving the (15)-(18).
d. If $\|F_2 - F^{new}\|_2^2 < tol$, then *break*.
e. $w_i = w_i^{new}, \ b_i = b_i^{new}, \ v_i = v_i^{new}, \ (i = 1, 2)$ and $F_2 = F^{new}$.

6. Use Y_K to determine if A^{new} and B^{new} can be further partitioned i.e. if there are labels from more than one clusters. If required, recursively partition A^{new} and B^{new} by calling
BTree-TWSVC(A^{new}, K_1)**;**
BTree-TWSVC(B^{new}, K_2)**;**
where $K = K_1 + K_2$.
7. End.

The Algorithm 1 for BTree-TWSVC generates the final solution in the form of clusters identified by LF-TWSVM, arranged as nodes of the tree. The root node contains the entire data and leaf nodes correspond to the final clusters. Thus, for a K-cluster problem, we obtain a tree with K leaf nodes and $(K - 1)$ internal nodes. Generally, most of the clustering algorithms like K-means [10] and KPC [17] initiate with randomly generated labels which leads to unstable results due to their dependency on the initial labels. For Tree-TWSVC, we use an initialization algorithm based on K-nearest neighbour graph [32], termed as localized fuzzy NNG (LF-NNG), discussed in Section 3.2.4.

BTree-TWSVC generates the fuzzy membership matrix $F_2 \in \mathbb{R}^{m \times 2}$ through LF-NNG and assigns either of the cluster labels $(+1, \ -1)$ to all data points based on higher membership value towards cluster 1 or -1 respectively. Then, the two cluster center planes are determined and the membership matrix F_2 is updated. BTree-TWSVC alternatively determines the cluster planes and membership matrix for data points until the convergence criterion is met (Step 5d) and the two clusters A^{new} and B^{new} are obtained. To decide whether the obtained clusters, A^{new} and B^{new}, can be further partitioned or not, the proposed algorithm uses K-means clustering [10] to get K clusters and labels $Y_k \in \{1, ..., K\}$ for all the data points. If clusters A^{new} or B^{new} are associated with more than one label from Y_k, then they can be further partitioned by recursively calling the same algorithm with new inputs. With the new inputs, size of the data is approximately reduced to half (assuming A^{new} and B^{new} contain approximately an equal number of data points), due to partitioning. Thus, the input data diminishes in size as we traverse down the cluster tree and creates a tree of height $\lceil log_2 K \rceil$.

3.2.2 One-against-all tree-based localized fuzzy twin support vector clustering (OAA-Tree-TWSVC)

OAA-Tree-TWSVC is another tree-based implementation of Tree-TWSVC and is explained in Algorithm 2.

Algorithm 2 OAA-Tree-TWSVC

Input: The dataset X; the number of clusters K.
Output: Hyperplane parameters for internal and leaf nodes of the tree **Process:** 1. Find a fuzzy membership matrix $F_K \in \mathbb{R}^{m \times K}$ through LF-NNG to get K initial clusters. Determine the initial labels, Y_{ini}, for the data points using F_K.
2. Select the value for parameters - c_1, c_2, c_3, tol, kernel type and kernel parameter (only for non-linear case).
3. All the data points with $Y_{ini} = 1$ are selected as patterns of positive cluster, whereas the rest of the points form the

negative cluster. Determine a new fuzzy membership matrix, for two clusters, $F_2 \in \mathbb{R}^{m\times 2}$ from F. Here, $F_2(j,1) = F_K(j,1)$ and $F_2(j,2) = \sum_{i=2}^{K} F_K(j,i)$ where $j = 1, ..., m$.
4. Based on higher membership value, get the initial labels $(+1, \ -1)$ from F_2 and partition the data into two sets A and B. Also determine the diagonal matrices S_{AA}, S_{AB}, S_{BA}, S_{BB}.
5. Find the initial planes $[w_1, \ b_1]$, $[w_2, \ b_2]$ and prototypes v_1, v_2 for the two clusters by solving the (15)–(18).
6. Repeat

1. Determine the distance of each data point from the two hyperplanes $[w_1, \ b_1]$ and $[w_2, \ b_2]$ and update the membership values as

$$F_{i,j}^{new} = \frac{1}{d_{i,j}}, \tag{27}$$

where $i = 1, ..., m$ and $j = 1, 2$. Here, $d_{i,j}$ represents the distance of i^{th} data point from j^{th} cluster and is given by

$$d_{i,j} = \|w_j^T x_i + b_j\|_2^2 + c_2\|x_i - v_j\|_2^2. \tag{28}$$

2. Create two modified clusters as A^{new} and B^{new} based on F^{new}.
3. Update the hyperplanes $[w_1^{new}, \ b_1^{new}]$, $[w_2^{new}, \ b_2^{new}]$ and prototypes $v_1^{new}, \ v_2^{new}$ for the new clusters A^{new} and B^{new} respectively, by solving the (15)-(18).
4. If $\|F_2 - F^{new}\|_2^2 < tol$, then *break*.
5. $w_i = w_i^{new}, \ b_i = b_i^{new}, \ v_i = v_i^{new}, \ (i = 1, 2)$ and $F_2 = F^{new}$.

7. Use Y_{ini} to determine if B^{new} can be further partitioned. If required, recursively partition B^{new} by calling **OAA-Tree-TWSVC**(B^{new}, F^{new}, $K - 1$).
(Here $F_{new} = F_K(i, j)$, i={Indexes of data points of B^{new}} and j=2,...,K.)
8. End.

The algorithm for OAA-Tree-TWSVC generates cluster model by arranging LF-TWSVM generated clusters in the form of a tree. At each internal node, one cluster is separated from rest of the clusters. Hence, this method represents modified one-against-all (OAA) multi-category strategy. The height of the tree is $(K - 1)$.

3.2.3 Binary tree (BTree) vs. One-against-all tree (OAA-Tree)

In this work, we have proposed two implementations for Tree-TWSVC as discussed in Section 3.2.1 and 3.2.2. Out of the two approaches, BTree-TWSVC is more robust and achieves better clustering accuracy than OAA-Tree-TWSVC which is experimentally proved in Section 4. One such scenario is presented in Fig. 2 i.e. clustering problem with four clusters (a.). Here, we present the clustering result with TWSVC, OAA-Tree-TWSVC and BTree-TWSVC. For TWSVC, the hyperplanes are obtained using OAA strategy (b.) and this leads to an ambiguous region i.e. the data points lying in this region might be wrongly clustered. With OAA-Tree-TWSVC, one of the clusters obtained with LF-NNG initialization is selected as the positive cluster (green squares) and the remaining are regarded as the negative cluster (red frames, violet triangles and blue dots), as shown in (c.). The localized hyperplanes are generated using LF-TWSVM, but it still leads to some ambiguity. Once the green cluster is identified, we apply the same procedure on *remaining* data points, as presented in (d-e.).The final OAA-tree obtained is shown in (e.). For BTree-TWSVC, LF-NNG is used to identify two clusters at a time, as demonstrated in (f.), which separates the blue-violet points from red-green points. BTree-TWSVC is able to generate a stable clustering model as depicted in (f-h.) and has got better clustering ability.

3.2.4 Initialization with localized fuzzy nearest neighbour graph (LF-NNG)

Wang et al. proposed NNG based initialization method for TWSVC [24] and F-LS-TWSVC [26] used fuzzy NNG (FNNG) to generate the membership values. For this work, we propose localized fuzzy nearest neighbour graph (LF-NNG) which generates a membership matrix F. This matrix is used to obtain the initial data labels to be used by Tree-TWSVC. The steps involved in LF-NNG for getting initial labels of K clusters are given in Algorithm 3.

Algorithm 3 Localized fuzzy nearest neighbour graph (LF-NNG) based cluster membership

Input: The dataset X; the number of clusters K; nearest neighbors p.
Output: F matrix
Process:
1. For the given data set $X \in \mathbb{R}^{m\times n}$ and a parameter p, construct p nearest neighbour undirected graph where edges represent the distance between the pattern x_i (i=1,...,m) and its p nearest neighbors.
2. From the graph, identify t clusters $(C_1, ..., C_t)$ by associating the nearest samples i.e. neighbors must fall in the same cluster.
3. If the current number of clusters t is equal to K, then construct a fuzzy membership matrix $F_{i,j}$ where $i = 1, ...m$ and $j = 1, ...t$ where $F_{i,j}$ is given as

$$F_{i,j} = \frac{1}{d_{i,j}}.$$

Here $d_{i,j}$ is the distance of i^{th} sample from j^{th} cluster prototype and is given by

$$d_{i,j} = \|x_i - v_j\|_2^2, \tag{29}$$

where v_j is the cluster prototype and is given as

$$v_j = (e^T C_j)/(e^T e),$$

and C_j represents the data points in j^{th} cluster. Go to Step 6.

Else, go to Step 4 or 5.

4. If $t < k$, disconnect the two connected samples with the maximum distance and go to Step 2.

5. If $t > k$, compute the Hausdorff distance [33] between every two clusters among the t clusters and sort all pairs in ascending order. Merge the nearest pair of clusters into one, until k clusters are formulated, where the Hausdorff distance between two sets S_1 and S_2 is defined as

$$h(S_1, S_2) = max\{\underset{i\in S_1}{max}\{\underset{j\in S_2}{min}||i-j||\}, \{\underset{i\in S_2}{max}\{\underset{j\in S_1}{min}||i-j||\}\}. \tag{30}$$

6. End.

3.2.5 Complexity analysis

The strength of Tree-TWSVC is the tree-based approach which reduces the complexity of the algorithm. The size of data diminishes as it is partitioned to obtain the clusters. This characteristic is of utmost importance for non-linear (kernel) classifiers where the complexity is dependent on the size of data. For a K-cluster problem, the OAA multi-category approach uses entire dataset K-times to determine the cluster planes. Assuming that all clusters have equal size i.e. m/K, where m is the number of data points. If any TWSVM-based classifier is used with OAA (as done in TWSVC), then the algorithm solves K QPPs, each of size $((K-1)/K) * m$. Hence, the complexity of TWSVM-based clustering algorithm is given by

$$\begin{aligned} T_{OAA} &= K * c * \left(\tfrac{K-1}{K} * m\right)^3, \\ &\simeq K * c * m^3, \end{aligned} \tag{31}$$

where c is a constant that includes the count for maximum number of iterations for finding the final cluster planes. So, the complexity of OAA TWSVM-based clustering is $T_{OAA} = O(m^3)$.

In BTree-TWSVC, the optimization problem is solved as a system of linear equations. For the linear case, LF-TWSVM finds the inverse of two matrices, each of dimension $(n+1) \times (n+1)$, where n is the number of features, for each internal node of the binary tree. As we traverse down the tree, size of the data is approximately reduced to half. Thus, the complexity of BTree-TWSVC can be recursively defined as

$$\begin{aligned} T(m) &= c(n+1)^3 + 2 * T\left(\tfrac{m}{2}\right), \\ T\left(\frac{m}{K}\right) &= 1, \end{aligned} \tag{32}$$

where m is the number of data points and c is the complexity constant. We assume that data is divided into two clusters of almost equal size. The base condition $T\left(\frac{m}{K}\right) = 1$ represents cost of leaf node that contains data from one cluster only. The time complexity of (32) is given as [34]

$$\begin{aligned} T(m) &= c(n+1)^3 + 2 * c(n+1)^3 \\ &\quad + ... + 2^{h-1} * c(n+1)^3 + 2^h * 1, \end{aligned} \tag{33}$$

where $h = \lceil log_2 K \rceil$. The height of the tree 'h' depends on the number of clusters K. The above equation can be solved as

$$\begin{aligned} T(m) &= c(n+1)^3(1 + 2 + 4 + ... + 2^{h-1}) + 2^h, \\ &= c(n+1)^3(2^h - 1) + 2^h, \\ &= c(n+1)^3(K-1) + K. \\ &\leq cK(n+1)^3 + K. \end{aligned} \tag{34}$$

Therefore, the complexity of linear BTree-TWSVC implemented as a binary tree (BT) is $T_{BT} = O(Kn^3)$ and is independent of the size of data. For large-sized datasets ($m \gg n$), the efficiency of BTree-TWSVC is not much affected, but for TWSVC (implemented using OAA-TWSVM) the learning time increases with size of data.

For kernel version, the complexity of BTree-TWSVC can be recursively defined as

$$\begin{aligned} T(m) &= c(m+1)^3 + 2 * T(\tfrac{m}{2}), \\ T\left(\frac{m}{K}\right) &= 1, \end{aligned} \tag{35}$$

where m is the number of data points and c is the complexity constant. The complexity (35) can be written as [34]

$$\begin{aligned} T(m) = c(m+1)^3 &+ \frac{1}{4}c(m+1)^3 + \frac{1}{16}c(m+1)^3 \\ &+ ... + \frac{1}{4^{h-1}}c(m+1)^3 + 2^h, \end{aligned} \tag{36}$$

where $h = \lceil log_2 K \rceil$. The above equation can be solved as

$$\begin{aligned} T(m) &= c(m+1)^3(1 + \frac{1}{4} + \frac{1}{16} + ... + \frac{1}{4^{h-1}}) + 2^h, \\ &\leq \frac{4}{3}c(m+1)^3 + K, \\ &\simeq \frac{4}{3}c(m+1)^3. \end{aligned} \tag{37}$$

So, the complexity of kernel BTree-TWSVC is independent of the number of clusters K. BTree-TWSVC is more time-efficient than OAA multi-category clustering for both linear and kernel versions. We can discuss the time complexity of OAA-Tree-TWSVC in a similar way as TWSVC as both of them are based on OAA strategy. But OAA-Tree-TWSVC

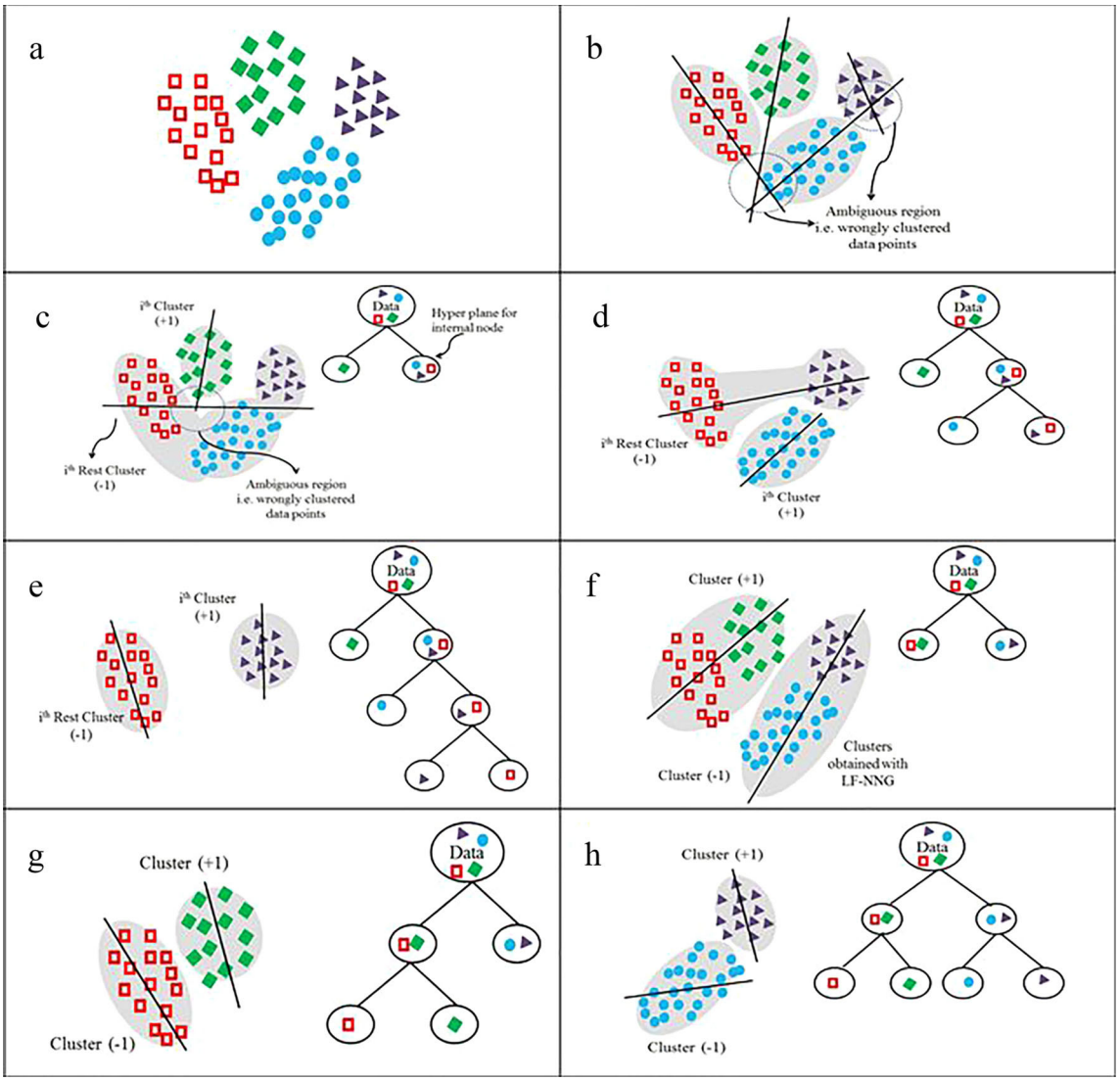


Fig. 2 Clustering with TWSVC and Tree-TWSVC for four clusters **(a)**. Dataset with four clusters **(b.)** Clustering model with TWSVC **(c-e.)** Clustering with OAA-Tree-TWSVC and resulting OAA-tree (**f–h**). Clustering with BTree-TWSVC and resulting binary tree

is more time-efficient than TWSVC because the number of data points get diminished as we traverse down the OAA-tree. To validate the efficiency of the proposed method, we have compared the learning time of OAA-Tree-TWSVC with TWSVC (also based on OAA strategy) in Section 4.

3.3 Discussion

In this section, we have discussed the comparison of proposed clustering algorithm with MMC [23], TWSVC [24] and F-LS-TWSVC [26].

3.3.1 Tree-TWSVC vs. MMC

The proposed clustering algorithm Tree-TWSVC determines multiple clusters by solving a system of linear equations, whereas MMC is a binary clustering method that solves a non-convex optimization problem which is relaxed to solving expensive SDP. Unlike MMC, Tree-TWSVC does not use alternate optimization to determine w and b, whereas they are obtained as vector $z_i = [w_i \;\; b_i]^T$, $(i = 1, 2)$, by solving a system of linear (15) and (17). Therefore, Tree-TWSVC is more time-efficient than MMC. Also, Tree-TWSVC uses fuzzy nearest neighbour based initialization method, which improves its clustering accuracy.

3.3.2 Tree-TWSVC vs. TWSVC

TWSVC involves constraints with mod function $(|.|)$ and it uses Taylor's series expansion to get an approximation of this function. Whereas Tree-TWSVC considers only

two clusters at each level of the tree and these clusters would lie on either side of the mean cluster plane, hence it does not require constraints with mod function. Also, Tree-TWSVC formulation involves square loss function which results in solving a series of systems of linear equations whereas TWSVC solves a series of QPPs using the concave-convex procedure. Therefore, Tree-TWSVC is more efficient in terms of computational effort as well as clustering accuracy than TWSVC. Also, TWSVC is based on OAA strategy, but Tree-TWSVC uses tree-based approach. Tree-TWSVC uses a better initialization algorithm (i.e. LF-NNG) and its decision function for test data also takes into account the distance from the cluster prototype. Hence, Tree-TWSVC achieves better results in lesser time than TWSVC.

3.3.3 Tree-TWSVC vs. F-LS-TWSVC

F-LS-TWSVC solves a series of systems of linear equations to get the cluster planes, but similar to TWSVC, it uses Taylor's series approximation for the constraints and therefore the results may not be accurate. Tree-TWSVC formulates a convex optimization problem which is solved as a series of systems of linear equations. F-LS-TWSVC is based on OAA multi-category strategy and Tree-TWSVC handles OAA using tree-based approach, which is more efficient. Also, Tree-TWSVC has BTree-TWSVC approach which is even faster than OAA-Tree-TWSVC.

4 Experiments and results

In this section, we compare the performance of two variations of Tree-TWSVC i.e. BTree-TWSVC and OAA-Tree-TWSVC, with other clustering methods and investigate their accuracy and computational efficiency. The other clustering methods used for comparison are Fuzzy C-means (FCM) clustering [35], TWSVC [24] and F-LS-TWSVC [26]. We have also implemented a non-fuzzy version of OAA-Tree-TWSVC, which is referred as OAA-T-TWSVC. These two algorithms are compared to study the effect of adding fuzziness to the clustering model. For OAA-T-TWSVC, the initial clusters are generated using NNG [32]. The experiments are performed in MATLAB version 8.0 under Microsoft Windows environment on a machine with 3.40 GHz CPU and 16 GB RAM. The experiments are conducted on benchmark UCI datasets [36]. In all experiments, the focus is on the comparison of proposed method with clustering methods listed above. The parameters c_1 and c_2 are selected in the range {0.01 to 1}; $c_3 \in \{10^{-i},\ i = 1, ..., 5\}$ and tol is selected to be very small value of order 10^{-5}. The kernel parameter is tuned in the range {0.1 to 1}. The grid search method [37] is applied to tune the parameters. For each dataset, a validation set comprising of 10% randomly selected samples from the dataset is used.

The metric *Accuracy* [38] is used to measure the performance of clustering methods. For finding the accuracy of clustering algorithm, a similarity matrix $S \in \mathbb{R}^{m \times m}$ is computed with the given data labels $y_i,\ y_j \in \{1 : K\},\ i = 1 : m,\ j = 1 : m$, where

$$S(i, j) = \begin{cases} 1, & if\ y_i = y_j \\ 0, & otherwise. \end{cases}$$

Let S_t and S_p be the similarity matrices computed by the true cluster labels and predicted labels respectively. The accuracy of clustering method is defined as the Rand statistic [38] and is given as

$$Accuracy = \frac{n_{zeros} + n_{ones} - m}{m^2 - m} \times 100 \tag{38}$$

where n_{zeros} is the number of zeros at corresponding indices in both S_t and S_p, and n_{ones} is the number of ones in both S_t and S_p.

4.1 Out-of-sample testing

In an unsupervised framework, generally the clustering model is built using an entire dataset. But, the formulation of proposed Tree-TWSVC allows it to obtain the clustering model with learning data and the accuracy of the model can be examined using out-of-sample (OoS) or unseen test data [39, 40]. The clustering model is built with some part of learning data provided as input to the Tree-TWSVC algorithm and is used to predict the labels of the unseen OoS data. This feature is particularly useful when working with very large datasets where the clustering model can be built with few samples and rest of the samples are assigned labels using OoS approach. Tree-TWSVC also takes advantage of the tree structure and LF-TWSVM formulation and generates the results in much less time. In our simulations with UCI dataset, we have given the results with entire dataset and OoS testing of clustering model. For OoS, 80 % samples are randomly selected from the entire data for learning the model and the remaining 20 % are used to determine the clustering accuracy.

4.2 Experiments on UCI datasets

We have selected 14 UCI multi-category datasets [36] for the experiments- Zoo, Iris, Wine, Seeds, Segment, Glass, Dermatology, Ecoli, Compound, Libra, Optical digits, Page-blocks, Satimage and Pen digits. The number of samples, features and classes are shown along with the datasets in Table 1.

Table 1 Clustering accuracy for UCI datasets (Linear version)

Data	K-means	FCM	TWSVC	F-LS-TWSVC	OAA-T-TWSVC	Tree-TWSVC	
						OAA-Tree	BTree
(m×n, K)	Accuracy (%)						
Zoo (101× 16, 7)	92.92	85.70	88.20	92.16	95.00	95.23	**95.49**
Iris (150 × 4, 3)	87.37	89.88	89.88	94.61	89.23	93.33	**95.33**
Wine (178 × 13, 3)	90.38	89.18	73.46	88.65	89.82	90.06	**90.84**
Seeds (210× 7, 3)	85.20	83.93	75.14	86.74	86.36	88.02	**91.20**
Segment (210 × 19, 7)	82.91	71.43	77.29	82.65	84.28	86.86	**88.56**
Glass (214×9, 6)	68.14	54.21	68.08	69.02	69.29	71.07	**73.25**
Dermatology (366 × 34, 6)	92.00	55.89	82.31	91.44	93.33	93.79	**94.30**
Ecoli (336×7, 8)	82.25	79.59	83.60	**86.24**	80.90	84.05	83.93
Compound (399×2, 6)	82.68	82.85	86.53	88.70	87.38	89.06	**90.06**
Libra (360 × 90, 15)	80.53	64.82	88.06	90.14	85.26	89.12	**92.76**
Large Datasets							
Pageblocks (5473 × 10, 5)	80.09	90.50	62.35	81.01	86.03	91.78	**92.56**
Optical digits (5620 × 64, 9)	73.47	42.15	48.45	80.17	78.74	81.76	**82.44**
Satimage (6435 × 36, 7)	78.05	73.07	59.95	75.29	73.96	**80.65**	79.18
Pen digits (10992 × 16, 9)	63.62	59.74	50.25	63.45	66.07	66.26	**68.78**
Average Accuracy	81.40	73.07	73.83	83.61	83.26	85.79	**87.05**

4.2.1 Results for linear case

The simulation results for UCI datasets with linear clustering methods are recorded in Table 1 for K-means, FCM, TWSVC, F-LS-TWSVC, OAA-T-TWSVC, OAA-Tree-TWSVC and BTree-TWSVC. The simulation results demonstrate that both versions of Tree-TWSVC i.e. BTree-TWSVC and OAA-Tree-TWSVC, outperform K-means, FCM, TWSVC and F-LS-TWSVC for clustering accuracy. In Table 1, the entire dataset is used for building the clustering model. For 13 out of 14 UCI datasets, one of the two versions of Tree-TWSVC achieves the highest accuracy. This can be attributed to the fact that a good initialization algorithm can improve the accuracy of the clustering algorithm. It is also observed that binary tree-based algorithm (BTree-TWSVC) generates better result than OAA-Tree-TWSVC.

Table 2 OoS Clustering accuracy for UCI datasets (Linear version)

Data	OAA-T-TWSVC	OAA-Tree-TWSVC	BTree-TWSVC
	Accuracy (%)		
Zoo	93.23	93.16	**93.18**
Iris	86.29	91.28	**93.56**
Wine	87.16	88.52	**89.71**
Seeds	81.84	87.65	**88.90**
Segment	81.51	83.75	**85.27**
Glass	65.26	65.72	**72.82**
Dermatology	90.56	**91.84**	91.14
Ecoli	78.19	**82.65**	79.61
Compound	85.88	86.34	**86.41**
Libra	82.45	88.45	**91.02**
Large Datasets			
Pageblocks	83.25	87.33	**88.73**
Opt.digits	75.28	77.86	**79.16**
Satimage	71.87	**78.52**	77.29
Pendigits	63.84	64.52	**65.29**
Average Accuracy	80.47	83.40	**84.43**

Table 3 Clustering accuracy for UCI datasets (Non-linear version)

Data	K-means	LkPPC	FCM	TWSVC	F-LS-TWSVC	OAA-T-TWSVC	OAA-Tree-TWSVC	BTree-TWSVC
	Accuracy (%)							
Zoo	91.56	96.83	83.17	89.18	95.14	96.15	**97.31**	97.13
Iris	89.88	94.19	91.33	92.67	96.66	92.55	95.68	**97.83**
Wine	91.09	95.93	91.57	95.59	94.66	95.43	**96.27**	96.13
Seeds	80.95	92.09	86.52	84.76	88.37	86.62	88.11	**92.85**
Segment	78.33	86.45	70.95	80.32	84.61	85.49	86.35	**88.87**
Glass	71.52	72.95	55.61	69.04	70.96	69.56	71.87	**73.56**
Dermatology	90.13	**96.28**	87.45	86.71	93.22	93.81	94.26	95.30
Ecoli	80.80	86.76	77.37	85.45	**90.17**	84.83	87.40	87.40
Compound	88.63	92.37	81.45	96.19	95.38	93.52	94.47	**96.43**
Libra	90.83	92.06	77.89	90.08	92.01	91.10	92.86	**93.64**
Large Datasets								
Pageblocks	89.29	75.20	92.56	64.01	82.38	91.69	93.65	**94.51**
Opt.digits	65.85	85.35	55.29	45.28	82.14	86.72	88.59	**91.69**
Satimage	78.35	78.36	78.82	77.29	81.02	80.94	87.42	**88.69**
Pendigits	69.61	**75.82**	63.85	53.94	62.27	66.29	73.51	73.56
Average Accuracy	82.63	87.19	78.13	79.32	86.35	86.76	89.12	**90.54**

The table demonstrates that OAA-Tree-TWSVC (fuzzy version) achieves better clustering accuracy than OAA-T-TWSVC (non-fuzzy version). Table 2 shows accuracy result with OoS clustering for OAA-T-TWSVC, OAA-Tree-TWSVC and BTree-TWSVC. The clustering algorithms achieve better results when entire data is used for clustering.

4.2.2 Results for non-linear case

The proposed method is extended using non-linear LF-TWSVM classifier and Table 3 compares the performance of Tree-TWSVC (both versions) with that of TWSVC, F-LS-TWSVC, LkPPC, K-means and FCM using RBF kernel,

Table 4 OoS Clustering accuracy for UCI datasets (Non-linear version)

Data	OAA-T-TWSVC	OAA-Tree-TWSVC	BTree-TWSVC
	Accuracy (%)		
Zoo	93.72	94.67	**95.80**
Iris	89.10	92.85	**95.09**
Wine	93.72	**95.03**	94.51
Seeds	84.50	87.91	**89.03**
Segment	82.11	84.82	**86.95**
Glass	65.42	67.55	**72.88**
Dermatology	91.12	92.17	**92.54**
Ecoli	82.49	**84.52**	84.66
Compound	88.50	90.28	**91.56**
Libra	90.46	91.14	**92.19**
Large Datasets			
Pageblocks	87.51	88.75	**90.03**
Opt.digits	83.46	86.15	**87.79**
Satimage	76.22	84.91	**86.34**
Pendigits	64.27	68.59	**71.94**
Average Accuracy	83.76	86.38	**87.95**

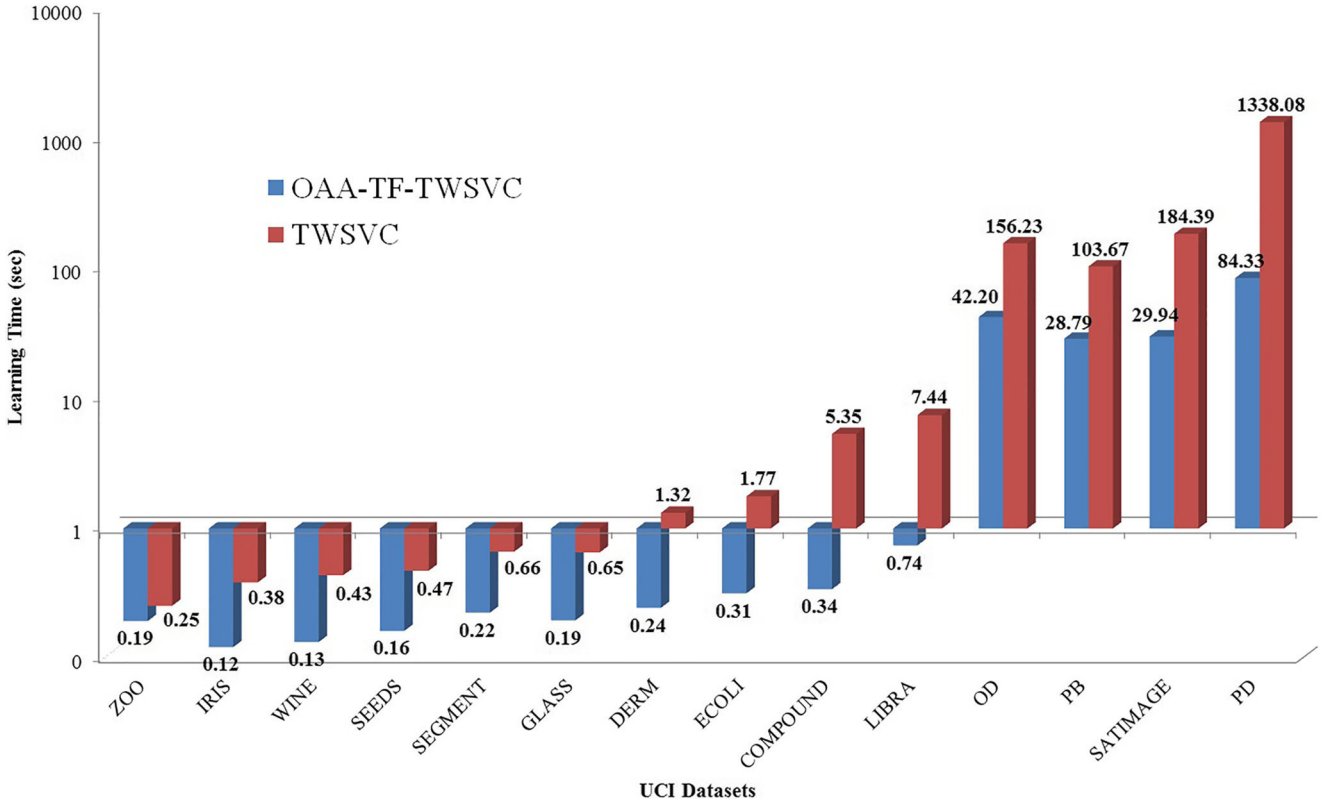


Fig. 3 Learning time (Linear)

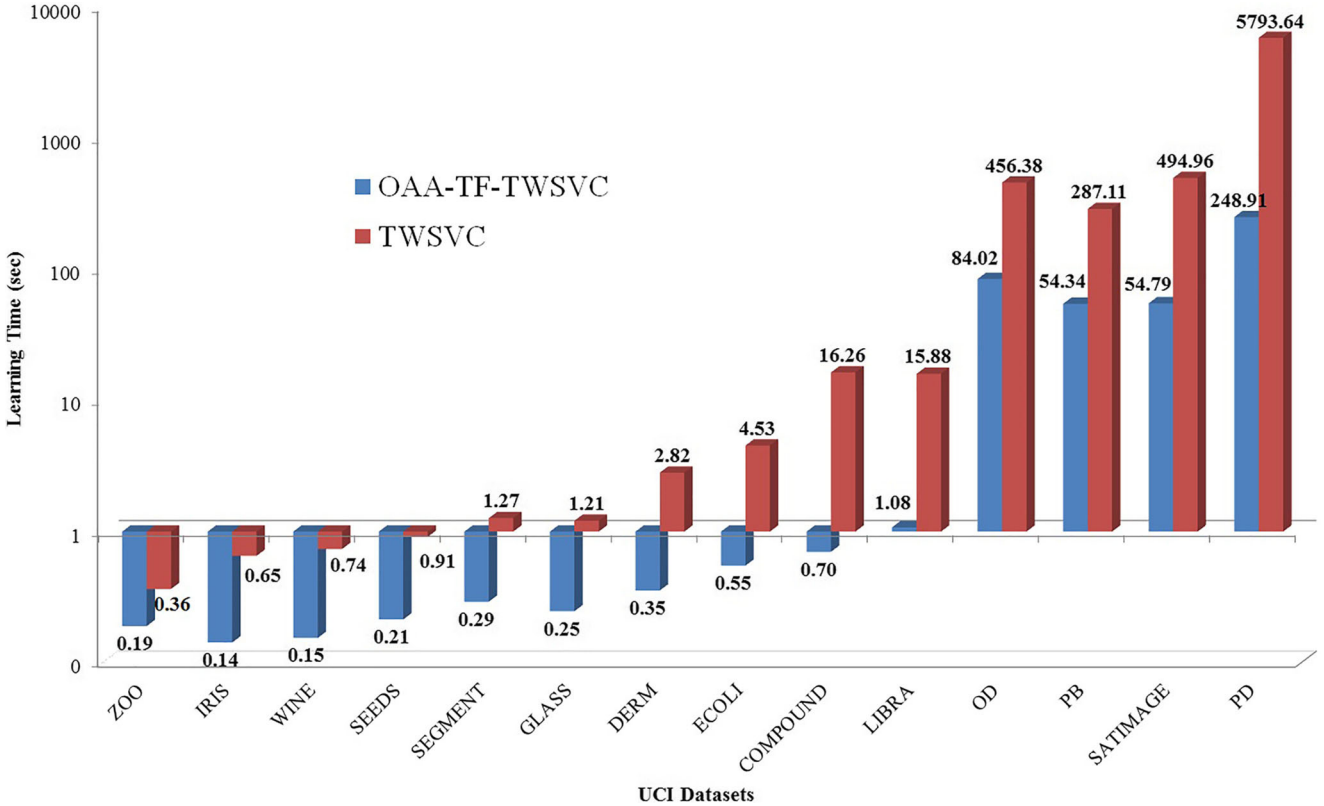


Fig. 4 Learning time (Non-linear)

$Ker(x, x') = exp(-\sigma\|x - x'\|^2)$. The table shows the clustering accuracy of these algorithms on UCI datasets. The results illustrate that Tree-TWSVC (both versions) achieve better accuracy for most of the datasets. It is also observed that the clustering results are better for non-linear versions as compared to linear ones. Table 4 shows accuracy result with OoS clustering for non-linear versions of OAA-T-TWSVC, OAA-Tree-TWSVC and BTree-TWSVC.

4.3 Learning time:

We have compared the learning time (i.e. time for building the clustering model) of OAA-Tree-TWSVC with TWSVC for UCI datasets in Fig. 3. In this figure, Derm, OD, SI, PB and PD represent Dermatology, Optical digits, Satimage, Pageblocks and Pen digits datasets respectively. Although, both of these clustering methods are based on OAA multi-category strategy, but OAA-Tree-TWSVC takes much less time for building the tree-based model than TWSVC. The efficiency of OAA-Tree-TWSVC is significant for datasets with large number of classes i.e. Libra, Compound, Satimage and Pen digits, where OAA-Tree-TWSVC is much faster than TWSVC. For pen digits dataset, OAA-Tree-TWSVC is almost 16 times faster than TWSVC. The learning time of non-linear versions of OAA-Tree-TWSVC and TWSVC are compared in Fig. 4. It is observed that OAA-Tree-TWSVC is very efficient in dealing with large datasets; whereas learning time of TWSVC is highly affected by size and number of classes in the dataset.

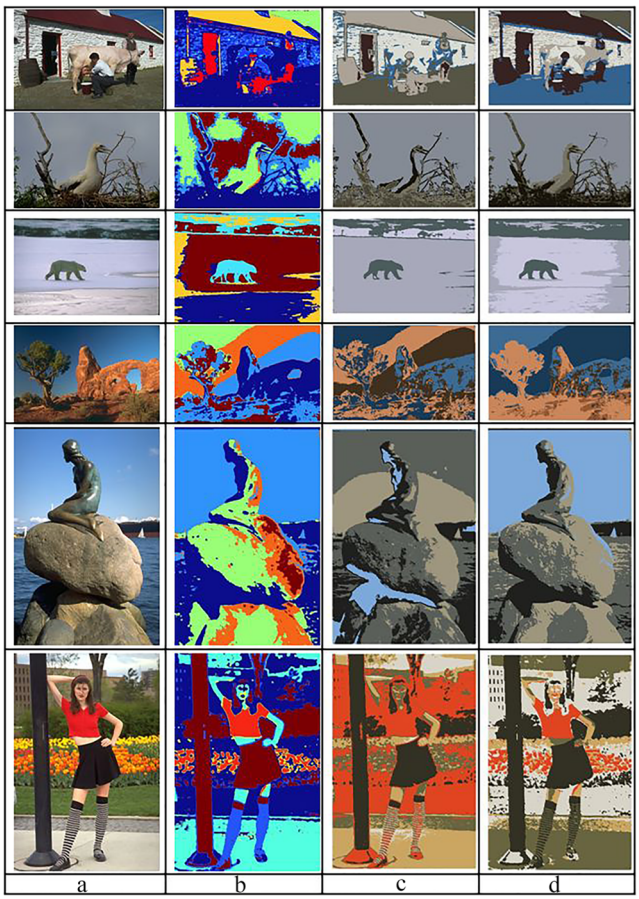


Fig. 5 Segmentation results on color images from BSD image dataset **(a.)** Original image **(b.)** MSS-KSC [43] **(c.)** TWSVC **(d.)** BTree-TWSVC

4.4 Experiments on large datasets

In order to demonstrate the scalability and effectiveness of Tree-TWSVC, we performed experiments on large UCI datasets i.e. Optical digits, Satimage, Pen digits and Page-blocks. It is observed that the performance of TWSVC deteriorates as the size of data increases; whereas Tree-TWSVC can efficiently handle large datasets. Similarly, FCM fails to give good accuracy for Pen digits and optical digits. From Table 1, there is a significant difference in the clustering accuracy achieved by Tree-TWSVC (both versions) as compared to FCM and TWSVC, for the above mentioned large datasets. Tree-TWSVC scales well on these datasets and is not much affected by the number of classes.

4.5 Application to image segmentation

To evaluate the performance of Tree-TWSVC on large datasets, we present its application on image segmentation which can be considered as a clustering problem. The image is partitioned into non-overlapping regions that share certain homogeneous features. For the experiments, we have taken color images from Berkeley image segmentation dataset (BSD) [41]. These images have dimensions 481×321 or 321×481 i.e. 154,401 pixels in each image. The problem is to partition the image pixels into disjoint sets called the regions. We use a dynamic method to determine the number of regions for each image. The histogram for the image is generated and the prominent peaks are identified. The number of prominent peaks determines the number of regions (L) in the image. The color image is then partitioned using minimum variance color quantization with L levels. For the experiments, we have taken a combination of color and texture features. The image features used for this work are Gabor texture features [42] and RGB color value of the pixel. Gabor features are extracted with 4-orientation $(0, 45, 90, 135)$ and 3-scale $(0.5, 1.0, 2.0)$ sub-bands and the maximum of the 12 coefficients determine the orientation at a given pixel location.

The segmentation model is built using OoS approach with BTree-TWSVC and 1 % pixels are randomly selected from the image for learning. The rest of the image pixels are used for testing the model. The images are segmented using BTree-TWSVC and the results are compared with linear TWSVC and multiclass semi-supervised kernel

Table 5 Segmentation result for BSD color images

Image	L	F-measure			Error rate		
		MSS-KSC	TWSVC	BTree-TWSVC	MSS-KSC	TWSVC	BTree-TWSVC
385039	5	0.49	0.52	**0.69**	0.0726	0.0818	**0.0499**
8049	4	0.71	0.63	**0.78**	0.0784	0.0800	**0.0676**
100007	3	0.57	0.57	**0.66**	0.0774	0.0798	**0.0463**
295087	5	0.62	0.69	**0.76**	0.0910	0.0844	**0.0527**
372019	4	0.49	0.47	**0.54**	0.0624	0.0755	**0.0420**
388067	5	0.62	0.65	**0.76**	0.0980	0.1185	**0.0887**
55067	3	0.58	0.55	**0.61**	0.0214	0.0234	**0.0201**
113044	3	0.71	0.63	**0.73**	0.0348	0.0400	**0.0312**
118035	3	0.72	0.69	**0.74**	0.0513	0.0473	**0.0431**
124084	3	0.54	0.48	**0.69**	0.0637	0.0818	**0.0577**
161062	4	0.62	0.58	**0.74**	0.0343	0.0639	**0.0168**
198023	4	0.57	0.58	**0.78**	0.0235	0.0363	**0.0228**
388016	3	0.46	0.41	**0.62**	0.0806	0.1369	**0.0490**
51084	4	0.66	0.64	**0.68**	0.0695	0.0743	**0.0613**
196027	4	0.63	0.45	**0.67**	0.0294	0.0359	**0.0159**

spectrum clustering (MSS-KSC) [43] segmentation methods, as shown in Fig. 5. MSS-KSC uses few labelled pixels to build the clustering model with kernel spectrum clustering approach. It is observed that the segmentation results of BTree-TWSVC are visually more accurate than other algorithms. For TWSVC, the image is over-segmented, which results in the formation of multiple smaller regions within one large region. For BSD images, the ground truth segmentations are known and the images segmented by BTree-TWSVC and TWSVC are compared with ground truth. To statistically evaluate the segmentation algorithms, two evaluation criteria are used: F-measure (FM) and error rate (ER). F-measure is determined as

$$FM = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{39}$$

and ER is given by

$$ER = \frac{FP + FN}{\text{Total pixels}}, \tag{40}$$

where $Precision$ and $Recall$ are defined as

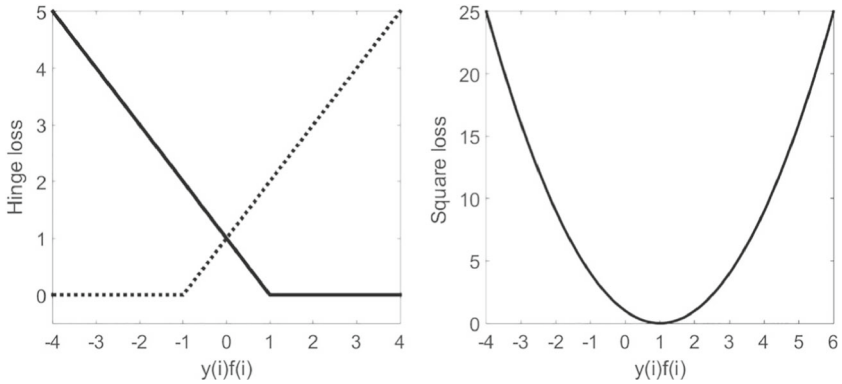
$$Precision = \frac{TP}{TP + FP},$$
$$Recall = \frac{TP}{TP + FN}.$$

TP, FP, TN, FN are true-positive, false-positive, true-negative and false-negative respectively. These measures are calculated with respect to ground-truth boundaries and results are presented in Fig. 5 and Table 5. BTree-TWSVC achieves better F-measure and error rate values than TWSVC and MSS-KSC.

5 Conclusions

In this paper, we propose Tree-based localized fuzzy twin support vector clustering (Tree-TWSVC) which is an iterative algorithm and extends the proposed classifier, localized fuzzy twin support vector machine (LF-TWSVM), in unsupervised framework. LF-TWSVM is a binary classifier that generates the nonparallel hyperplanes by solving a system of linear equations. Tree-TWSVC develops a tree-based clustering model which consists of several LF-TWSVM classifiers. In this work, we propose two implementations of Tree-TWSVC, namely Binary Tree-TWSVC and One-against-all Tree-TWSVC. The proposed clustering algorithm outperforms the other TWSVM-based clustering methods like TWSVC and F-LS-TWSVC, which are based on classical one-against-all multi-category approach and they use Taylor's series for approximating the constraints of the optimization problem. Experimental results have proved that Tree-TWSVC has superior clustering accuracy and efficient learning time for UCI datasets as compared to FCM, TWSVC and F-LS-TWSVC. The proposed clustering algorithm is extended for image segmentation problems also. The future line of work is to validate the performance of the proposed algorithm with different distance measures for calculating fuzzy memberships.

Acknowledgments We would like to take this opportunity to thank Dr.Suresh Chandra, for his guidance and constant encouragement throughout the preparation of the manuscript.



(a) Hinge loss function (b) Square loss function

Fig. 6 Flipping of labels. **a.** Hinge loss function; **b.** Square loss function

Appendix A: Loss function of TWSVM

TWSVM uses hinge loss function which is given by

$$L_h = \begin{cases} 0, & y_i f_i \geq 1 \\ 1 - y_i f_i, & otherwise \end{cases}$$

For any SVM or TWSVM based clustering method, the clustering error or the hyperplanes change little after initial labeling or during subsequent iterations. This arises due to hinge loss function, as shown in Fig. 6a, where the classifier tries to push $y_i f_i$ to the point beyond $y_i f_i = 1$ (towards right) [23]. Here, solid line shows loss with initial labels and dotted line shows loss after flipping of labels. As observed from the empirical margin distribution of $y_i f_i$, most of the patterns have margins $y_i f_i \gg 1$. If the label of a pattern is changed, the loss will be very large and the classifier is unwilling to flip the class labels. So, the procedure gets stuck in a local optimum and adheres to the initial label estimates. To prevent the premature convergence of the iterative procedure, the loss function is changed to square loss and is given as $L_s = (1 - y_i f_i)^2$. This loss function is symmetric around $y_i f_i = 1$, as shown in Fig. 6b and penalizes preliminary wrong predictions. Therefore, it permits flipping of labels if needed and leads to a significant improvement in the clustering performance.

References

1. Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Networks 10(5):988–999
2. Jayadeva, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
3. Khemchandani R (2008) Mathematical programming applications in machine learning, Ph.D. dissertation. Indian Institute of Technology Delhi New Delhi-110016, India
4. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
5. Sartakhti JS, Ghadiri N, Afrabandpey H (2015) Fuzzy least squares twin support vector machines. arXiv: 1505.05451
6. Tanveer M, Khan MA, Ho SS (2016) Robust energy-based least squares twin support vector machines. Appl Intell:1–13
7. Shao YH, Wang Z, Chen WJ, Deng NY (2013) Least squares twin parametric-margin support vector machine for classification. Appl Intell 39(3):451–464
8. Khemchandani R, Pal A (2016) Multi-category laplacian least squares twin support vector machine. Appl Intell 45(2):458–474
9. Hastie T, Tibshirani R, Friedman J (2009) Unsupervised learning. Springer
10. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recogn Lett 31(8):651–666
11. Jain AK, Dubes RC (1998) Algorithms for clustering data. Prentice-Hall Inc.
12. Ng AY, Michael IJ, Yair W (2002) On spectral clustering: Analysis and an algorithm. Adv Neural Inf Proces Syst:849–856
13. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
14. Wu W, Xiong H, Shekhar S (eds.) (2013) Clustering and information retrieval (Vol. 11) Springer Science and Business Media
15. Moon TK (1996) The expectation-maximization algorithm. IEEE Signal Process Mag 13(6):47–60
16. Al-Harbi SH, Rayward-Smith VJ (2006) Adapting k-means for supervised clustering. Appl Intell 24(3):219–226
17. Bradley PS, Mangasarian OL (2000) k-plane clustering. J Global Optim 16(1):23–32
18. Yang ZM, Guo YR, Li CN, Shao YH (2015) Local k-proximal plane clustering. Neural Comput & Applic 26(1):199–211
19. Xu L, Neufeld J, Larson B, Schuurmans D (2004) Maximum margin clustering. Adv Neural Inf Proces Syst 17:1537–1544
20. Valizadegan H, Jin R (2006) Generalized maximum margin clustering and unsupervised kernel learning. Adv Neural Inf Proces Syst:1417–1424
21. Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge university press
22. Lobo MS, Vandenberghe L, Boyd S (1998) Applications of second-order cone programming. Linear Algebra Appl 284(1):193–228
23. Zhang K, Tsang IW, Kwok JT (2009) Maximum margin clustering made practical. IEEE Trans Neural Networks 20(4):583–596
24. Wang Z, Shao YH, Bai L, Deng NY (2015) Twin support vector machine for clustering. IEEE Transactions Neural Networks and Learning Systems 26(10):2583–2588
25. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. IEEE Trans Neural Networks 13(2):415–425
26. Khemchandani R, Pal A (2016) Fuzzy least squares twin support vector clustering. Accepted by Neural computing and applications

27. Mangasarian OL (1993) Nonlinear programming. SIAM 10
28. Yuille AL, Rangarajan A (2003) The concave-convex procedure. Neural Comput 15(4):915–936
29. Smola AJ, Scholkopf B (1998) Learning with kernels. Citeseer
30. Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. Neural Process Lett 9(3):293–300
31. Gunn SR (1998) Support vector machines for classification and regression. ISIS technical report 14
32. Larose DT (2005) k-nearest neighbor algorithm. Discovering Knowledge in Data: An Introduction to Data Mining: 90-106
33. Huttenlocher DP, Klanderman GA, Rucklidge WJ (1993) Comparing images using the hausdorff distance. IEEE Trans Pattern Anal Mach Intell 15(9):850–863
34. Cormen TH (2009) Introduction to algorithms. MIT press
35. Wang X, Wang Y, Wang L (2004) Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recogn Lett 25(10):1123–1132
36. Blake C, Merz CJ (1998) Uci repository of machine learning databases. Available: www.ics.uci.edu
37. Hsu CW, Chang CC, Lin CJ (2003) A practical guide to support vector classification
38. Tan PN, Steinbach m, Kumar V (2005) Introduction to data mining. Addison-Wesley
39. Alzate C, Suykens JAK (2010) Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA. IEEE Trans Pattern Anal Mach Intell 32(2):335–347
40. Alzate C, Suykens JAK (2011) Out-of-sample eigenvectors in kernel spectral clustering. In: The 2011 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 2349–2356
41. Arbelaez P, Fowlkes C, Martin D (2007) The berkeley segmentation dataset and benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds
42. Khan JF, Adhami RR, Bhuiyan SM (2009) A customized gabor filter for unsupervised color image segmentation. Image Vis Comput 27(4):489–501
43. Mehrkanoon S, Alzate C, Mall R, Langone R, Suykens JAK (2015) Multiclass semi-supervised learning based upon kernel spectral clustering. IEEE Transactions on Neural Networks and Learning Systems 26(4):720–733

Reshma Rastogi (nee Khemchandani) earned her M.Sc. degree in Mathematics from Indian Institute of Technology, Delhi, India, in 2003 and Ph.D. degree in Machine Learning from the Indian Institute of Technology Delhi, India, in 2008. Currently, she is an Assistant Professor in the Department of Computer Science, South Asian University, Delhi, India. Her research interests include machine learning, image processing, financial modelling and optimization. She has published over 30 papers in refereed international journals and international conferences.

Pooja Saigal received her MCA degree from Guru Gobind Singh Indraprestha University, Delhi, India in 2004 and BCA degree from Maharishi Dayanand University, India in 2001. She is awarded gold medal by the Honble President of India for securing first position in University in MCA. She is currently pursuing the Ph.D. degree in machine learning with the Department of Computer Science, South Asian University, India. Her research interests include optimization, machine learning, pattern recognition, image retrieval and image segmentation.