2-Sided Best Fitting Hyperplane Classifier

Hakan Cevikalp
Electrical and Electronics Engineering Department
Machine Learning and Computer Vision Laboratory
Eskisehir Osmangazi University, Meselik, 26480, Eskisehir, Turkey
http://www2.ogu.edu.tr/~mlcv/, hakan.cevikalp@gmail.com

***Abstract*—In this paper, we propose a novel method that is more appropriate than classical large-margin classifiers for open set recognition and object detection problems. The proposed method uses the best fitting hyperplanes approach, and the main idea is to find the best fitting hyperplanes such that each hyperplane is close to the samples of one of the two classes and as far as possible from the other class samples. As opposed to the most common hyperplane fitting classifiers in the literature, the proposed classifier allows the negative samples to lie on both sides of the fitting hyperplane and hence it is based on a non-convex optimization problem. We use concave-convex procedure to solve this non-convex problem. Then, the method is extended to the nonlinear case by using the kernel trick. The proposed method is also suitable for large-scale problems, and it returns sparse solutions in contrast to the other hyperplane fitting methods in the literature. The experiments on several databases show that our proposed method typically outperforms other hyperplane fitting classifiers in term of classification accuracy, and it performs as good as the SVM classifier if not any better.**

Keywords—classifier; open set recognition; hyperplane fitting; kernel methods; support vector machines.

I. INTRODUCTION

Large margin classifiers have been successfully used in many fields including computer vision, text analysis, biometrics and bioinformatics. The prototypical method of this kind, the Support Vector Machine (SVM) [1] finds a linear hyperplane in feature space that maximizes the "margin" – the Euclidean distance between the hyperplane and the closest training samples of each class. The resulting optimization task requires the minimization of a convex quadratic function subject to linear inequality constraints, and this problem can be efficiently solved by various methods [2,3,4]. The solution is sparse in the sense that once the closest points (so-called support vectors) have been found, it depends only on them.

Recently, Mangasarian and Wild [5] proposed the first hyperplane fitting classifier so-called generalized eigenvalue proximal support vector machine (GEPSVM). GEPSVM finds two non-parallel hyperplanes (as opposed to the parallel hyperplanes returned by the Proximal SVMs [6]) by solving two generalized eigenvalue problems so that each hyperplane best fits to the corresponding class samples, and at the same time, it is as far as possible from the other class samples. Once the best fitting hyperplanes are found, the new samples are classified based on the minimum distances to the returned hyperplanes. By using a similar idea, Jayadeva et al. [7] proposed the Twin Support Vector Machine (TSVM) classifier. This classifier also aims at finding two non-parallel hyperplanes such that each hyperplane is closer to the one of the two classes and is as far as possible from the other. However, TSVM solves a pair of quadratic programming (QP) problems instead of a generalized eigenvalue problem. It has been reported that the total training time of TSVM takes less time than training time of SVM classifier since TSVM solves a pair of smaller sized QP problems instead of a large QP problem as in SVM. Shao et al. [8] further improved the TSVM classifier by introducing a regularization term on the hyperplane parameters. Kumar and Gopal [9] proposed the least squares version of the TSVM and smooth TSVM [10]. Some other extensions of TSVM can also be found in [11,12].

Classifiers based on the best-fitting hyperplanes have been proposed as alternatives to the binary SVM classifier, but there are better application areas of these methods in recognition problems known as "***open set recognition"*** [13]. In classical classification problems, it is assumed that all testing classes are known at training time. But, in more realistic applications, samples may come from unknown classes during testing time. Margin-based classifiers such as SVM or affine hull based classifier [14] seek to maximize the distance between the known class samples and the decision boundary. Regions far from the known data (it is called as open space in [13]) are also assigned to the known classes although we do not have a good basis for assigning labels to these regions. As a result, these classifiers may largely fail during testing when there are samples coming from unknown classes. This is illustrated in Fig. 1. On the other hand, as we show in this study, the classifiers using the best fitting hyperplanes are more appropriate for these kinds of applications. They are also more suitable than large margin classifiers for visual object detection tasks, in which there is a limited number of object class samples whereas there are millions of negative samples coming from thousands of different classes [15].

Although classifiers using the best fitting hyperplane models have wider application areas compared to the large-margin classifiers, the current methods in the literature are not perfect. More precisely, there are two major limitations of the hyperplane fitting classifiers, GEPSVM and TSVM: As a first limitation, these classifiers are not suitable for large-scale classification problems. Kernel GEPSVM requires eigen-decomposition on $(n+1)\times(n+1)$ matrices where n is the number of all samples in the training set. Similarly, kernel TSVM requires taking inverse of a $(n+1)\times(n+1)$ matrix. For small size classification problems, it is reported that training

times of GEPSVM and TSVM take less time compared to the training time of SVM classifier. But, for large scale problems, it is difficult (most of the time it is impossible) to fit those large matrices into memory and to operate on them. The second limitation is related to the sparsity of the solution. As opposed to the SVM classifier, the solution returned by the GEPSVM or TSVM is not sparse, i.e., all training samples become support vectors. Therefore, testing times of those classifiers are much slower compared to the testing time of SVM. There has been an attempt to modify TSVM method so that it returns sparse solutions in [16]. However, it is based on the assumption that the best fitting hyperplane is constructed by the support vectors coming from only one class samples. This assumption does not hold in many classification problems since the best fitting discriminating hyperplanes are determined by the samples lying in the critical regions where the two-class boundaries are close to each other.

In this paper, we introduce a novel classifier that uses the best fitting hyperplane approach. Our method does not have limitations of both GEPSVM and TSVMs. In particular, the proposed method is suitable for large-scale classification problems, and it always returns sparse solutions. It is also better suited for open set recognition problems as well as object detections problems. The rest of the paper is organized as follows. In Section 2, we introduce the proposed method. Section 3 presents our experimental results and Section 4 concludes the paper.

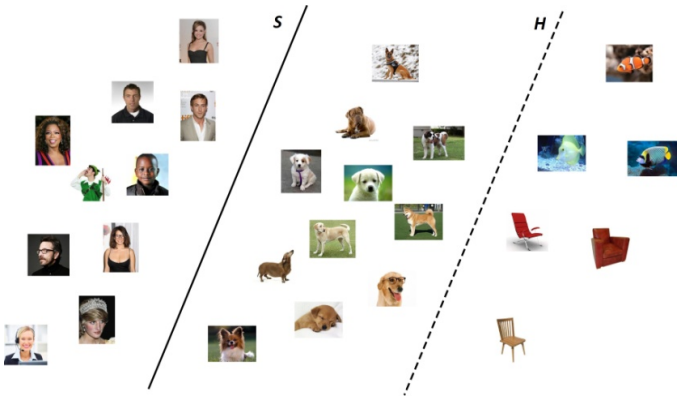

Fig. 1. The separating hyperplane ***S*** returned by the SVM classifier separates people and dog classes. All samples under the separating hyperplane are assigned to the dog class. When there are test samples coming from the unknown classes such as chair and fish, these samples will be erroneously assigned to the dog class with high confidence scores. Adding another parallel hyperplane ***H*** helps to localize dog class samples better, and misclassifications can be reduced.

II. METHOD

We propose a new classification method that uses the best fitting hyperplanes for pattern classification. As opposed to the TSVM and its extensions, the proposed method allows the negative samples to lie on both sides of the fitting hyperplane and it requires solving a more complicated non-convex optimization problem as described below.

A. Two-Sided Best Fitting Hyperplane Classifier (2S-BFHC)

This method searches for the best fitting hyperplanes such that each hyperplane is closer to the samples of one of the two classes and far from the other class samples. Let $\mathbf{x}$ be the sample's feature vector and let $\mathbf{w}_+^T\mathbf{x}+b_+=0$ be the equation of the best fitting hyperplane for the positive class samples. The proposed method allows the negative samples to lie on both sides of the fitting hyperplanes separated from the positive samples with a margin of at least $2\Delta/\|\mathbf{w}_+\|$ as shown in Fig. 2, where Δ is a parameter that must be set between 0 and 1 by the user. However, this problem is no longer convex and it is typically difficult to solve for large-scale data. To solve this non-convex problem for large scale data, we employ concave-convex procedure [17].

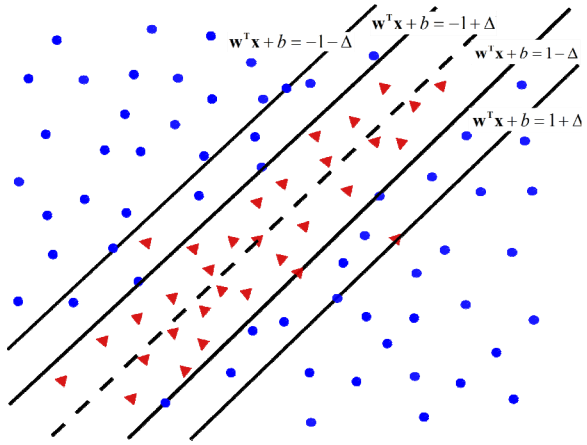

Fig. 2. In the proposed method, positive class samples (shown with red triangles) lie between two parallel hyperplanes characterized by $\mathbf{w}^T\mathbf{x}+b=1-\Delta$ and $\mathbf{w}^T\mathbf{x}+b=-1+\Delta$. The negative samples shown with blue circles can lie on both sides of the fitting hyperplane (fitting hyperplane, $\mathbf{w}^T\mathbf{x}+b=0$, is shown with the dashed line), and they are separated from the positive samples with a margin of at least $2\Delta/\|\mathbf{w}\|$ in separable case.

Let us define the decision function we will use for label assignment in the form

$$f_\theta(\mathbf{x})=\mathbf{w}_+^T\mathbf{x}+b_+, \qquad (1)$$

where $\theta=(\mathbf{w}_+,b_+)$ includes the parameters that define the best fitting hyperplane for the positive class samples. In the proposed method, we constrain positive samples to lie between two parallel hyperplanes $\mathbf{w}_+^T\mathbf{x}+b_+=1-\Delta$ and $\mathbf{w}_+^T\mathbf{x}+b_+=-1+\Delta$. To implement this goal, we use the symmetric Ramp Loss function (shown in Fig. 3) defined as

$$J_{pos}(t)=R_{pos}(t)+R_{pos}(-t), \qquad (2)$$

where $R_{pos}(t)=\min(1-s,\max(0,-1+\Delta-t))$ is the so-called "Ramp Loss" function [18] illustrated in Fig. 4, where $-1<s\le 0$ is a parameter that must be set by the user. As shown in the figure, the Ramp Loss function can be written as the sum of the convex Hinge loss and a concave loss function (or as the difference between two Hinge losses), i.e., $R_{pos}(t)=H_{-1+\Delta}(t)-H_{s-2+\Delta}(t)$, where $H_a(t)=\max(0,a-t)$ is the

classical Hinge loss function. The Ramp loss function can be seen as a "clipped" version of the Hinge loss, and the parameter s controls where we clip the Ramp Loss. We set it to $s = -0.20$ in our experiments.

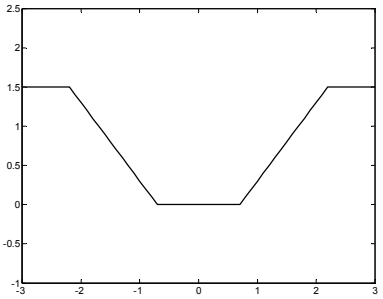

Fig. 3. Illustration of the cost function, $J_{pos}(t) = R_{pos}(t) + R_{pos}(-t)$, of positive samples. $R_{pos}(t) = \min(1-s, \max(0, -1+\Delta-t))$ can be written as the sum of the convex Hinge loss and a concave loss function, i.e., $R_{pos}(t) = H_{-1+\Delta}(t) - H_{s-2+\Delta}(t)$. Δ is set to 0.3 and $s = -0.20$.

The negative samples are forced to lie outside of the hyperplane shaped slabs, and they are separated from the positive samples with a margin of at least $2\Delta / \| \mathbf{w}_+ \|$. Thus the constraints for negative samples become $| \mathbf{w}_+^T \mathbf{x}_i + b_+ | > 1 + \Delta$. To implement this, we use another symmetric Ramp Loss (shown in Fig. 5) defined as

$$J_{neg}(t) = R_{neg}(t) + R_{neg}(-t), \tag{3}$$

where $R_{neg}(t) = \min(1+\Delta-s, \max(0, 1+\Delta-t))$ is illustrated in Fig. 4. Similar to the previous case, this Ramp Loss function can also be written as the sum of the convex Hinge loss and a concave loss function i.e., $R_{neg}(t) = H_{1+\Delta}(t) - H_s(t)$. For this loss function, s parameter controls the wideness of the flat part of the symmetric part of the symmetric Ramp Loss plotted in Fig. 5.

In order to use the symmetric Ramp Loss functions defined for positive and negative samples, each sample in the training set appears as two examples labeled with both negative and positive classes. More precisely, if we let n_+ be the number of positive samples, and n_- be the number of negative samples under the assumption that the positive and negative samples are ordered, then we create the new samples as follows

$$\begin{aligned}
&y_i = 1, \ i \in [1,...,n_+]; \ y_i = -1, \quad i \in [n_+ + 1,...,2n_+] \\
&\mathbf{x}_i = \mathbf{x}_{i-n_+}, \ i \in [n_+ + 1,...,2n_+] \\
&y_i = -1, \ i \in [2n_+ + 1,...,2n_+ + n_-]; \ y_i = 1, \ i \in [2n_+ + n_- + 1,...,2n_+ + 2n_-] \\
&\mathbf{x}_i = \mathbf{x}_{i-n_-}, \ i \in [2n_+ + n_- + 1,...,2n_+ + 2n_-]
\end{aligned}$$

In this case our total cost function can be written as

$$J(\theta) = \frac{1}{2} \| \mathbf{w}_+ \|^2 + C_+ \sum_{i=1}^{2n_+} R_{pos}(y_i f_\theta(\mathbf{x}_i)) + C_- \sum_{i=2n_++1}^{2n_++2n_-} R_{neg}(y_i f_\theta(\mathbf{x}_i)). \tag{4}$$

Here, $C_+ (C_-)$ is a user defined parameter that controls the weight of the errors associated with the positive (negative) samples. By using the equations $R_{pos}(t) = H_{-1+\Delta}(t) - H_{s-2+\Delta}(t)$ and $R_{neg}(t) = H_{1+\Delta}(t) - H_s(t)$, the above cost function can be written as

$$J(\theta) = J_{convex}(\theta) + J_{concave}(\theta), \tag{5}$$

where

$$J_{convex}(\theta) = \frac{1}{2} \| \mathbf{w}_+ \|^2 + C_+ \sum_{i=1}^{2n_+} H_{-1+\Delta}(y_i f_\theta(\mathbf{x}_i)) + C_- \sum_{i=2n_++1}^{2n_++2n_-} H_{1+\Delta}(y_i f_\theta(\mathbf{x}_i)), \tag{6}$$

and

$$J_{concave}(\theta) = -C_+ \sum_{i=1}^{2n_+} H_{s-2+\Delta}(y_i f_\theta(\mathbf{x}_i)) - C_- \sum_{i=2n_++1}^{2n_++2n_-} H_s(y_i f_\theta(\mathbf{x}_i)). \tag{7}$$

Since the cost function $J(\theta)$ can be decomposed into a convex and a concave part, we can apply the concave-convex procedure (CCCP) to solve the problem. The CCCP solves this non-convex optimization problem by an iterative procedure, where each iteration approximates the concave part by its tangent, and minimizes the resulting convex function as given in Algorithm 1. The convergence of CCCP has been proved in [17]. It should be noted that the convex optimization problem that constitutes the core of the CCCP algorithm can be solved by using efficient convex algorithms.

Algorithm 1: The Concave-Convex Procedure

Initialize θ^0

repeat

$$\theta^{t+1} = \arg\min_\theta (J_{convex}(\theta) + J'_{concave}.\theta)$$

until convergence of θ^t.

Now, in order to simplify the first order approximation of the concave part of the optimization problem, let us define

$$\begin{aligned}
\beta_i^0 &= y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} \\
&= \begin{cases} C_+, & \text{if } y_i f_\theta(\mathbf{x}_i) < s - 2 + \Delta \text{ and } 1 \le i \le 2n_+ \\ C_-, & \text{if } y_i f_\theta(\mathbf{x}_i) < s \text{ and } 2n_+ + 1 \le i \le 2n_+ + 2n_- \end{cases}
\end{aligned} \tag{8}$$

After some standard derivations shown in the Appendix (it is available at http://mlcv.ogu.edu.tr/pdf/appendix.pdf), the method can be summarized as in Algorithm 2.

Algorithm 2: Two Sided Best Fitting Hyperplane Classifier

Initialize $\theta^0 = (\mathbf{w}_+^0, b_+^0)$

compute

$$\begin{aligned}
\beta_i^0 &= y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} \\
&= \begin{cases} C_+ & \text{if } y_i f_\theta(\mathbf{x}_i) < s - 2 + \Delta \text{ and } 1 \le i \le 2n_+ \\ C_- & \text{if } y_i f_\theta(\mathbf{x}_i) < s \text{ and } 2n_+ + 1 \le i \le 2n_+ + 2n_- \end{cases}
\end{aligned}$$

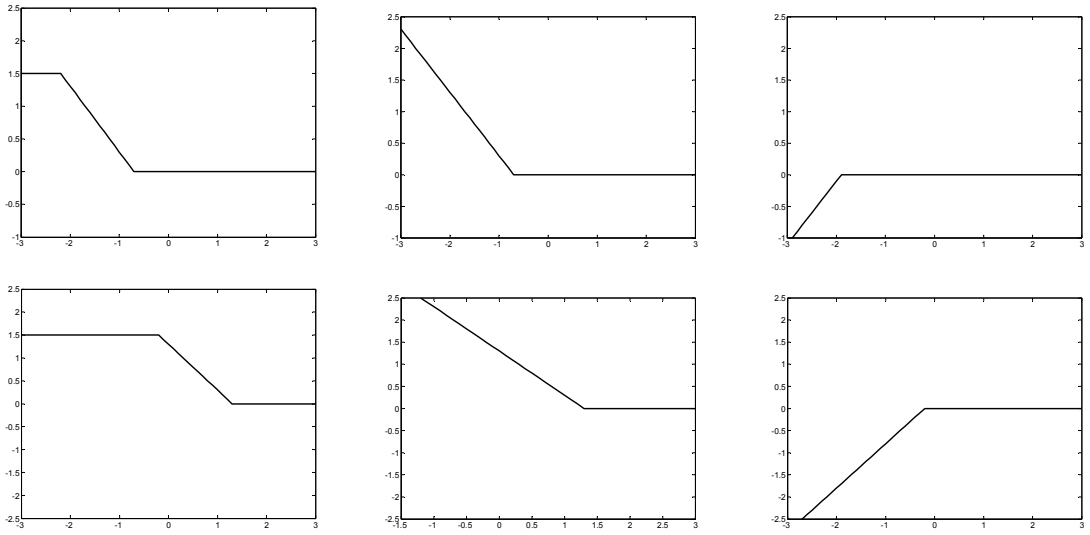

Fig. 4. The illustrations of Ramp Loss functions $R_{pos}(t) = \min(1-s, \max(0, -1+\Delta-t))$ (top left figure) and $R_{neg}(t) = \min(1+\Delta-s, \max(0, 1+\Delta-t))$ (bottom left). Each loss can be written as sum of the convex Hing loss (center) and the concave loss (right), i.e., $R_{pos}(t) = H_{-1+\Delta}(t) - H_{s-2+\Delta}(t)$, and $R_{neg}(t) = H_{1+\Delta}(t) - H_s(t)$, where $H_a(t) = \max(0, a-t)$ is the classical Hinge loss. Here, the parameters are set to $s = -0.20$ and $\Delta = 0.30$.

repeat

- Solve the following convex quadratic optimization problem

$$\min \ \frac{1}{2}\sum_{i=1}^{2n_+ + 2n_-}\sum_{j=1}^{2n_+ + n_-} h_i h_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle + (1-\Delta)\sum_{i=1}^{2n_+} y_i h_i - (1+\Delta)\sum_{i=2n_++1}^{2n_+ + 2n_-} y_i h_i$$

$$\text{subject to} \sum_{i=1}^{2n_+ + 2n_-} h_i = 0; \ -\beta_i \le y_i h_i \le C_+ - \beta_i, \ 1 \le i \le 2n_+;$$

$$-\beta_i \le y_i h_i \le C_- - \beta_i, \ 2n_+ + 1 \le i \le 2n_+ + 2n_-.$$

- **compute** $\mathbf{w}_+^{t+1}$ by using $\mathbf{w}_+^{t+1} = \sum_{i=1}^{2n_+ + 2n_-} h_i \mathbf{x}_\mathbf{i}$ and b_+^{t+1} as described in the Appendix.
- **compute**

$$\beta_i^{t+1} = \begin{cases} C_+ & \text{if } \ y_i f_\theta(\mathbf{x}_i) < s - 2 + \Delta \ \text{ and } 1 \le i \le 2n_+ \\ C_- & \text{if } \ y_i f_\theta(\mathbf{x}_i) < s \ \text{ and } 2n_+ + 1 \le i \le 2n_+ + 2n_- \end{cases}$$

until $\| \mathbf{w}_+^{t+1} - \mathbf{w}_+^t \| \le \varepsilon_1$ or $\| \boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t \| \le \varepsilon_2$, where ε_1 and ε_2 are some pre-defined small thresholds.

Note that the convex part of the algorithm becomes a convex quadratic optimization with equality and inequality constraints, and it can be efficiently solved by using Sequential Minimal Optimization (SMO) [2] for large-scale data. More precisely, it is not necessary to construct the full Hessian matrix: only the Hessians of the active sets of samples need to be considered in each iteration. Our proposed method usually takes 3-5 iterations to converge a solution based on the initialization of the algorithm. It should be noted that only the bounds on h_i have to be adjusted after each update of $\boldsymbol{\beta}$.

Another advantage of the proposed method is that the solution returned by the optimization problem is sparse, i.e., the most of the h_i coefficients are zero. Therefore 2S-BFHC is more suitable than GEPSVM or TSVM for real-time classification applications where the speed is important.

The second fitting hyperplane is also obtained by the same procedure by interchanging the roles of positive and negative samples. For multi-class classification problems, test samples are classified based on the minimum distances to the returned hyperplanes, i.e., $g(\mathbf{x}_{test}) = \arg\min_{i=1,\ldots,K} (|\langle \mathbf{w}_i, \mathbf{x}_{test} \rangle + b_i|)$, where K is the number of classes in the training set.

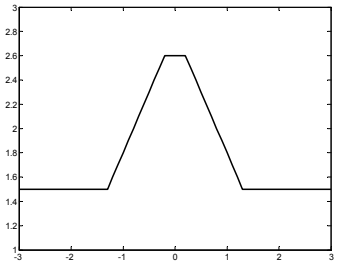

Fig. 5. The Symmetric Ramp Loss function, $J_{neg}(t) = R_{neg}(t) + R_{neg}(-t)$, for negative samples. Parameters are set to $s = -0.20$ and $\Delta = 0.30$.

III. EXPERIMENTS

We tested the proposed method 2S-BFHC on synthetic and real databases to assess its performance and we compared our results to those obtained by the related classification methods including Generalized Eigenvalue Proximal Support Vector Machine (GEPSVM), Twin Support Vector Machines

(TSVMs) and SVM. One-Against-Rest (OAR) regime was used for multi-class classification problems. For the nonlinear (kernelized) classifiers, we only used the Gaussian kernels.

A. Experiments on Synthetic Data

Here we consider an object detection scenario where the positive class samples are surrounded by the negative samples. To this end, we created two-dimensional normally distributed data plotted in Fig. 6. The positive class samples have a mean of $\mu = [0 \quad 0]$ and x and y dimensions are uncorrelated with the corresponding standard deviations of 0.1 and 0.9, respectively. Positive class samples are surrounded with negative samples having the same covariance structure but different means of $\mu = [-1.5 \quad 0]$ and $\mu = [1.5 \quad 0]$. Thus, the optimal best fitting hyperplane coincides with the y-axis. Consequently, the normal of the optimal hyperplane must be in the direction of x-axis.

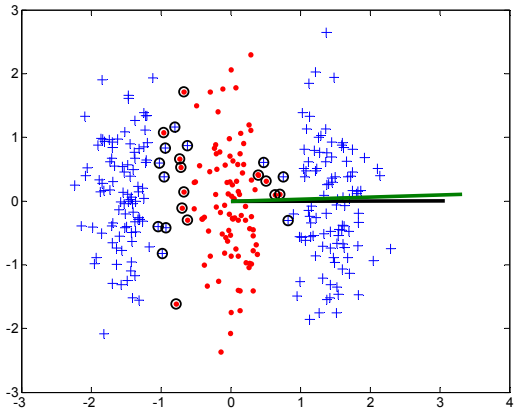

Fig. 6. Normal vectors of the fitting hyperplanes returned by the proposed method and GEPSVM. The proposed method successfully returns the correct support vectors in the vicinity of the positive class boundary as shown with the circles around the samples

Among all the best fitting hyperplane based methods, only GEPSVM and our proposed 2S-BFHC methods allow negative samples to lie on both sides of the separating hyperplane. As a result, all linear methods except GEPSVM and 2S-BFHC will fail for this problem. Thus, we compare only these two methods. We initialized the normal of the best fitting separating hyperplane with $\mathbf{w}0_{+} = [1 \quad 1]$, and the proposed method converged to the solution in 3 iterations and returned the solution of $\mathbf{w}_{+} = [1.397 \quad 0.004]$ and $b = 0.05$ which is very close to the optimal solution. The normal of the separating hyperlane returned by the proposed method is shown with black solid line in Fig. 6. Our proposed method also successfully returns the correct support vectors in the vicinity of the positive class boundary as shown with the circles around the samples in the figure. GEPSVM on the other hand returns the normal $\mathbf{w} = [0.99 \quad 0.03]$ shown with the green solid line in Fig. 6. For quantitative comparison, we create same amount of test data with the same distributions and compute the Average Precision (AP) scores obtained from precision-recall curves. The proposed method achieves the highest accuracy 0.923 as expected, and GEPSVM achieves 0.918.

B. Experiments on the UCI Repository and VOC datasets

We tested the proposed method on the Volatile Organic Compound (VOC) identification (available at *http://users.rowan.edu/polikar/RESEARCH/vocdb.html*) and 6 lower-dimensional datasets from the UCI repository: Ionosphere, Iris, Image Segmentation (IS), Pima Indian Diabetes (PID), Wine, and Wisconsin Diagnostic Breast Cancer (WDBC). The sizes and dimensionalities of the datasets are given in Table I. Here we tested both linear and kernel methods. We initialized the linear 2S-BFHC with the hyperplane returned by the GEPSVM, and we used the hyperplanes returned by linear 2S-BFHC method to initialize the kernel 2S-BFHC.

TABLE I. VOC AND LOW-DIMENSIONAL DATASETS FROM THE UCI REPOSITORY

Datasets	# Classes	# Examples	Dimension
Ionosphere	2	351	34
Iris	3	150	4
IS	7	2310	19
PID	2	768	8
VOC	4	384	6
Wine	3	178	13
WDBC	2	569	30

We used 5-fold cross-validation over random partition of the samples of each class to evaluate the performance, averaging over all five choices of 4 fold for training and the remaining 1 for testing. The results are given in Table III. Although being quite mixed, results indicate that the generalization performance of the proposed method compares favorably with SVM and other related hyper-plane fitting algorithms. More precisely, the proposed 2S-BFHC classifier achieves the 4 best results out of 7 for both linear and nonlinear cases. SVM also performs well yielding the 3 best results in nonlinear case and 2 best results in linear case. GEPSVM generally produces the worst classification accuracies. It was significantly outperformed by the best performing method on 5 datasets in linear case, and on 3 datasets in nonlinear case. Using kernels typically increases the classification accuracy with the exception on the Iris and Wine databases, where the accuracies of kernel methods are slightly behind the accuracies of linear methods. It should be noted that only SVM and the proposed method return sparse solutions in nonlinear case.

IV. SUMMARY AND CONCLUSION

Classifiers using the best fitting hyperplanes are becoming increasingly popular owing to the fact that they are better suited for open set recognition and object detection problems. Especially, in case of open set recognition problems, one can easily reject the test samples if the distances from those samples to the fitting hyperplanes are larger than empirically determined thresholds. However, existing hyperplane fitting algorithms have two major limitations: They are not suitable

TABLE II. CLASSIFICATION RATES (%) ON VOC AND UCI DATASETS

Linear Methods	Ionosphere	Iris	IS	PID	VOC	Wine	WDBC
GEPSVM	74.91, $\sigma = 5.8$	97.33, $\sigma = 2.7$	68.23, $\sigma = 5.5$	75.00, $\sigma = 3.1$	64.07, $\sigma = 5.4$	81.48, $\sigma = 14.1$	88.23, $\sigma = 2.1$
TSVM	**88.86**, $\sigma = 6.3$	96.67, $\sigma = 3.3$	91.94, $\sigma = 1.6$	77.21, $\sigma = 1.8$	79.57, $\sigma = 4.3$	97.65, $\sigma = 3.8$	96.48, $\sigma = 0.9$
SVM	86.59, $\sigma = 5.1$	93.33, $\sigma = 5.7$	**92.55**, $\sigma = 1.2$	**77.22**, $\sigma = 2.1$	75.76, $\sigma = 4.7$	97.17, $\sigma = 2.0$	96.66, $\sigma = 1.4$
2S-BFHC	85.46, $\sigma = 3.4$	**98.00,** $\sigma = 2.9$	90.00, $\sigma = 2.3$	75.91, $\sigma = 2.5$	**79.73**, $\sigma = 7.2$	**98.35**, $\sigma = 1.5$	**97.54**, $\sigma = 0.7$
Kernel Methods							
GEPSVM	88.87, $\sigma = 4.4$	95.33, $\sigma = 3.8$	87.83, $\sigma = 2.0$	75.92, $\sigma = 4.0$	82.71, $\sigma = 4.9$	97.06, $\sigma = 3.6$	95.07, $\sigma = 1.0$
TSVM	93.72, $\sigma = 4.5$	94.00, $\sigma = 2.7$	92.90, $\sigma = 1.9$	**77.74**, $\sigma = 2.3$	91.74, $\sigma = 3.2$	**97.76**, $\sigma = 1.2$	97.53, $\sigma = 0.9$
SVM	**94.30**, $\sigma = 3.1$	96.67, $\sigma = 4.0$	**97.14**, $\sigma = 0.4$	77.61, $\sigma = 1.8$	93.10, $\sigma = 3.6$	**97.76**, $\sigma = 1.2$	97.54, $\sigma = 0.7$
2S-BFHC	93.73, $\sigma = 3.4$	**97.33**, $\sigma = 2.7$	96.71, $\sigma = 0.6$	76.42, $\sigma = 2.0$	**93.40**, $\sigma = 4.2$	**97.76**, $\sigma = 1.2$	**97.89**, $\sigma = 1.8$

for large-scale classification problems since one has to make eigen-decomposition on the resulting large matrices or inverses of those large matrices must be taken. Existing classifiers are also not efficient in terms of real-time performance because the classifiers do not return sparse solutions. In this paper, we have proposed a novel hyperplane fitting classifier that does not have the limitations mentioned above. More precisely, the proposed classifier uses SMO algorithm, which does not require constructing large Hessian matrices, making the method suitable for large-scale problems. Moreover, the returned solutions are sparse similar to SVMs, thus the proposed method is very efficient in terms of testing time.

We tested classification accuracies of the proposed method on both synthetic and real-world classification problems. The proposed method typically outperformed other best-fitting hyperplane classifiers in most of the cases, and it produced comparable results to the SVM classifier.

ACKNOWLEDGMENT

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant number EEEAG-109E279.

REFERENCES

[1] C. Cortes, V. Vapnik, "Support vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.

[2] J. Platt, "Fast training of support vector machines using sequential minimal optimization," Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp. 185-208, 1999.

[3] R.-E. Fan, P.-H. Chen, C.-J. Lin, "Working set selection using second order information for training SVM," Journal of Machine Learning Research, vol. 6, pp.1889-1918, 2005.

[4] T. Joachims, "Making large-scale support vector machine learning practical," Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, pp. 185-208, 1999.

[5] O. L. Mangasarian, E. W. Wild, "Multisurface proximal support vector machine classification via generalized eigenvalues," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, pp. 69-74, 2006.

[6] G. Fung, O. L. Mangasarian, "Proximal support vector machine classifiers," Proceedings of Knowledge Discovery and Data Mining, 2001.

[7] Jayadeva, R. Khemchandani, S. Chandra, "Twin support vector machines for pattern classification," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, pp. 905-910, 2007.

[8] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, N.-Y. Deng, "Improvements on twin support vector machines," IEEE Transactions on Neural Networks, vol. 22, pp. 962-968, 2011.

[9] M. A. Kumar, M. Gopal, "Least squares twin support vector machines for pattern classification," Expert Systems with Applications, vol. 36, pp. 7535-7543, 2009.

[10] M. A. Kumar, M. Gopal, "Application of smoothing technique on twin support vector machines," Pattern Recognition Letters, vol. 29, pp. 1842-1848, 2008.

[11] S. Gao, Q. Ye, N. Ye, "1-Norm least squares twin support vector machines," Neurocomputing, vol. 74, pp. 3590-3597, 2011.

[12] X. Peng, "TPMSVM: A novel twin parametric-margin support vector machine for pattern recognition," Pattern Recognition, vol. 44, pp. 2678-2692, 2011.

[13] W. J. Scherier, A. Rocha, A. Sapkota, T. E. Boult, "Towards open set recognition," IEEE Transactions on PAMI, vol. 35, pp. 1757 – 1772, 2013.

[14] H. Cevikalp, B. Triggs, H. S. Yavuz, Y. Kucuk, M. Kucuk, A. Barkana, "Large margin classifiers based on affine hulls," Neurocomputing, vol. 73, pp. 3160-3168, 2010.

[15] H. Cevikalp, B. Triggs, "Efficient object detection using cascades of nearest convex model classifiers," IEEE Society Conference on Computer Vision and Pattern Recognition, 2012.

[16] X. Peng, "Building sparse twin support vector machine classifiers in primal space," Information Sciences, vol. 181, pp. 3967-3980, 2011.

[17] A. L. Yuille, A. Rangarajan, "The concave-convex procedure (CCCP)," Advances in Neural Information Processing Systems, 2002.

[18] R. Collobert, F. Sinz, J. Weston, and L. Bottou, "Large scale transductive SVMs," Journal of Machine Learning Research, vol. 7, pp. 1687-1712, 2006.