Best Fitting Hyperplanes for Classification

Hakan Cevikalp, *Member, IEEE*

Abstract—In this paper, we propose novel methods that are more suitable than classical large-margin classifiers for open set recognition and object detection tasks. The proposed methods use 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 classes and is as far as possible from the other class samples. To this end, we propose two different classifiers: The first classifier solves a convex quadratic optimization problem, but negative samples can lie on one side of the best fitting hyperplane. The second classifier, however, allows the negative samples to lie on both sides of the fitting hyperplane by using concave-convex procedure. Both methods are extended to the nonlinear case by using the kernel trick. In contrast to the existing hyperplane fitting classifiers in the literature, our proposed methods are suitable for large-scale problems, and they return sparse solutions. The experiments on several databases show that the proposed methods typically outperform other hyperplane fitting classifiers, and they work as good as the SVM classifier in classical recognition tasks. However, the proposed methods significantly outperform SVM in open set recognition and object detection tasks.

Index Terms—best fitting hyperlane classifier, open set recognition, large margin classifier, kernel methods, support vector machines.

1 Introduction

Large margin classifiers have been successfully used in many fields including computer vision, text analysis, biometrics and bioinformatics. The prototypical method of this type, 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 [2,3,4,5]. The solution is sparse in the sense that, once the closest points (so-called support vectors) have been found, it depends only on them. Because SVM classifiers try to ensure the widest possible margin for error, the resulting classifiers have very good practical performance. However, SVMs are not perfect. Roughly speaking, the setting for which they were designed is one where the classes can be modeled as non-overlapping convex clouds in a feature space, which can thus be separated using affine hyperplanes. Approximating each class with a convex hull may be problematic in that it can seriously underestimate the true extent of the classes involved. Therefore, new maximum margin classifiers, which approximate classes with other convex class models, have been introduced. The maximum margin affine hull case is studied in [6]. This method is a special case of the Least Squares [7] and Proximal [8] SVMs, in which disjoint affine hulls of two classes lie on two parallel hyperplanes generated such that each hyperplane is closest to the affine hull of the corresponding class and two hyperplanes are as far apart as possible. The Minimax Probability Machine [9] and its variants [10] are alternative large margin classifiers that use hyper-ellipsoids to approximate classes. Recently, we proposed a new large-margin classifier [11] that approximates the classes with hyper-disks.

Mangasarian and Wild [12] proposed the generalized eigenvalue proximal support vector machine (GEPSVM) classifier. GEPSVM finds two non-parallel hyperplanes (as opposed to the parallel hyperplanes returned by the Proximal SVMs and affine hull based large margin classifier) by solving two generalized eigenvalue problems such 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, [13] 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 samples of one of the two classes and is as far as possible from the samples of other class. 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 the 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. [14] further improved the TSVM classifier by introducing a regularization term on the hyperplane parameters. Kumar and Gopal proposed the least squares version of TSVM [15] and smooth TSVM [16]. Some other extensions of TSVM can also be found in [17,18].

Although classifiers based on the best-fitting hyperplanes

• *H. Cevikalp is with the Electrical and Electronics Engineering Department, Machine Learning and Computer Vision Laboratory, Eskisehir Osmangazi University Eskisehir, Turkey.*
E-mail: hakan.cevikalp@gmail.com

have been proposed as alternatives to the binary SVM classifier, there are better application areas of these methods in recognition problems known as "open set recognition" [19]. In classical classification problems, it is assumed that all testing classes are known at training time. However, in more realistic applications, samples may come from unknown classes during testing time. Margin-based classifiers, such as SVM or affine hull based classifiers seek to maximize the distance between the known class samples and the decision boundary. Regions far from the known data (called open space in [19]) 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 types of applications. They are also more suitable than large margin classifiers for visual object detection tasks, in which there are a limited number of object class samples, whereas there are millions of negative samples coming from thousands of different classes [20,21].

The classifiers using the best fitting hyperplane models have wider application areas compared to the large-margin classifiers. However, 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 the inverse of a $(n+1)\times(n+1)$ matrix. For small sized classification problems, it is reported that the training times of GEPSVM and TSVM take less time compared to the training time of the SVM classifier. However, for large-scale problems, it is difficult (most of the time, impossible) to fit those large matrices into memory and operate on them. The second limitation is related to the sparsity of the solution. As opposed to the SVM classifier, the solution returned by GEPSVM or TSVM is not sparse, i.e., all training samples become support vectors. Therefore, the testing times of those classifiers are much slower compared to the testing time of SVM.

In this paper, we introduce novel classifiers that use the best fitting hyperplane approach. Our methods do not have the limitations of both GEPSVM and TSVMs. In particular, the proposed methods are suitable for large-scale classification problems, and they always return sparse solutions. They are also better suited for open set recognition and visual object detection problems as demonstrated in the experiments. The rest of the paper is organized as follows. In Section 2, we give related methods and discuss their limitations. Section 3 introduces the proposed methods. Section 4 presents our experimental results, and Section 5 concludes the paper.

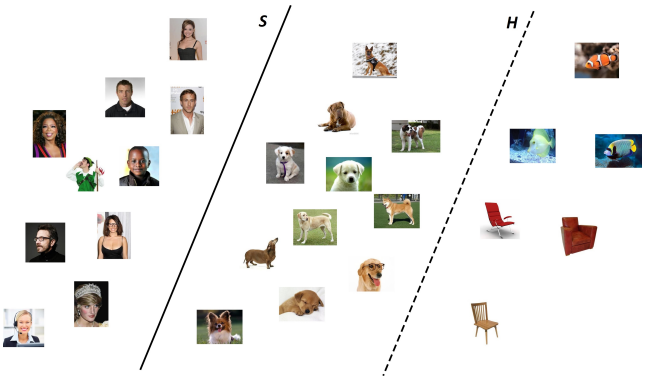


Fig. 1. The separating hyperplane ***S*** returned by the SVM classifier separates people and dog classes. All samples under the separating hyperplane are assigned to the dog class. When there are test samples coming from the unknown classes, such as chair and fish, these samples will be erroneously 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.

2 Related Work

Here, we briefly describe two related best fitting hyperplane classification methods, and discuss their limitations.

2.1 Generalized Eigenvalue Proximal Support Vector Machine (GEPSVM) Classifier

This method searches for two nonparallel 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. Let data samples belonging to the positive (+1) and negative (-1) classes be represented by matrices $\mathbf{A}\in\mathbb{R}^{n_+\times d}$ and $\mathbf{B}\in\mathbb{R}^{n_-\times d}$, respectively. Here, $n_+(n_-)$ denotes the number of positive (negative) samples, and d is the dimensionality of the input space. The GEPSVM classifier [12] returns two hyperplanes in $\mathbb{R}^d$

$$\mathbf{w}_+^\top\mathbf{x}+b_+=0,\ \ \mathbf{w}_-^\top\mathbf{x}+b_-=0,$$

where the first hyperplane is closest to the samples in the positive class and furthest from the samples in the negative class, while the second hyperplane is closest to the samples in the negative class and furthest from the samples in the positive class. To compute the first hyperplane, characterized by $(\mathbf{w}_+,b_+)$, the method minimizes the sum of the squares of L_2 (Euclidean) distances between each of the samples of the positive class to the hyperplane divided by the squares of L_2 distances between each of the samples of the negative class to the hyperplane. This leads to the following optimization problem

$$\underset{\mathbf{w}_+,b_+,\|\mathbf{w}_+\|=1}{\arg\min}\ \frac{\|\mathbf{A}\mathbf{w}_+ + \mathbf{e}_+b_+\|^2+\delta(\|\mathbf{w}_+\|^2+b_+^2)}{\|\mathbf{B}\mathbf{w}_+ + \mathbf{e}_-b_+\|^2}, \quad (1)$$

where δ is a user-set regularization constant, and $\mathbf{e}_+(\mathbf{e}_-)$ is a column vector of ones with appropriate dimension for the positive (negative) class. Let $\mathbf{G}=[\mathbf{A}\ \ \mathbf{e}_+]^\top[\mathbf{A}\ \ \mathbf{e}_+]+\delta\mathbf{I}$,

$\mathbf{H} = [\mathbf{B} \quad \mathbf{e}_-]^\top[\mathbf{B} \quad \mathbf{e}_-]$, and $\mathbf{z} = \begin{pmatrix}\mathbf{w}_+\\ b_+\end{pmatrix}$. In this case, optimization problem (1) can be re-expressed as

$$\arg\min_{\mathbf{z}} \ \frac{\mathbf{z}^\top \mathbf{G}\,\mathbf{z}}{\mathbf{z}^\top \mathbf{H}\,\mathbf{z}}. \tag{2}$$

The solution of this problem reduces to finding the eigenvector corresponding to the smallest eigenvalue of the generalized eigenvalue problem, $\mathbf{G}\,\mathbf{z} = \lambda\,\mathbf{H}\,\mathbf{z}$, and renormalizing it to find $(\mathbf{w}_+, b_+)$. To find the second hyperplane, $(\mathbf{w}_-, b_-)$, the roles of $\mathbf{A}$ and $\mathbf{B}$ in the optimization problem (1) are interchanged, and all steps are repeated. Once the hyperplanes are found, test samples are classified based on the minimum distances to the returned hyperplanes. The details of the kernelization procedure can be found in [12].

In the nonlinear case, the method requires eigendecomposition on $(n+1)\times(n+1)$ matrices, where $n = n_+ + n_-$ is the total number of samples. Thus, the method is not suitable for large-scale problems. The method also does not return sparse solutions, and kernel evaluations against all samples in the training set are required during the testing of a single test example. This reduces the real-time efficiency of the method. Lastly, since GEPSVM uses the squares of L_2 distances, its generalization accuracy significantly drops when there are outliers in the training set, as we observed in [20].

2.2 Twin Support Vector Machine (TSVM) Classifier

Similar to GEPSVM, the TSVM classifier [13] also searches for two non-parallel hyperplanes such that each hyperplane is closer to the samples of one of the two classes and as far as possible from the other class samples. However, fitting hyperplanes are obtained by solving the following pair of convex quadratic programming problems

$$\begin{aligned}\arg\min_{\mathbf{w}_+,b_+,\boldsymbol{\xi}} \ & \frac{1}{2}(\mathbf{A}\mathbf{w}_+ + \mathbf{e}_+ b_+)^\top(\mathbf{A}\mathbf{w}_+ + \mathbf{e}_+ b_+) + C_-\mathbf{e}_-^\top\boldsymbol{\xi}\\ \text{s.t.} \ & -(\mathbf{B}\mathbf{w}_+ + \mathbf{e}_- b_+) + \boldsymbol{\xi} \geq \mathbf{e}_-, \ \ \boldsymbol{\xi}\geq \mathbf{0},\end{aligned} \tag{3}$$

and

$$\begin{aligned}\arg\min_{\mathbf{w}_-,b_-,\boldsymbol{\xi}} \ & \frac{1}{2}(\mathbf{B}\mathbf{w}_- + \mathbf{e}_- b_-)^\top(\mathbf{B}\mathbf{w}_- + \mathbf{e}_- b_-) + C_+\mathbf{e}_+^\top\boldsymbol{\xi}\\ \text{s.t.} \ & (\mathbf{A}\mathbf{w}_- + \mathbf{e}_+ b_-) + \boldsymbol{\xi} \geq \mathbf{e}_+, \ \ \boldsymbol{\xi}\geq \mathbf{0},\end{aligned} \tag{4}$$

where C_+ and C_- are user defined parameters that control the weight of the errors associated to the samples' violated constraints. Let $\mathbf{H} = [\mathbf{A} \quad \mathbf{e}_+]$, $\mathbf{G} = [\mathbf{B} \quad \mathbf{e}_-]$, and $\mathbf{u} = \begin{pmatrix}\mathbf{w}_+\\ b_+\end{pmatrix}$. Then, the dual of the first optimization problem becomes

$$\begin{aligned}\arg\min_{\boldsymbol{\alpha}} & \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{G}(\mathbf{H}^\top\mathbf{H})^{-1}\mathbf{G}^\top\boldsymbol{\alpha} - \mathbf{e}_-^\top\boldsymbol{\alpha}\\ \text{s.t.} \ & 0\leq\alpha_i\leq C, \ i=1,\ldots,n.\end{aligned} \tag{5}$$

Once the solution of this quadratic programming problem is found, the first fitting hyperplane is obtained by using

$$\mathbf{u} = -(\mathbf{H}^\top\mathbf{H} + \delta\mathbf{I})^{-1}\mathbf{G}\boldsymbol{\alpha}. \tag{6}$$

The second fitting hyperplane is also obtained by the same procedure by interchanging the matrices $\mathbf{A}$ and $\mathbf{B}$. Test samples are classified based on the minimum distances to the returned hyperplanes. The details of the kernelization procedure can be found in [13].

As seen in the optimization problems in (3) and (4), the TSVM classifier does not allow negative samples to lie on both sides of the fitting hyperplanes. Instead, the negative samples are forced to lie to the left of the fitting hyperplane (below the hyperplane in (3)), whereas the positive samples are forced to lie to the right of the hyperplane (above the hyperplane in (4)), separated from the fitting hyperplane by a distance of at least $1/||\mathbf{w}_{+(-)}||$. Allowing further class samples to lie only on one side of the hyperplane is a limitation because there are many classification problems in which one class's samples are surrounded by a diffuse sea of other classes' samples. In addition, the nonlinear TSVM classifier requires taking the inverses of $(n+1)\times(n+1)$ matrices, where $n = n_+ + n_-$ is the total number of samples. Thus, the method is not suitable for large-scale problems as in the GEPSVM classifier. The method also does not return sparse solutions.

3 METHOD

We propose two classification methods that use the best fitting hyperplanes: the One-Sided Best Fitting Hyperplane Classifier (1S-BFHC) and the Two-Sided Best Fitting Hyperplane Classifier (2S-BFHC). The first method allows the negative samples to lie only on one side of the fitting hyperplane, and it requires solving a convex quadratic optimization problem. The second method allows the negative samples to lie on both sides of the hyperplane, and it requires solving a more complicated non-convex optimization problem.

3.1 One-Sided Best Fitting Hyperplane Classifier (1S-BFHC)

Similar to the hyperplane fitting classifiers in the literature, our first proposed method also searches for the best fitting hyperplanes such that each hyperplane is closer to the samples of one of the classes and far from the other classes' samples. We first proposed this method for face detection and focused on a single best fitting separating hyperplane [20]. Let $\mathbf{x}$ be the sample's feature vector and let $\mathbf{w}_+^\top + b_+ = 0$ be the equation of the best fitting hyperplane for the positive class samples. In this case, finding the best fitting discriminative hyperplane problem can be formulated as the following convex quadratic programming problem

$$\begin{aligned}\arg\min_{\mathbf{w}_+,b_+,\boldsymbol{\xi}\geq 0} \ & \frac{1}{2}||\mathbf{w}_+||^2 + C_+\sum_i(\xi_i+\xi_i^*) + C_-\sum_j\xi_j\\ \text{s.t.} \ & \mathbf{w}_+^\top\mathbf{x}_i + b_+ \leq \Delta + \xi_i,\\ & \mathbf{w}_+^\top\mathbf{x}_i + b_+ \geq -\Delta - \xi_i^*,\\ & \mathbf{w}_+^\top\mathbf{x}_j + b_+ \geq \Delta + 1 - \xi_j, \ \ i\in I_+, \ \ j\in I_-,\end{aligned} \tag{7}$$

where $C_+(C_-)$ is a user defined parameter that controls the weight of the errors associated with the positive (negative)

samples, $I_+(I_-)$ is the set including indices of the positive (negative) samples, and Δ is a constant that can be set to a value larger than zero (we typically set it to a value between 0 and 1). With this formulation, we are constraining the positive class samples to lie between two parallel hyperplanes $\mathbf{w}_+^\top\mathbf{x}+b_+ = \Delta$ and $\mathbf{w}_+^\top\mathbf{x}+b_+ = -\Delta$. The negative samples lie to the right of the hyperplane $\mathbf{w}_+^\top\mathbf{x} + b_+ = 1 + \Delta$, separated from the positive samples by a margin of at least $1/||\mathbf{w}_+||$. Therefore, our method also allows the negative class samples to lie only on one side of the hyperplane as in TSVM, which is a limitation as described before. Positive slack variables, ξ_i, ξ_i^*,ξ_j, are introduced for the samples violating the constraints.

To derive the dual, we consider the Lagrangian

$$L(\mathbf{w}_+, b_+, \boldsymbol{\xi}, \boldsymbol{\alpha}_+, \boldsymbol{\alpha}_+^*, \boldsymbol{\alpha}_-, \boldsymbol{\kappa}_+, \boldsymbol{\kappa}_+^*, \boldsymbol{\kappa}_-) = \frac{1}{2}||\mathbf{w}_+||^2 + C_+ \sum_i(\xi_i + \xi_i^*) + C_- \sum_j \xi_j + \sum_i \alpha_{+i}(\mathbf{w}_+^\top\mathbf{x}_i + b_+ - \Delta - \xi_i) - \sum_i \alpha_{+i}^*(\mathbf{w}_+^\top\mathbf{x}_i + b_+ + \Delta + \xi_i^*) - \sum_j \alpha_{-j}(\mathbf{w}_+^\top\mathbf{x}_i + b_+ - \Delta - 1 + \xi_j) - \sum_i \kappa_{+i}\xi_i - \sum_i \kappa_{+i}^*\xi_i^* - \sum_j \kappa_{-j}\xi_j,$$

where $\boldsymbol{\alpha}_+$, $\boldsymbol{\alpha}_+^*$, $\boldsymbol{\alpha}_-$, $\boldsymbol{\kappa}_+$, $\boldsymbol{\kappa}_+^*$, $\boldsymbol{\kappa}_- \geq \mathbf{0}$. The Lagrangian, L, has to be maximized with respect to $\boldsymbol{\alpha}_+$, $\boldsymbol{\alpha}_+^*$, $\boldsymbol{\alpha}_-$, $\boldsymbol{\kappa}_+$, $\boldsymbol{\kappa}_+^*$, $\boldsymbol{\kappa}_-$, and minimized with respect to $\mathbf{w}_+$, b_+, $\boldsymbol{\xi}$. The optimality conditions yield

$$\frac{\partial L}{\partial \mathbf{w}_+} = \mathbf{0} \rightarrow \mathbf{w}_+ = \sum_i(\alpha_{+i}^* - \alpha_{+i})\mathbf{x}_i + \sum_j \alpha_{-j}\mathbf{x}_j, \tag{8}$$

$$\frac{\partial L}{\partial b_+} = 0 \rightarrow \sum_i(\alpha_{+i}^* - \alpha_{+i}) + \sum_j \alpha_{-j} = 0,$$
$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_{+i} \leq C_+,\ i \in I_+,$$
$$\frac{\partial L}{\partial \xi_i^*} = 0 \rightarrow 0 \leq \alpha_{+i}^* \leq C_+,\ i \in I_+,$$
$$\frac{\partial L}{\partial \xi_j} = 0 \rightarrow 0 \leq \alpha_{-j} \leq C_-,\ j \in I_-.$$

Let $\mathbf{X} = [\mathbf{X}_+ \quad -\mathbf{X}_+ \quad \mathbf{X}_-]$ be the matrix of training samples, where $\mathbf{X}_+$ is the matrix whose columns are the positive samples and $\mathbf{X}_-$ is the matrix with negative samples. By defining $\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_+^* \\ \boldsymbol{\alpha}_+ \\ \boldsymbol{\alpha}_- \end{pmatrix}$, the dual becomes

$$\begin{aligned} \arg\min_{\boldsymbol{\alpha}} \ & \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{H}\boldsymbol{\alpha} + \mathbf{c}^\top\boldsymbol{\alpha} \\ \text{s.t. } & \sum_i(\alpha_{+i}^* - \alpha_{+i}) + \sum_j \alpha_{-j} = 0, \\ & 0 \leq \alpha_{+i}, \alpha_{+i}^* \leq C_+,\ 0 \leq \alpha_{-j} \leq C_-,\ i \in I_+,\ j \in I_- \end{aligned} \tag{9}$$

where $\mathbf{H} = \mathbf{X}^\top\mathbf{X}$ and $\mathbf{c} = \begin{pmatrix} \Delta\mathbf{e}_+ \\ \Delta\mathbf{e}_+ \\ -(\Delta+1)\mathbf{e}_- \end{pmatrix}$. This is a convex-quadratic programming problem that can be solved by any quadratic program solver. Once $\boldsymbol{\alpha}$ is found, the normal of the hyperplane can be found by using (8), and the offset b_+ of the hyperplane can be computed by using the constraints given in the primal problem (7) for the samples with Lagrange coefficients $0 < \alpha_{+i}, \alpha_{+i}^* < C_+$, $0 < \alpha_{-j} < C_-$. This formulation resulted in a more robust discriminating hyperplane compared to the least squares based GEPSVM in our experiments in [20], most likely due to the high-dimensionality of the feature space.

The second fitting hyperplane is also obtained by the same procedure by interchanging the roles of positive and negative samples. Once the best fitting hyperplanes are found, test samples are classified based on the minimum distances to the returned hyperplanes. For multi-class classification problems with K classes, we find a fitting hyperplane $(\mathbf{w}_i, b_i)$, $i = 1, \ldots, K$, for each class in the training set, and a new test sample is classified by using the decision function

$$g(\mathbf{x}_{test}) = \arg\min_{i=1,\ldots,K}(\left|\mathbf{w}_i^\top\mathbf{x}_{test} + b_i\right| / ||\mathbf{w}_i||). \tag{10}$$

The proposed method can be extended to the nonlinear case by using the kernel trick. Notice that the objective function of the convex QP problem in (9) can be written in terms of the dot products of the sample pairs. Thus, we replace all $\langle\mathbf{x}_i, \mathbf{x}_j\rangle = \mathbf{x}_i^\top\mathbf{x}_j$ with the kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$ where $\phi : \mathbb{R}^d \rightarrow \Im$ is the mapping function from the input space to the feature space $\Im$.

There are several advantages of our proposed method compared to the GEPSVM and TSVM classifiers. First of all, the quadratic optimization problem given in (9) is very similar to the SVM formulation, and it can be efficiently solved by using Sequential Minimal Optimization [2]. 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. Therefore, our proposed method is suitable for large-scale applications. Second, the solution returned by the optimization problem in (9) is sparse, i.e., most of the α_i coefficients are zero.

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

This method also searches for the best fitting hyperplanes such that each hyperplane is closer to the samples of one of the classes and far from the other samples. As opposed to the first proposed method, this 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 illustrated in Fig. 2, where Δ is a user defined constant that can be set between 0 and 1. 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 [22].

Let us define the function we will use for label assignment as being of the form

$$f_\theta(\mathbf{x}) = \mathbf{w}_+^\top\mathbf{x} + b_+, \tag{11}$$

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, $\mathbf{w}_+^\top \mathbf{x} + b_+ = 1 - \Delta$ and $\mathbf{w}_+^\top \mathbf{x} + b_+ = -1 + \Delta$. To implement this goal, we use the symmetric Ramp Loss cost function (shown in Fig. 3) defined as

$$J_{pos}(t) = R_{pos}(t) + R_{pos}(-t), \tag{12}$$

where $R_{pos}(t) = \min(1 - s, \max(0, -1 + \Delta - t))$ is the so-called "Ramp Loss" function [23] illustrated in Fig. 4, where $-1 < s \leq 0$ 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 convex 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.

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 the negative samples become $|\mathbf{w}_+^\top + b_+| > 1 + \Delta$. To implement this, we use another Symmetric Ramp loss (shown in Fig. 5) defined as

$$J_{neg}(t) = R_{neg}(t) + R_{neg}(-t), \tag{13}$$

where $R_{neg}(t) = \min(1 + \Delta - s, \max(0, 1 + \Delta - t))$, illustrated in Fig. 4. Similar to the previous case, this Ramp Loss function can 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, the s parameter controls the wideness of the flat part of the symmetric

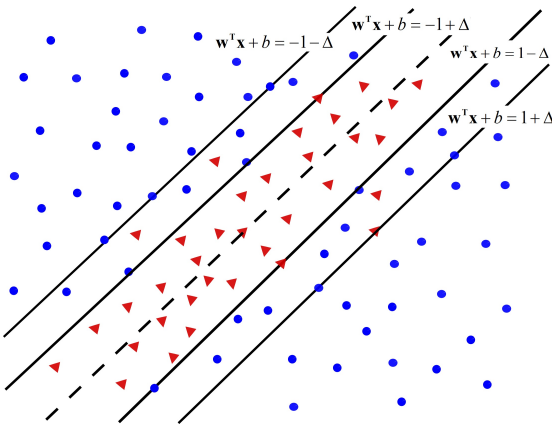


Fig. 2. In the proposed method, positive class samples (shown with red triangles) lie between two parallel hyperplanes characterized by $\mathbf{w}_+^\top \mathbf{x} + b_+ = 1 - \Delta$ and $\mathbf{w}_+^\top \mathbf{x} + b_+ = -1 + \Delta$. The negative samples shown with blue circles can lie on both sides of the fitting hyperplane (the fitting hyperplane, $\mathbf{w}_+^\top \mathbf{x} + b_+ = 0$, 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 the separable case.

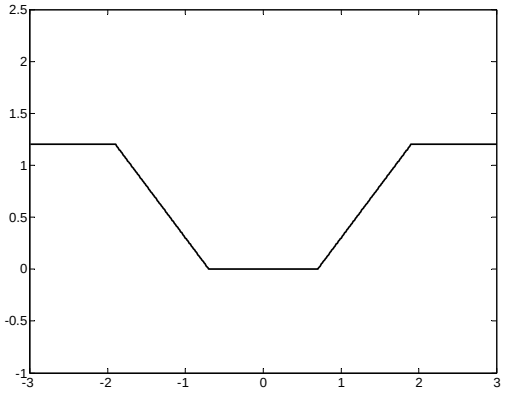


Fig. 3. Illustration of the cost function, $J_{pos}(t) = R_{pos}(t) + R_{pos}(-t)$, of the positive samples. $R_{pos}(t) = H_{-1+\Delta}(t) - H_{s-2+\Delta}(t)$ can be written as the sum of the convex Hinge loss and a concave loss function. Δ is set to 0.3 and $s = -0.20$.

component of the symmetric Ramp Loss plotted in Fig. 4.

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

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

In this case, our total cost function can be written as

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

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

$$J(\theta) = J_{convex}(\theta) + J_{concave}(\theta), \tag{14}$$

where

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

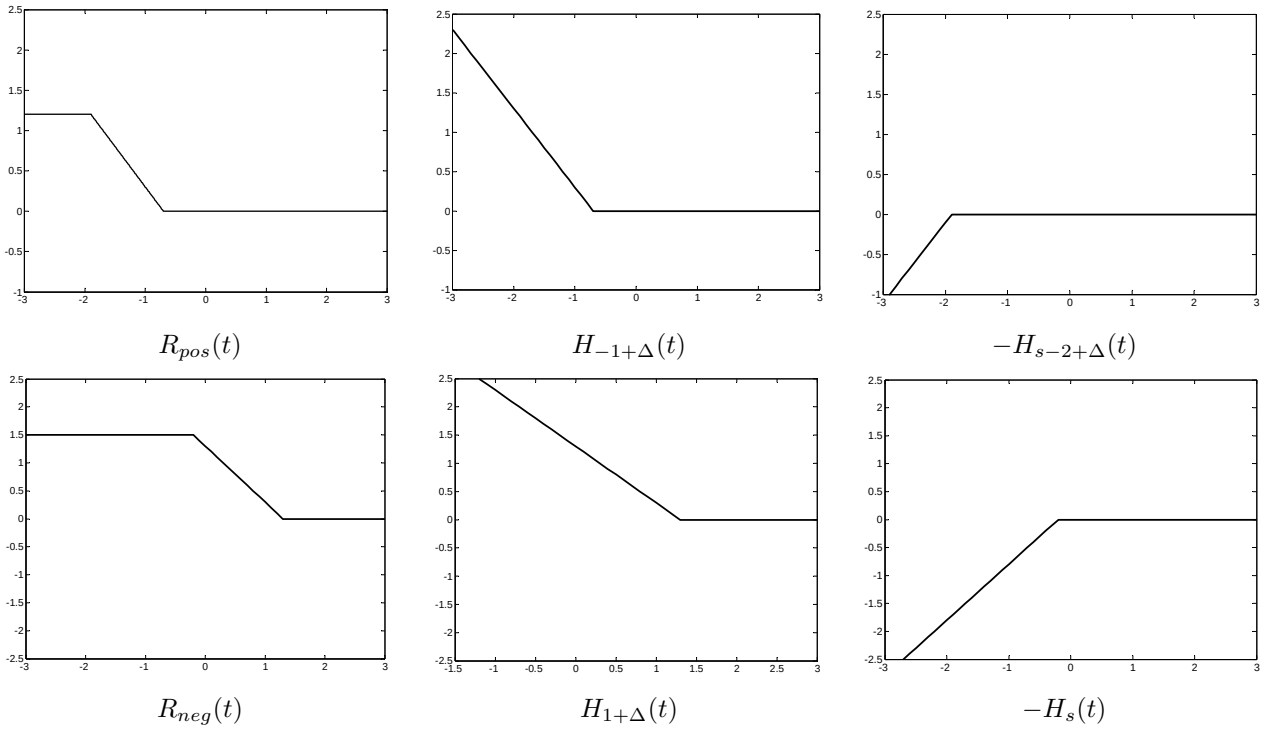


Fig. 4. Illustrations of the Ramp Loss functions, $R_{pos}(t) = \min(1-s, \max(0, -1+\Delta-t))$ (top left figure) and $R_{neg}(t) = \min(1+\Delta-s, \max(0, 1+\Delta-t))$ (bottom left). Each loss can be written as the sum of the convex Hinge loss (center) and the concave loss (right), i.e., $R_{pos}(t) = H_{-1+\Delta}(t) - H_{s-2+\Delta}(t)$ 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 $\Delta = 0.3$ and $s = -0.20$.

and

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

Because the cost function (14) can be decomposed into a convex and a concave part, we can apply 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 proven in [22]. 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
$\theta^{t+1} = \underset{\theta}{\operatorname{argmin}}(J_{convex}(\theta) + J'_{concave}(\theta^t)\theta)$
until convergence of θ^t

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

$$\beta_i = y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} = \begin{cases} C_+ & \text{if } y_i f_\theta(\mathbf{x}_i) < s-2+\Delta \text{ and } 1 \le i \le 2n_+ \\ C_- & \text{if } y_i f_\theta(\mathbf{x}_i) < s \text{ and } 2n_++1 \le i \le 2n_++2n_- \end{cases} \quad (15)$$

After some standard derivations shown in Appendix A (given as supplementary material), the method can be summarized as in Algorithm 2.

Note that the convex part of the algorithm becomes a

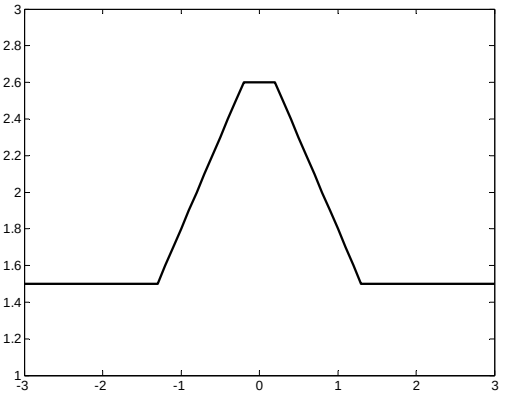


Fig. 5. The Symmetric Ramp Loss function, $J_{neg}(t) = R_{neg}(t) + R_{neg}(-t)$, for negative samples. The parameter s is set to -0.20.

Algorithm 2 Two Sided Best Fitting Hyperplane Classifier

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

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

repeat

- Solve the following convex QP problem

$$\begin{aligned} \arg\min_{\mathbf{h}} \quad & \sum_{i=1}^{2n_+ + 2n_-} \sum_{j=1}^{2n_+ + 2n_-} h_i h_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + (1-\Delta) \sum_{i=1}^{2n_+} y_i h_i - (1+\Delta) \sum_{i=2n_+ + 1}^{2n_+ + 2n_-} y_i h_i \\ \text{s.t.} \quad & \sum_{i=1}^{2n_+ + 2n_-} h_i = 0; \;\; -\beta_i^t \le y_i h_i \le C_+ - \beta_i^t, \;\; 1 \le i \le 2n_+; \\ & -\beta_i^t \le y_i h_i \le C_- - \beta_i^t, \;\; 2n_+ + 1 \le i \le 2n_+ + 2n_-. \end{aligned} \tag{16}$$

- Compute $\mathbf{w}_+^{t+1}$ and b_+^{t+1} as described in Appendix B.
- Compute

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

until $||\mathbf{w}_+^{t+1} - \mathbf{w}_+^t|| \le \epsilon_1$ or $||\boldsymbol{\beta}^{t+1} - \boldsymbol{\beta}^t|| \le \epsilon_2$, where ϵ_1 and ϵ_2 are some pre-defined small thresholds.

convex quadratic optimization with equality and inequality constraints, and it can be efficiently solved by using Sequential Minimal Optimization (SMO) for large-scale data. The method usually takes 3-5 iterations to converge to a solution based on the initialization of the algorithm. It should be noted that only the bounds on Lagrange coefficients 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., 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 as before. Once the best fitting hyperplanes are found, test samples are classified based on the minimum distances to the returned hyperplanes using (10). As in the previous 1S-BFHC method, this method can also be extended to the nonlinear case by using the kernel trick since the QP problem in Algorithm 2 can be written in terms of inner product of samples.

3.2.1 Initialization

Because the objective function is non-convex, the initialization of the best fitting hyperplane is very important (the returned solution will be a local minimum). For the synthetic and low-dimensional datasets used in our experiments, initialization with the hyperplane returned by GEPSVM or random initializations produced good results. However, as the dimensionality is increased, GEPSVM produced very poor results and thus initialization by using GEPSVM failed to produce good results. Therefore, we initialized the best fitting hyperplane by using the hyperplanes returned by the proposed 1S-BFHC or SVM. Since these methods only allow the negative samples to lie on one side of the hyperplane, the proposed 2S-BFHC classifier also returned the best fitting hyperplanes, where the negative samples typically fall on one side of the fitting hyperplane. Therefore, we searched for other hyperplanes for initialization. To this end, we proposed two alternatives. In the first approach, we approximated the positive samples by an affine hull (affine subspace) and used the so-called "common vector" [24] that gives the minimum distance from the origin to the affine hull, as illustrated in Fig. 6. This method works only for high-dimensional data samples where the dimensionality is higher than the total number of samples in a class. Note that when the data samples are projected onto the common vector, they all collapse to the same point. Thus, the hyperplane, using this vector as its normal, behaves like a perfect fitting hyperplane on which all positive class samples lie. The common vector of the positive class samples can be found by using

$$\mathbf{x}_{com} \;=\; \boldsymbol{\mu} - \mathbf{U}\mathbf{U}^\top \boldsymbol{\mu}, \tag{17}$$

where $\boldsymbol{\mu}$ is the mean of the samples in the positive class and $\mathbf{U}$ is the basis for the span of centered examples $\{\mathbf{x}_1 - \boldsymbol{\mu}, \mathbf{x}_2 - \boldsymbol{\mu}, \ldots, \mathbf{x}_{n_+} - \boldsymbol{\mu}\}$. $\mathbf{U}$ can be found by truncated SVD of centered samples or eigen-decomposition on the covariance matrix in the noisy case or Gram-Schmidt decomposition of centered samples in the noise-free case. If the dimensionality is not higher than the number of samples, $\mathbf{U}$ must be formed by using some of the most significant eigenvectors corresponding to the largest eigenvalues.

It should be noted that the common vector is obtained by using only the samples belonging to the positive class.

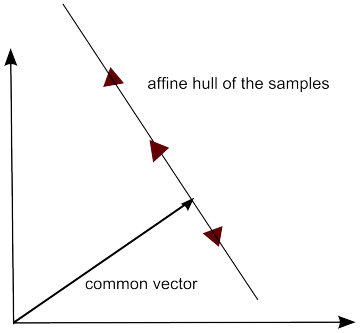


Fig. 6. The common vector determines the minimum distance from the origin to the affine hull of the samples. All points on an affine hull collapse to the same point when they are projected onto the common vector, and thus the common vector can be seen as the normal of an affine hull.

To obtain a common vector sensitive to the negative-class samples, one may use discriminative common vectors [25,26,27,28]. In this case, $\mathbf{U}$ is obtained from all class samples where each sample of a class is subtracted from its corresponding class center. More precisely, $\mathbf{U}$ is now the basis for the span of the samples

$$\left\{\mathbf{x}_1^1 - \boldsymbol{\mu}_1,\, \mathbf{x}_2^1 - \boldsymbol{\mu}_1, \ldots, \mathbf{x}_j^i - \boldsymbol{\mu}_i, \ldots, \mathbf{x}_{n_K}^K - \boldsymbol{\mu}_K\right\}, \tag{18}$$

where $\mathbf{x}_j^i$ represents the j-th sample of the i-th class, and $\boldsymbol{\mu}_i$ is the mean of all samples in the i-th class. The discriminative common vector is also found by using (17), and samples in each class collapse onto their corresponding point as before (in that way, there will be K different points for each class in the projected space). We obtained the best results by initializing the best fitting hyperplane with discriminative common vectors when the dimensionality is too high compared to the total number of samples in the training set.

3.2.2 Discussion on Cost Functions and Parameters

In the 2S-BFHC method, positive samples lie between two parallel hyperplanes, and the distance between these hyperplanes is given by $2(1-\Delta)/||\mathbf{w}_+||$. In a similar manner, the distance between positive and negative samples is at least $2\Delta/||\mathbf{w}_+||$. In the proposed method, we set the best value of Δ by using a validation set. The values of Δ closer to 1 indicate that a linear hyperplane approximates the positive class samples well and the best fitting hyperplane is further from the negative samples (and the classes are well separated). Alternately, the values of Δ close to 0 indicate that positive and negative classes samples are close to each other, and it is difficult to separate them.

An alternative approach might be to let the algorithm automatically determine the best value of Δ. This can be performed by adding the Δ term in the objective function as a parameter that must be maximized since bigger values of Δ indicate better separability. In this case, the objective function becomes

$$\begin{aligned}\underset{(\mathbf{w}_+,b_+,\Delta)}{\arg\min}\ \frac{1}{2}||\mathbf{w}_+||^2 - H\Delta + C_+ \sum_{i=1}^{2n_+} R_{pos}(y_i f_\theta(\mathbf{x}_i)) \\ + C_- \sum_{i=2n_++1}^{2n_++2n_-} R_{neg}(y_i f_\theta(\mathbf{x}_i)),\end{aligned}$$

where H is the weight of the Δ parameter that must be set by the user. After applying the similar steps given in the Appendix A, the dual of the convex optimization problem becomes

$$\begin{aligned}\underset{\mathbf{h}}{\arg\min}\ & \sum_{i=1}^{2n_++2n_-}\sum_{j=1}^{2n_++2n_-} h_i h_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle + \sum_{i=1}^{2n_+} y_i h_i \\ & - \sum_{i=2n_++1}^{2n_++2n_-} y_i h_i \\ \text{s.t.}\ & \sum_{i=1}^{2n_++2n_-} h_i = 0; \\ & \sum_{i=1}^{2n_++2n_-} y_i h_i \geq H - \sum_{i=1}^{2n_++2n_-} \beta_i, \\ & -\beta_i \leq y_i h_i \leq C_+ - \beta_i,\ 1 \leq i \leq 2n_+; \\ & -\beta_i \leq y_i h_i \leq C_- - \beta_i,\ 2n_+ + 1 \leq i \leq 2n_+ + 2n_-.\end{aligned} \tag{19}$$

There are two differences compared to our initial convex optimization problem given in Algorithm 2. First, the linear coefficient vector in the objective function changes. Second, an additional inequality constraint appears in the quadratic optimization problem. From Karush-Kuhn-Tucker (KKT) conditions, $\Delta > 0$ implies that the inequality constraint becomes an equality. Therefore, the parameter H behaves like a lower bound on the fraction of support vectors. This is expected because the Δ term is related to both the margin between positive and negative samples and the distance between the two parallel hyperplanes where the positive class samples lie. In our experiments, we preferred to use the convex optimization given in Algorithm 2 since the training time was shorter (current fast quadratic programming solvers only allow equality constraints with lower and upper bounds on the Lagrange coefficients), and we did not observe a significant performance difference between those two formulations.

For positive samples, we use the non-convex cost function plotted in Fig. 3. One may also use the following convex cost function for positive samples

$$R_{pos}(t) \;=\; \arg\max(|t| - (1-\Delta), 0). \tag{20}$$

As shown in the Fig. 7, this cost function can be written as the sum of two Hinge losses. Incorporating this cost function in the proposed method yields a slightly different algorithm. However, it is not robust against the outliers compared to the non-convex cost function we used in our method, and it did not offer any improvements in terms of the accuracy or speed (In fact, the non-convex cost typically yielded better results because it is more robust to outliers.

Similar findings are reported in [29] and [30]). One may also adopt some robust regression functions such as Tukey loss [31] for positive samples. Similar to our proposed robust loss function, Tukey loss is also robust to outliers and non-convex. However, it is computationally expensive compared to our proposed loss function.

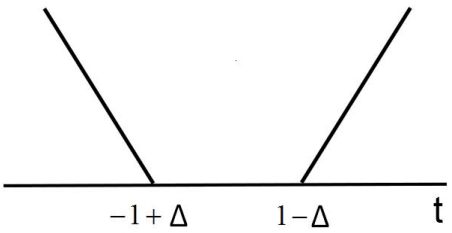


Fig. 7. The convex loss function for positive samples, $R_{pos}(t) = \arg\max(|t| - (1-\Delta), 0)$. It can also be written as the sum of two Hinge losses, $R_{pos}(t) = H_{1+\Delta}(t) + H_{1+\Delta}(-t)$.

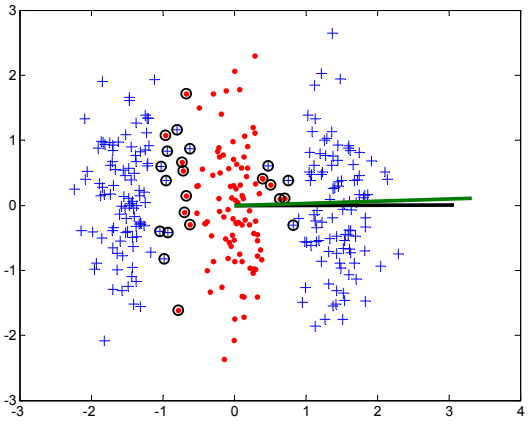


Fig. 8. Synthetic data created by using normal distributions. The support vectors returned by the proposed method are denoted by circles around the samples. Our proposed method finds a better fitting hyperplane (its normal vector is shown by the black line) compared to the GEPSVM's (its normal is the green line).

4 Experiments

We tested the proposed methods[1], 1S-BFHC and 2S-BFHC, on both synthetic and real databases to assess their 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), SVM and robust SVMs [29]. The One-Against-Rest (OAR) regime was used for multi-class classification problems. For the nonlinear (kernelized) classifiers, we only used the Gaussian kernels.

4.1 Experiments on Synthetic Data

We first tested the proposed methods on a 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. 8. The positive class has a mean of $\boldsymbol{\mu} = \left(\begin{smallmatrix}0\\0\end{smallmatrix}\right)$, and x and y dimensions are uncorrelated with corresponding standard deviations of 0.1 and 0.9, respectively. The positive class samples are surrounded with negative samples having the same covariance structure but different means of $\boldsymbol{\mu} = \left(\begin{smallmatrix}-1.5\\0\end{smallmatrix}\right)$ and $\boldsymbol{\mu} = \left(\begin{smallmatrix}1.5\\0\end{smallmatrix}\right)$. Thus, the optimal best fitting hyperplane is closer to the y-axis. Consequently, the normal of the optimal hyperplane must be in the direction of the x-axis.

Among all best fitting hyperplane based methods, only the GEPSVM and our proposed 2S-BFHC methods allow negative samples to lie on both sides of the separating hyperplane. Thus, we compare these two methods on this problem (all other linear fitting hyperplane methods and linear SVM will return bad results since data are not separable by using a single hyperplane). We initialized the normal of the best fitting separating hyperplane with $\mathbf{w}_+^0 = \left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, and the proposed method converged to the solution in 3 iterations and returned the solution of $\mathbf{w}_+ = \left(\begin{smallmatrix}1.397\\0.004\end{smallmatrix}\right)$ 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 the black solid line in Fig. 8. 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, in contrast, returned the normal shown with the green solid line in Fig. 8. For quantitative comparison, we created the same amount of test data (with the same covariance structure but closer means to increase the overlap between positive and negative classes) and computed the Average Precision (AP) scores obtained from the precision-recall curves. The proposed method achieves 0.923, and GEPSVM achieves 0.918.

4.2 Experiments on Visual Object Detection and Classification

4.2.1 Face Detection

For face detection, we collected 12500 subimages of frontal faces from the web for training. The images are rescaled and aligned to a resolution of 35×28. For the negative set, we randomly collected 40000 windows from images without faces. We used Local Binary Patterns (LBP)+Histograms of Oriented Gradients (HOG) as visual features. Classifiers trained with initial samples were used to scan a set of thousands of images to collect both false negatives and false positives. These hard examples were added to the training set and classifiers are retrained. The final size of the training set is over 200K, which can be considered as large-scale data.

We tested linear classifiers on 2845 image FDDB (Face Detection Data set and Benchmark) [32]. To measure the accuracy, we used the PASCAL VOC metric. In this metric, detections are considered as true or false based on the

1. Our software package is available at http://mlcv.ogu.edu.tr/softwares.html. In this package, we also provide the codes for other hyperplane fitting methods (GEPSVM and TSVM) beside the codes for the proposed methods.

TABLE 1
AP Scores (%) on the FDDB Dataset.

Methods	AP Scores
GEPSVM	29.16
SVM	37.60
Robust SVM	62.50
1S-BFHC	70.50
2S-BFHC	73.26
Viola & Jones	67.14
Kalal et al.	66.25
Cevikalp & Triggs	**74.10**

overlap with ground-truth bounding boxes. The overlap between the bounding boxes, R returned by the classifiers, and the ground truth box Q, is computed as $\frac{\text{area}|Q \cap R|}{\text{area}|Q \cup R|}$. In our experiments, bounding boxes with an overlap of 45% or greater are considered as true positives. We then report the average precision (AP) given in Table 2 for the whole precision-recall curves plotted in Fig. 9. In addition, we plotted ROC curves to compare our results to the ones reported at (http://vis-www.cs.umass.edu/fddb/results.html). As a comparison, we also report the accuracies obtained by cascade face detector of Viola&Jones [33], face detector of Kalal et al. [34] and cascade detector of Cevikalp&Triggs [21]. We could not test TSVM classifier because of large training set size. Our proposed 2S-BFHC classifier achieves the second best result after the cascade detector of Cevikalp&Triggs [21]. The proposed 1S-BFHC is third best performing method. GEPSVM and SVM produce very bad results. SVM classifier failed for face detection owing to the fact that many false positives lie above the separating hyperplane far from the separating hyperplane (see Appendix C for illustrative examples). These false positives have high confidence scores which in turn significantly decreases precision. Using Robust SVM formulation significantly improves the SVM performance from 37.6% to 62.50%. The improvement is clearly shown in both Precision-Recall and ROC curves. But the accuracy is still behind the accuracies of our proposed classifiers.

4.2.2 Pedestrian Detection

We trained and tested a series of detectors using 1S-BFHC, 2S-BFHC, SVM and robust SVM classifiers on Inria Person dataset [35] with identitical setting for each. We used the latent training methodology of Felzenszwalb *et al*. [36] and trained a pair of roots without parts. The roots are initialized with applying k-means clustering to mirror-image pairs. We used HOG features as in [36]: 8×8 pixel cells with window steps of 8 pixels and pyramid scales spaced by a factor of 1.07. Table 2 shows the resulting accuracies. 2S-BFHC detector achieves the best result among those trained followed by linear SVM detector. As opposed to the face detection, robust SVM detector performs poorly and it yields approximately 2% lower accuracy than SVM detector.

TABLE 2
AP Scores (%) on the INRIA Person Dataset.

Methods	AP Scores
SVM	80.4
Robust SVM	78.3
1S-BFHC	78.5
2S-BFHC	**82.6**

4.2.3 ILSVRC2013 Object Detection

The ILSVRC2013 detection dataset contains images of everyday scenes with annotation of all full and partial instances of 200 object categories. We test our detectors on 10 categories that are carefully selected from both easy and difficult classes. The tested classes are airplane, bee, bicycle, bow tie, butterfly, camel, car, coffee maker, tennis ball and tv/monitor. We first split images into two groups based on the aspect ratio of their annotated bounding boxes. Then, we applied k-means clustering to divide object images into four classes for each group. So, we had 8 root detectors in total. We used HOG features and latent training methodology as before. The results are shown in Fig. 10. Among tested 3 methods, 1S-BFHC detector achieves the best results for 6 classes and 2S-BFHC detector achieves the best results for the remaining 4 classes. On the average, 2S-BFHC, 1S-BFHC and SVM respectively yields %14.14, %14.01 and %11.34. So, the detector using 2S-BFHC wins with a slight edge in front of 1S-BFHC detector.

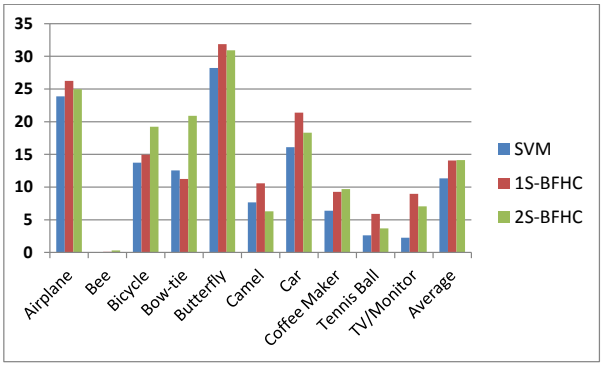
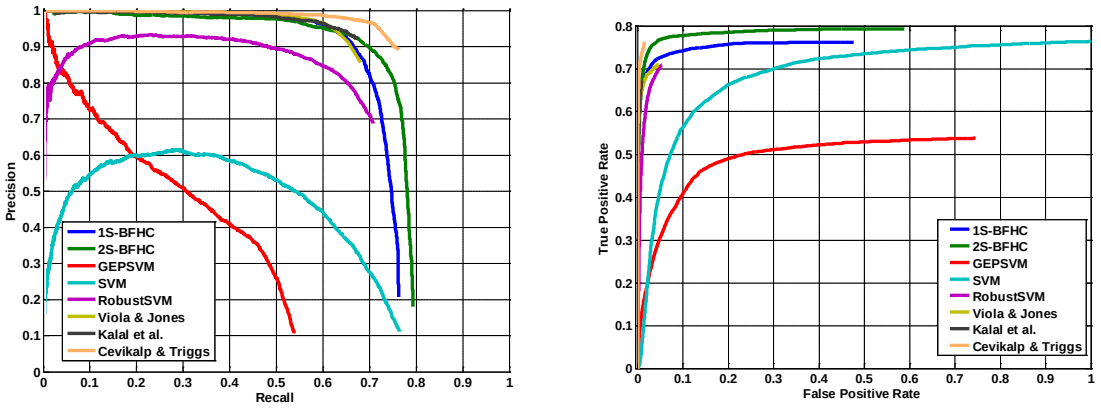


Fig. 10. AP Scores (%) on Selected ImageNet Classes.

4.2.4 Experiments on PASCAL VOC 2007 Dataset

We tested the proposed method on PASCAL VOC 2007 visual object detection and classification tasks. For detection, we used the state-of-art detector of Girshick et al [37,38] using region proposals and CNN (convolutional neural network) features. Briefly, this detector has three stages where the first stage returns region proposals, the second stage extracts 4096 dimensional CNN features, and the last stage includes a set of class-specific linear SVMs. Linear SVM classifiers are trained with negative hard mining since the training set size is too large to fit in memory. We refer readers to [37,38] for more details



(a) (b)

Fig. 9. (a) Precision-Recall and (b) ROC curves for FDDD dataset.

on the detector. We re-trained the linear SVM classifiers using pre-trained CNN models and obtained a mAP of 54.12% without using bounding box regression, which is very close to 54.20% reported in [37]. By using the same features we trained the proposed best fitting hyperplane classifiers by using negative hard mining. In order to speed-up the training phase, at each iteration, we collected the hardest 40K negative examples coming from 50 images. We initialized 2S-BFHC classifier using linear SVM at the beginning. The proposed 2S-BFHC method achieves a mAP of 53.83% which is very close to the one obtained by linear SVM, but 1S-BFHC performs badly and returns 50.76% accuracy as seen in Table 3.

For classification task, we again used 4096-dimensional CNN features. To extract CNN features, all images are first resized to 256×256 and then we used pre-trained ILSVRC2012 Caffe model [39] of the CNN described by Krizhevsky et al. [40]. We did not apply fine-tuning and linear SVM classifier is used to initialize the 2S-BFHC. Results are given in Table 3. The proposed 2S-BFHC achieves the best accuracies for all classes. As opposed to the object detection, we obtained a significant improvement over linear SVMs. The performance difference is very significant especially on bottle, chair, potted-plant and sofa classes (the performance gain is always larger than 5% for these classes).

4.2.5 Experiments on the Caltech-256 and USPS Digit Datasets

Here we test our classifiers on Caltech-256 (http://www.vision.caltech.edu/Image_Datasets/Caltech256/) and USPS (retrieved from http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#usps) datasets. Caltech-256 is a visual object recognition dataset containing images belonging to 256 object categories. There is also an additional background class, making the total number of classes 257. The images have background clutter and significant intra-class and scale variability. The USPS dataset contains 9298 16×16 gray-scale of hand-written digits, with 7291 reserved for training and validation and the remaining 2007 for testing. For USPS we use raw gray-scale values of pixels as features without any pre-processing or feature extraction. For Caltech 256, we follow the standard procedure and pick 60 images from each class and split them into 30 for training and 30 for test. Then, we reverse the role of training and test. Fisher vector (FV) representation is used to represent images and we used the same setup as in [41]. More precisely, we extracted approximately 10K descriptors per image from 24×24 patches on a regular grid every four pixels at 5 scales. The dimensionality of the tested descriptors is reduced to 80 by using Principal Component Analysis. We used 6×10^6 descriptors to learn PCA projections and 256-component Gaussian mixture model (GMM) components. The final dimension of the image FVs is around 164K. The results are given in Table 4 (We could not obtain results for TSVM because of computational complexity of training). For both datasets, the proposed 2S-BFHC yields the best result followed by 1S-BFHC. Accuracies of GEPSVM are again low. The improvement achieved by the 2S-BFHC over linear SVM is especially significant for challenging Caltech-256 dataset.

TABLE 4
Classification Rates (%) on the Caltech-256 and USPS Digit Datasets.

Linear Methods	Caltech 256	USPS
GEPSVM	13.3±0.7	55.2
SVM	37.6± 0.7	91.3
1S-BFHC	38.3± 1.0	91.8
2S-BFHC	**40.1± 0.7**	**92.1**

4.3 Experiments on Open Set Recognition

At Introduction section, we argued that our proposed hyperplane fitting classifiers are better suited for open set recognition problems because they try to approximate the positive class samples with an hyperplane. To support this claim, we set up an open set recognition scenario for two datasets. First we used USPS dataset and randomly

TABLE 3
Average Precision scores (%) on PASCAL VOC 2007 detection and classification datasets.

Detection	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	TV Monitor	Average
SVM	**64.3**	**69.6**	**50.2**	**41.9**	31.9	**62.5**	**70.9**	60.4	32.7	**58.5**	46.3	56.2	60.4	**66.7**	**54.1**	**31.6**	52.9	**48.9**	57.7	64.8	**54.12**
1S-BFHC	61.7	65.2	47.1	35.2	28.3	62.4	67.1	56.5	29.6	55.9	43.9	53.4	55.2	65.5	50.8	28.1	48.5	45.0	54.6	61.5	50.76
2S-BFHC	**64.3**	69.1	49.7	40.8	**32.0**	62.4	70.4	**60.7**	**33.0**	57.5	**48.1**	**56.3**	**61.0**	66.2	**54.1**	31.0	**53.4**	47.1	**57.8**	**64.9**	53.83
Classification																					
SVM	87.0	75.7	81.7	80.4	31.2	63.6	80.4	79.1	47.1	58.1	64.2	74.0	81.0	73.0	87.4	41.3	68.5	50.6	86.3	61.4	68.60
1S-BFHC	85.9	74.0	79.9	77.4	30.3	63.0	78.5	78.0	46.2	56.6	62.0	72.0	79.7	71.9	83.2	39.2	63.1	51.0	84.4	59.5	66.77
2S-BFHC	**88.4**	**76.8**	**82.3**	**80.9**	**37.6**	**65.8**	**82.7**	**80.2**	**56.6**	**59.7**	**67.0**	**77.7**	**82.3**	**74.2**	**89.3**	**47.3**	**70.2**	**57.3**	**86.5**	**64.1**	**71.34**

TABLE 5
Classification Rates (%) on the Open Set USPS Dataset.

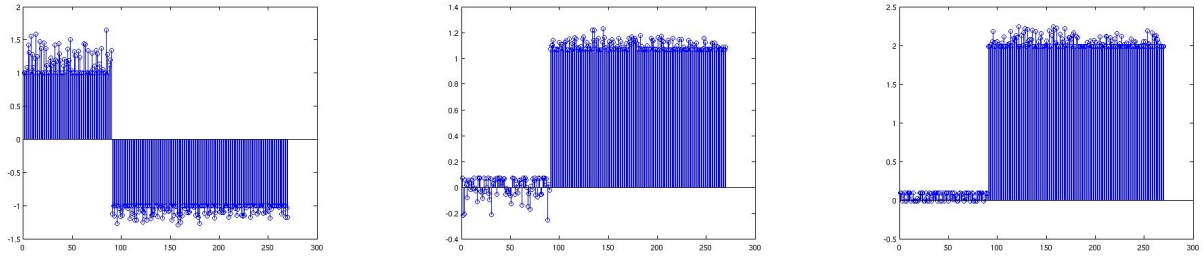
Linear Methods	AP Scores
GEPSVM	40.51
SVM	61.90
1S-BFHC	66.01
2S-BFHC	**67.81**

chose 3 positive classes. For each positive class, training dataset is created by using all training samples belonging to the positive classes. During testing, we used the samples belonging to the positive classes in the allocated test set as well as test samples belonging to other 7 classes that are not used for training. Then we compute AP scores from Precision-Recall curves for each class and take the average as the final score. This procedure is repeated 10 times and the final accuracies are the averages of accuracies obtained in each trial. The results are given in Table 5. The best accuracy is obtained by the proposed 2S-BFHC initialized with 1S-BFHC. The proposed 1S-BFHC comes the second followed by the SVM. If we initialize the 2S-BFHC with SVM, which performs poorly compared to 1S-BFHC, the accuracy is 62.73%.

For the second set of open set recognition experiments, we have chosen three classes (aeroplanes, cars, and faces) having more than 120 samples per class from the Caltech 101 database. Then, we obtained bag of words models by using images coming from only these three classes. We used 90 images for training the classifiers and 30 images for testing. During testing, we use the test samples of three classes and randomly choose 2000 samples from the remaining classes for testing. To measure the accuracy, we use AP scores as before. We tested only linear proposed classifiers and SVM since other best fitting algorithms perform poorly on this problem. We set the dimensionality of the image histograms to different values between 50 and 10000 to demonstrate the performance change as a function of the dimensionality. These experiments are repeated 5 times by choosing random samples for training and testing, and the final accuracies are averages over these 5 trials in Table 6. The best classification accuracies are typically obtained by using the proposed 2S-BFHC. In this setup, the proposed method significantly outperforms SVM when the dimensions of the image feature histograms are 50, 7000 and 10000, whereas the accuracies are similar when the dimensions are set to 500 and 1000. It is expected that 2S-BFHC will outperform both SVM and 1S-BFHC classifiers in open set recognition setup when the image histogram sizes are small (see the case for dimensionality is equal to 50). It is because of the fact the positive class samples cannot be separated from the other samples with a single hyperplane since they typically lie in specific regions surrounded by a diffuse sea of so-called background samples. However, it was surprising that both of our proposed hyperplane fitting classifiers achieved better results than SVM when the dimensionality of the image histograms is very high. We observed the remarkable fact that SVM also behaves like an hyperplane fitting classifier for high-dimensional spaces. Majority of the training samples (around 55% of the whole training data) lie on the supporting hyperplanes as shown in Fig. 11, which illustrates the signed distances from samples to the separating/best-fitting hyperplanes for dimension 7000 (the figure is plotted for only one class trained during OAR regime). Since these samples become support vectors, the common belief that "only a few of the training samples become support vectors" does not hold anymore for high-dimensional spaces.

5 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. However, existing hyperplane fitting algorithms have two major limitations: They are not suitable for large-scale classification problems since one has to perform eigen-decomposition on the resulting large matrices or the inverses of those large matrices must be found. 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 novel hyperplane fitting classifiers that do not have the limitations mentioned above. More precisely, the proposed classifiers use the SMO algorithm, which does not require constructing large



(a) (b) (c)

Fig. 11. Signed distances from the training samples to the separating/best-fitting hyperplanes: (a) distances for SVM, (b) distances for 1S-BFHC, b) distances for 2S-BFHC.

TABLE 6
Classification Rates (%) as a Function of Dimensionality for the Open Set Recognition Setup.

Dimension 50	SVM	9.34, $\sigma = 2.4$
	1S-BFHC	8.93, $\sigma = 1.1$
	2S-BFHC	**14.07**, $\sigma = 1.6$
Dimension 500	SVM	24.81, $\sigma = 3.3$
	1S-BFHC	24.27, $\sigma = 5.7$
	2S-BFHC	**25.04**, $\sigma = 5.9$
Dimension 1000	SVM	**25.40**, $\sigma = 4.9$
	1S-BFHC	24.88, $\sigma = 3.7$
	2S-BFHC	24.05, $\sigma = 4.1$
Dimension 7000	SVM	37.35, $\sigma = 3.0$
	1S-BFHC	42.00, $\sigma = 6.2$
	2S-BFHC	**43.76**, $\sigma = 6.3$
Dimension 10000	SVM	39.65, $\sigma = 2.9$
	1S-BFHC	42.33, $\sigma = 5.9$
	2S-BFHC	**44.19**, $\sigma = 5.6$

Hessian matrices, making the methods suitable for large-scale problems. Moreover, the returned solutions are sparse similar to SVMs, and thus the proposed methods are efficient in terms of testing time.

We tested the classification accuracies of the proposed methods on both synthetic and real-world classification problems. The proposed methods typically outperformed other best-fitting hyperplane classifiers in most of the cases, and they produced comparable results to the SVM classifier in classical classification tasks. Other hyperplane fitting classifiers such as GEPSVM and TSVM performed poorly on most challenging data sets where the number of classes is large or the dimensionality of the input space is high. On the other hand, the proposed methods significantly outperformed SVMs for the open set recognition and object detection tasks. Especially, performance difference is too significant (around 36%) for face detection. These results verify that our prosed best fitting hyperplane methods are more suitable than SVMs for open set recognition and object detection problems.

ACKNOWLEDGMENTS

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

REFERENCES

[1] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

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

[3] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training svm. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[4] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Learning*, pages 169–184. MIT Press, Cambridge, 1999.

[5] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[6] H. Cevikalp, B. Triggs, H. S. Yavuz, Y. Kucuk, M. Kucuk, and A. Barkana. Large margin classifiers based on affine hulls. *Neurocomputing*, 73:3160–3168, 2010.

[7] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

[8] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proc. of Knowledge Discovery and Data Mining*, pages 77–86, 2001.

[9] G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2003.

[10] Z. Deng, F. L. Chung, and S. Wang. A new minimax probability based classifier using fuzzy hyperellipsoid. In *Proc. of International Joint Conference on Neural Networks*, 2007.

[11] H. Cevikalp and B. Triggs. Hyperdisk based large margin classifier. *Pattern Recognition*, 46:1523–1531, 2013.

[12] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vcetor machine classification via generalized eigenvalues. *IEEE T-PAMI*, 28:69–74, 2006.

[13] Jayadeva, R. Khemchandani, and S. Chandra. Twin support vector machines for pattern classification. *IEEE T-PAMI*, 29:905–910, 2007.

[14] Y.H. Shao, C. H. Zhang, X. B. Wang, and N. Y. Deng. Improvements on twin support vector machines. *IEEE Transactions on Neural Networks*, 22:962–968, 2011.

[15] M. A. Kumar and M. Gopal. Least squares twin support vector machines fpr pattern classification. *Expert Systems with Applications*, 36:7535–7543, 2009.

[16] M. A. Kumar and M. Gopal. Application of smoothing tehcnique on twin support vector machines. *Pattern Recognition Letters*, 29:1842–1848, 2008.

[17] S. Gao, Q. Ye, and N. Ye. 1-norm least squares twin support vectorm machines. *Neurocomputing*, 74:3590–3597, 2011.

[18] X. Peng. Tpmsvm: A novel twin parametric-margin support vector machine for pattern recognition. *Pattern Recognition*, 44:2678–2692, 2011.

[19] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE T-PAMI*, 35:1757–1772, 2013.

[20] H. Cevikalp, B. Triggs, and V. Franc. Face and landmark detection by using cascade of classifiers. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2013.

[21] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[22] A. L. Yuille and A. Rangarajan. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems*, 2002.

[23] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[24] M. B. Gulmezoglu, V. Dzhafarov, and A. Barkana. The common vector approach and its relation to principal component analysis. *IEEE Trans. Speech Audio Proc.*, 9:655–662, 2001.

[25] H. Cevikalp, M. Neamtu, M. Wilkes, and A. Barkana. Discriminative common vectors for face recognition. *IEEE T-PAMI*, 27:4–13, 2005.

[26] H. Cevikalp and M. Wilkes. Face recognition by using discriminative common vectors. In *International Conference on Pattern Recognition*, 2004.

[27] H. Cevikalp, M. Neamtu, and M. Wilkes. Discriminative common vector method with kernels. *IEEE Transactions on Neural Networks*, 17:1550–1565, 2006.

[28] H. Cevikalp, M. Neamtu, and A. Barkana. The kernel common vector method: A novel nonlinear subspace classifier for pattern recognition. *IEEE Transactions on Systems, Mand and Cybernetics, Part B: Cybernetics*, 37:937–951, 2007.

[29] S. Ertekin, L. Bottou, and C. L. Giles. Nonconvex online support vector machines. *IEEE Transactions on on Pattern Analysis and Machine Intelligence*, 33:368–381, 2011.

[30] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *International Conference on Machine Learning*, 2006.

[31] John Fox. Robust regression: Appendix to an r and s-plus companion to applied regression, 2002.

[32] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[33] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

[34] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.

[35] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[36] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE T-PAMI*, 32(9), September 2010.

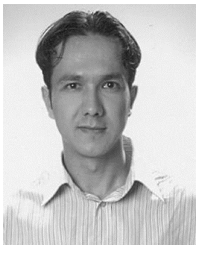
[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convoulutional networks for accurate object detection and segmentation. *IEEE T-PAMI*, 38:142–158, 2015.

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[41] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 34:1704–1716, 2013.



Hakan Cevikalp received his M.S. degree from the Department of Electrical and Electronics Engineering, Eskisehir Osmangazi University, Eskisehir, Turkey, in 2001 and his Ph. D. degree from the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, in 2005. He is currently working as an Associate Professor at Electrical and Electronics Engineering department of Eskisehir Osmangazi University, Eskisehir, Turkey. His research interests include pattern recognition, neural networks, image and signal processing, optimization, and computer vision. He is a member of the IEEE.