Polyhedral Conic Classifiers for Computer Vision Applications and Open Set Recognition

Hakan Cevikalp, Halil Saglamlar
Eskisehir Osmangazi University, Electrical and Electronics Engineering, Eskisehir, Turkey.
hakan.cevikalp@gmail.com, hsaglamlar@gmail.com

Abstract—This paper introduces a family of quasi-linear discriminants that outperform current large-margin methods in sliding window visual object detection and open set recognition tasks. In these applications, the classification problems are both numerically imbalanced – positive (object class) training and test windows are much rarer than negative (non-class) ones – and geometrically asymmetric – the positive samples typically form compact, visually-coherent groups while negatives are much more diverse, including anything at all that is not a well-centered sample from the target class. For such tasks, there is a need for discriminants whose decision regions focus on tightly circumscribing the positive class, while still taking account of negatives in zones where the two classes overlap. To this end, we propose a family of quasi-linear "polyhedral conic" discriminants whose positive regions are distorted L_1 or L_2 balls. In addition, we also integrated the proposed classification loss into deep neural networks so that both the features and classifier can be learned simultaneously end-to-end fashion to improve the classification accuracies. The methods have properties and run-time complexities comparable to linear Support Vector Machines (SVMs), and they can be trained from either binary or positive-only samples using constrained quadratic programs related to SVMs. Our experiments show that they significantly outperform linear SVMs, deep neural networks using softmax loss function and existing one-class discriminants on a wide range of object detection, face verification, open set recognition and conventional closed-set classification tasks.

Index Terms—Polyhedral conic classifiers, object detection, large margin classifiers, open set recognition.

1 Introduction

Conventional machine learning classifiers such as linear Support Vector Machine (SVM) [1] and other large-margin classifiers [2], [3], [4], [5] have been successfully used in many fields including computer vision, text analysis, biometrics and bioinformatics. In general, SVM classifier 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. If the data are not linearly separable, the kernel trick is used to estimate the non-linear decision boundaries, but the kernelized SVM classifier becomes too slow for many real-time applications such as visual object detection and tracking.

Linear SVMs and other conventional linear large-margin discriminants are intended for "closed set" scenarios [6] in which the class labels are mutually exclusive and exhaustive and every class seen at test time is known during training. These methods try to attribute each test sample to a class even when it has little resemblance to the training samples of any known one – a semantics that is fragile because it ignores the possibility that outliers (samples with no meaningful class) and novel classes (ones not foreseen during training) may occur at test time. In contrast, "open set" methods [6] try to handle these issues by rejecting test samples that do not appear to belong to any of the known training classes. To do this, they need to estimate some kind of inlier or validation region for each target class in addition to the conventional inter-class decision boundaries.

Visual object detection and face verification also benefit from discriminants that tightly constrain the positive class. In sliding-window object detection, the discrimination problem is highly asymmetric because the positive samples (windows that correctly frame instances of the target class) form a variable-but-coherent appearance class, whereas the negatives (anything at all that is not a well framed object instance) are much more diverse. Moreover the data is highly imbalanced in that there are many more negative (non-object) training and test windows than positive (object) ones. For both reasons it is useful for the discriminant to focus on tightly bounding the positive class whereas conventional discriminants such as SVMs treat the two classes as though they were equal, interchangeable alternatives. Owing to the many ways in which a window can fail to be a positive, most of the SVM support vectors turn out to be 'hard negatives' and with existing feature sets it is not unusual to find that these completely surround the positives in feature space (*c.f.* the scatter plots of projected class densities in [7], [8]). In a similar manner, face verification problems assume images of a single individual as the positive class while the set of images of all other possible people are treated as potential impostors. Therefore, it is impossible to know all testing classes during training, thus we need to estimate some kind of validation region for each individual person of interest.

For visual object detection and verification applications as well as for more general open set recognition tasks, there is a need for reliable, scalable, asymmetric discriminants that focus on modeling the positive class as a compact, coherent set surrounded by a disparate sea of negatives. The pitfalls of not doing so are illustrated in Fig. 1. This is for a recognition problem in which unforeseen classes occur at run time, but object detectors face similar issues with unforeseen kinds of hard negatives.

This paper introduces a new family of quasi-linear discriminants that return compact polyhedral acceptance regions for positive class samples based on linear sections through L_1/L_2 cones. By supplying tighter bounds on the positive class, this geometry systematically outperforms half-space based decision rules such as linear SVMs in both open set recognition problems and

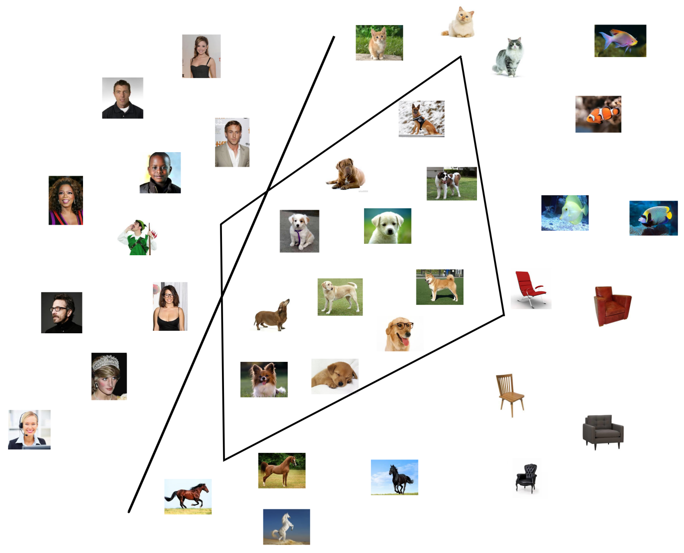


Fig. 1. A decision hyperplane returned by an SVM successfully separates its training classes, dogs (positive) and people (negative). However it also assigns instances of new classes such as cats, horses, fish and chairs to the dog class, sometimes with higher confidence scores than for dogs themselves. The problem is the too large acceptance region – SVM only tries to separate dogs and people, not to bound the dog class. A tighter (*e.g.* polyhedral or ellipsoidal) decision boundary improves classification, reducing mis-classifications caused by unforeseen classes and outliers.

detection/verification problems with unforeseen hard negatives. In fact it often improves the performance even in conventional closed set problems especially when the number of classes is large since the proposed classifiers are better suited for the one-against-all classification rule. Training of the proposed classifiers is formulated as an efficient convex program as for linear SVM, and run times are also similar to linear SVM. Preliminary version of this paper has appeared in [9]. This paper extends our previous work with (1) a more detailed analysis of the recent related work on polyhedral conic functions; (2) a more detailed description of the proposed methods and introducing new extensions using L_2 norm in addition to L_1 norm; (3) introduction of a novel method that finds the cone vertex and classifier parameters optimally; (4) implementing deep neural networks using the proposed loss function to handle different classification scenarios such as one-class, multi-class and multi-label classification problems, and (5) more experiments on new datasets.

Related Work: Recently, several studies introduced classifiers (especially for object detection) that abandon the formal classification setup and adopt loss functions designed to provide tighter modeling of the positive class. These are typically called one-class[1] classifiers and they can learn the positive class models even by using the positive class samples only. Several methods have been proposed in this direction. Support Vector Data Description (SVDD) method of [10] aims to find a closed boundary around the positive data. To this end, it finds a compact hypersphere that includes the majority of the positive class samples. Cevikalp and Triggs [11], [12] use a cascade of nearest convex model classifiers to progressively cut out a compact, coherent positive region from a broad sea of negative examples for face and people detection. Scheirer et al. [6], [13] and Rudd et al. [14] introduced classifiers for open set recognition problems where the test set includes samples of novel classes that are not foreseen during training. The generalized eigenvalue proximal support vector machine (GEPSVM) classifier [5] finds a hyperplane that best fits the positive class samples while avoiding the negative samples as far as possible. By following a similar idea, [15] proposed the Twin Support Vector Machine (TSVM) classifier. This classifier also finds a best fitting hyperplane but it is found by solving a quadratic programming problem. Different variants of the best fitting hyperplane classifiers are proposed in [16], [17].

Other approaches such as Additive Kernels [18] and Random features [19] try to approximate kernel classifiers in a fixed-complexity setting by explicitly mapping samples to higher-dimensional spaces that provide nonlinear class separation circumscribing the positive class region. Similar to these approaches, the proposed method can also be seen as a nonlinear classifier where the linearly non-separable data samples are explicitly mapped to a higher-dimensional feature space that allows the polyhedral conic separation.

Another strategy is exemplified by the colorectal cancer detector of Dundar et al. [20], which learns polyhedral acceptance regions by jointly optimizing a set of hyperplane classifiers, each designed to classify positives against a subgroup of the negative samples. However the required partitioning of the negative set is both expensive for large-scale problems and problematic if the negatives do not naturally separate into well defined clusters, particularly as the overall performance turns out to be sensitive to both the number and the detailed form of the partitions. Several studies [21], [22] focused on different techniques to construct polyhedra that approximately bound positive classes. However these methods scale poorly with training set size, suffer from local optima or over-fitting, or need ancillary clustering or labeling which makes them unsuitable for large-scale applications. Manwani et al. [23] proposed a classifier using logistic function to learn polyhedral acceptance regions, whereas they used a perceptron-like algorithm for the same goal in [24]. However, they form the polyhedral acceptance regions by using intersection of several hyperplanes, whose number must be pre-defined before training. This is a major limitation since we do not have a-priori information on the number of hyperplanes for most of the classification problems. Furthermore, these methods are only tested on small datasets and comparisons are made against a very limited number of classifiers.

Kantchelian et al. [25] proposed Convex Polytope Machine (CPM) classifier to find polyhedral acceptance regions for large-scale problems. As in [23], [24], they also learn a fixed amount of linear hyperplanes for separating positive class samples from the negatives. However, there is no guarantee that the resulting classifier forms polyhedral acceptance regions. In fact, CPM classifier is equivalent to latent SVM training methodology of [26], that is widely used for visual object detection problems. Latent SVM just learns a set of linear hyperplanes separating positive samples lying in different sub-regions of the input space from the negatives. The resulting acceptance region is not necessarily a polyhedral acceptance region since latent SVM does not enforce the positives to lie in the intersection of the hyperplanes.

In contrast to these methods mentioned above, our proposed methods have a convex formulation that ensures globally optimal solutions, they scale efficiently to large problems, they do not

1. The name “one class” is conventional. It emphasizes the origin of these methods in density modeling and the predominant role of the positive class but it is something of a misnomer in that negative examples usually can be, often are, and in some formulations must be included during training.

require negative samples to be clustered, they resist over-fitting by using a robust margin-based cost function, and there is no need to define the number of linear hyperplanes in advance.

There are also deep learning based studies showing the importance of the compact class decision boundaries in classification problems [27], [28], [29], [30]. All these studies argue that the accuracies of deep neural network classifiers using softmax loss function can be significantly improved by enforcing the networks to return more compact class decision boundaries. To this end, [27] uses center loss function along with softmax loss to minimize the intra-class variations while keeping the features of different classes separable. A similar strategy is also adopted in [28], where an additional loss term minimizing the pair-wise sample distances in the same classes is used with softmax loss. In addition to these, there are deep learning based distance metric learning methods proposed to return more compact class decision boundaries by minimizing the intra-class variations [29], [30]. As opposed to these methods, our deep neural network classifiers using the proposed polyhedral conic functions use a single loss function that minimizes the intra-class variations and maximizes the inter-class separability at the same time.

2 Method

We build our methods based on polyhedral conic functions defined in [21]. We formulate the learning problem as a constrained quadratic programming (QP) problem, which makes the method suitable for large-scale classification. We also propose several variants of polyhedral conic functions to estimate the positive class regions by using different norms and extensions.

2.1 Preliminaries

Here we describe different notions of separability and make a brief introduction to polyhedral conic separation. Assume that we are given two nonempty sets of classes denoted by S_+ (positive class) and S_- (negative class) including samples in $\mathbb{R}^d$. It is well-known that if the convex hulls of such sets do not overlap, i.e., $co(S_+) \cap co(S_-) = \emptyset$, a hyperplane separates these two classes. In such a case, a linear SVM can be used to find the best separating hyperplane[2]. Whenever two classes cannot be separated by a hyperplane, i.e., $co(S_+) \cap co(S_-) \neq \emptyset$, but the convex hull of the positive class and the samples of negative class do not intersect, i.e., $co(S_+) \cap S_- = \emptyset$, they are h-polyhedrally separable [32]. That is, there exists h hyperplanes such that the samples of S_+ are contained in a convex polyhedron (intersection of h half-spaces) and the samples of S_- are left outside the polyhedron. [33] introduced the notion of max-min separability which can be considered as a generalization of the h-polyhedral separability. It was shown that if the sets S_+ and S_- are disjoint, then they are max-min separable. Gasimov and Ozturk [21] defined polyhedral conic functions which are used to construct a separation function for the given two arbitrary finite point disjoint sets. These functions are formed by using an augmented L_1 norm with a linear part added. A graph of such a function is a polyhedral cone with a sub-level set including at the utmost an intersection of 2^d half spaces. See Fig. 2 for visualization of different separation types.

2. A best separating hyperplane can be found even if the convex hulls are slightly overlap. The keen reader is referred to [31] for more information.

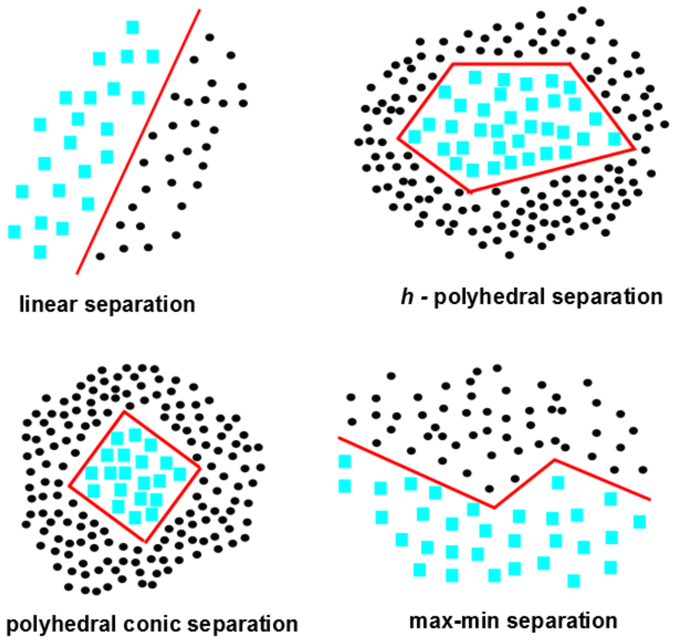


Fig. 2. Geometric interpretation of four separation types: linear, h-polyhedral, polyhedral conic and max-min separation.

2.2 Polyhedral Conic Classifiers

Consider a classification problem with training data given in the form $\{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, n$, $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in \{-1, +1\}$. We first need the following definition from [21].

Definition 1. A function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is called polyhedral conic if its graph is a cone and all its level sets

$$S_\alpha = \left\{ \mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) \leq \alpha \right\} \tag{1}$$

for $\alpha \in \mathbb{R}$, are polyhedrons.

Let us define a polyhedral conic function (PCF) – essentially projections of hyperplane sections through L_1 cones – $f_{\mathbf{w},\gamma,\mathbf{c},b}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ as

$$f(\mathbf{x}) = \mathbf{w}^\top(\mathbf{x} - \mathbf{c}) + \gamma||\mathbf{x} - \mathbf{c}||_1 - b, \quad \text{(PCF)}. \tag{2}$$

Here $\mathbf{x} \in \mathbb{R}^d$ is a test point, $\mathbf{c} \in \mathbb{R}^d$ is the cone vertex, $\mathbf{w} \in \mathbb{R}^d$ is a weight vector and b is an offset. The term, $\|\mathbf{u}\|_1 = \sum_{i=1}^d |u_i|$ denotes the vector L_1 norm and γ is a corresponding weight. The fact that such a function defines a polyhedral cone follows from the following Lemma [21].

Lemma 2.1. *A graph of the function $f_{\mathbf{w},\mathbf{c},\gamma,b}(\mathbf{x})$ defined in (2) is a polyhedral cone with a vertex at $(\mathbf{c}, -b)$.*

In [9], we defined extended polyhedral conic function (EPCF) as

$$f_{\mathbf{w},\boldsymbol{\gamma},\mathbf{c},b}(\mathbf{x}) = \mathbf{w}^\top(\mathbf{x} - \mathbf{c}) + \boldsymbol{\gamma}^\top|\mathbf{x} - \mathbf{c}| - b \qquad \text{(EPCF)} \tag{3}$$

where $|\mathbf{u}| = (|u_1|, ..., |u_d|)^\top$ denotes the component-wise modulus and $\boldsymbol{\gamma} \in \mathbb{R}^d$ is a corresponding weight vector.

Our classifiers use these polyhedral conic functions to define their acceptance regions for positives. This choice provides a convenient family of compact and convex (for suitable weights) region shapes for discriminating relatively well localized positive classes from broader negative ones. It naturally allows robust margin-based learning, and the number of free parameters remains modest, thus controlling both over-fitting and run times. The proposed classifiers have decision regions $f(\mathbf{x}) < 0$ for positives and $f(\mathbf{x}) > 0$ for negatives. Similarly, our margin based training methods enforce $f(\mathbf{x}) \leq -1$ for positives and $f(\mathbf{x}) \geq +1$ for negatives. In both cases the positive region is essentially a hyperplane-section through an L_1 cone centered at $\mathbf{c}$, specifically the region $\mathbf{x} \in \mathbb{R}^d$ in which the hyperplane $z = \mathbf{w}^\top(\mathbf{x} - \mathbf{c}) - b$ lies above the L_1 cone $z = \gamma\|\mathbf{x} - \mathbf{c}\|_1$ (PCF) or the diagonally-scaled L_1 cone $z = \boldsymbol{\gamma}^\top|\mathbf{x} - \mathbf{c}| = \|\mathrm{diag}(\boldsymbol{\gamma})\,(\mathbf{x} - \mathbf{c})\|_1$ (EPCF). See Fig. 3.

Note that for PCF with $b > 0$, $\gamma > 0$, $\|\mathbf{w}\|_\infty < \gamma$ (where $\|\mathbf{u}\|_\infty = \max_{i=1}^d |u_i|$ is the ∞ norm) and any τ, the region $f(\mathbf{x}) < \tau$ is convex and compact in $\mathbb{R}^d$ and it contains the vertex $\mathbf{c}$. Analogously, for EPCF with $b > 0$, $\boldsymbol{\gamma} > 0$, $|w_i| < \gamma_i, i = 1, ..., d$, and any τ, the region $f(\mathbf{x}) < \tau$ is again convex and compact and it again contains $\mathbf{c}$. It would be straightforward to enforce these inequalities during learning but at present we simply leave the decision regions free to adapt to the training data: compact positive classes naturally tend to produce compact acceptance regions in any case.

Geometrically, under the above constraints the resulting regions are bounded octahedroids with $2d$ vertices, one along each positive and negative coordinate half-axis starting from $\mathbf{c}$. The lines joining opposite vertices thus intersect at $\mathbf{c}$, giving the region a deformed but still axis-aligned octahedral "kite" shape with overall size governed by b. In EPCF, the region widths can be scaled independently along each axis, while in PCF they are coupled together but a more limited form of anisotropy is still possible.

To define margin-based classifiers over input feature vectors $\mathbf{x}$ from this, for PCF we augment the feature vector to $\tilde{\mathbf{x}} \equiv \left(\begin{smallmatrix} \mathbf{x}-\mathbf{c} \\ \|\mathbf{x}-\mathbf{c}\|_1 \end{smallmatrix}\right) \in \mathbb{R}^{d+1}$ and the weight vector to $\tilde{\mathbf{w}} \equiv \left(\begin{smallmatrix} -\mathbf{w} \\ -\gamma \end{smallmatrix}\right) \in \mathbb{R}^{d+1}$, and let $\tilde{b} = b$. Then the PCF decision function takes the familiar linear SVM form $\tilde{\mathbf{w}}^\top\tilde{\mathbf{x}} + \tilde{b} > 0$ for positives and $\tilde{\mathbf{w}}^\top\tilde{\mathbf{x}} + \tilde{b} < 0$ for negatives. Similarly, for EPCF we augment the feature vector to $\tilde{\mathbf{x}} \equiv \left(\begin{smallmatrix} \mathbf{x}-\mathbf{c} \\ |\mathbf{x}-\mathbf{c}| \end{smallmatrix}\right) \in \mathbb{R}^{2d}$ and the weight vector to $\tilde{\mathbf{w}} \equiv \left(\begin{smallmatrix} -\mathbf{w} \\ -\boldsymbol{\gamma} \end{smallmatrix}\right) \in \mathbb{R}^{2d}$ and again let $\tilde{b} = b$, again giving the SVM form $\tilde{\mathbf{w}}^\top\tilde{\mathbf{x}} + \tilde{b} > 0$ for positives, but now in $2d$ dimensions. The above ∓1 margins for PCF and EPCF translate to the familiar ±1 SVM margins, allowing us to use standard SVM software for maximum margin training[3]. It thus suffices to run the familiar SVM quadratic program on the augmented feature vectors:

$$\begin{aligned} \underset{\tilde{\mathbf{w}},\tilde{b}}{\arg\min} \quad & \tfrac{1}{2}\tilde{\mathbf{w}}^\top\tilde{\mathbf{w}} + C_+ \textstyle\sum_i \xi_i + C_- \sum_j \xi_j \\ \text{s.t.} \quad & \tilde{\mathbf{w}}^\top\tilde{\mathbf{x}}_i + \tilde{b} + \xi_i \geq +1, \;\; i \in I_+, \\ & \tilde{\mathbf{w}}^\top\tilde{\mathbf{x}}_j + \tilde{b} - \xi_j \leq -1, \;\; j \in I_-, \\ & \xi_i, \, \xi_j \geq 0, \end{aligned} \tag{4}$$

where the $I_\pm$ are indexing sets for the positive and negative training samples, the ξ's are slack variables for the samples' margin constraint violations, and the $C_\pm$ are corresponding penalty weights.

Inserting the PCF and EPCF feature vectors into the above training procedure respectively gives our *Polyhedral Conic Classifier* (PCC) and *Extended Polyhedral Conic Classifier* (EPCC) methods. Note that despite their ostensibly linear symmetric form, these classifiers are intrinsically asymmetric: they force the positives to lie inside, and the negatives to lie outside, polyhedral conic regions that are typically compact and centered on the positives. Our formulation is robust to over-fitting and it scales well because standard SVM technology such as cutting plane methods [34] and fast primal space solvers (*e.g.* [35]) can be used.

3. This only holds if we agree to ignore the optional compact-convex-region constraints $\|\mathbf{w}\|_\infty < \gamma$ (PCC) or $|w_i| < \gamma_i, i = 1, ..., d$ (EPCC).

The above procedure does not attempt to optimize the position $\mathbf{c}$ of the cone vertex as that would lead to a non-convex problem. We can simply set it to a pre-specified position in the positive training set. The mean, medoid, or coordinate-wise median of the training positives can all be used for this with good results. We mostly used the mean in our experiments. The intuition for using positive class mean is that the resulting classifier assigns its highest positive confidence scores to the samples near the cone vertex. We also introduced a method to estimate the best cone vertex position optimally in Section 2.3 and conducted experiments to compare the accuracies obtained by the optimum point and the positive class mean in Section 3.7.

One-Class EPCC (OC-EPCC): EPCC usually outperforms both linear SVM and PCC owing to its flexibility, but its positive acceptance regions are bounded and convex only when $|w_i| < \gamma_i$ for all i – *i.e.* when the hyperplane section has a shallower slope than every facet of the L_1 cone. This sometimes fails to hold for feature space dimensions along which the negatives do not surround the positives on all sides. Even though such EPCC acceptance regions are typically still much smaller than the corresponding linear SVM ones, to ensure tighter bounding we would like to enforce $|w_i| < \gamma_i$, $i = 1, \ldots, d$. Moreover, in EPCC the ∓1 margin is the only thing that fixes the overall weight scale and hence prevents a degenerate solution, and negative data is essential for this. To ensure that EPCC works well for open set problems and ones with only positive samples, we need to force its acceptance regions to stay bounded and compact. The acceptance region has width $O(b/\gamma_i)$ along axis i, so we need to ensure that the γ_i can not shrink to zero. The easiest way to achieve this is to replace the ±1 margin scaling with a $b = 1$ offset scaling and include negative cost penalties on the γ_i and on the geometric width of the new positive-negative margin $[0, 1]$ so that these quantities will tend to increase and hence keep the acceptance region widths small and the sets well separated. This leads to the following "One-Class EPCC" formulation:

$$\begin{aligned} \underset{\mathbf{w},\boldsymbol{\gamma}}{\arg\min} \quad & \tfrac{\lambda}{2}\mathbf{w}^\top\mathbf{w} + \tfrac{1}{n_+}\textstyle\sum_i \xi_i + \tfrac{1}{n_-}\sum_j \xi_j - \mathbf{s}^\top\boldsymbol{\gamma} \\ \text{s.t.} \quad & \mathbf{w}^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}^\top|\mathbf{x}_i - \mathbf{c}| - 1 \leq \xi_i, \;\; i \in I_+, \\ & \mathbf{w}^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}^\top|\mathbf{x}_i - \mathbf{c}| - 1 \geq 1 - \xi_j, \;\; j \in I_-, \\ & \xi_i, \, \xi_j \geq 0. \qquad \text{(OC-EPCC)} \end{aligned} \tag{5}$$

Here λ is a regularization weight for $\mathbf{w}$ and $\mathbf{s} > 0$ is a user-supplied vector of cost penalties for increasing $\boldsymbol{\gamma}$. At present we use the simple stochastic gradient (SG) method given in Algorithm 1 to solve this optimization problem. For $\mathbf{s}$, we set all its entries to the same value. Experimental results show that the accuracy is very sensitive to the values of $\mathbf{s}$. To fix λ, we tried four values (0.01, 0.001, 0.0001, 0.00001), whereas we used five values starting from 0.1 along with other four values used for λ to set $\mathbf{s}$. A grid search algorithm is used to determine the best values on a small randomly sampled validation data chosen from training sets.

$\mathbf{L_2}$ Norm Extensions: If we use L_2 norm instead of L_1 norm

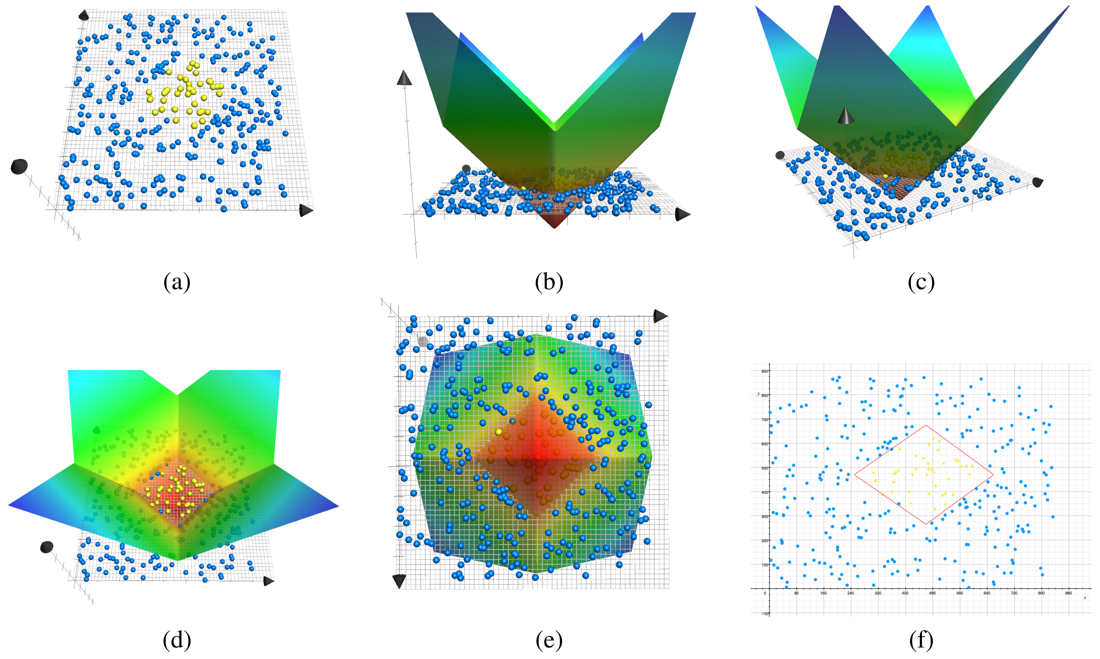


Fig. 3. Visualization of PCC classifiers for 2D synthetic data: The positive acceptance regions are "kite-like" octahedroids containing the points for which a linear hyperplane lies above an L_1 cone.(a): 2D positive (yellow) and negative (blue) samples, (b)-(e): views of positive-class acceptance regions from different angles in 3D, (f): Resulting kite-like acceptance region in 2D space.

Algorithm 1 Stochastic Gradient Based Solver for One-Class EPCC

```
Initialize
$\mathbf{w}_1, \boldsymbol{\gamma}_1, T > 0, \alpha_0 > 0, \epsilon_{\mathbf{w}} > 0, \epsilon_{\boldsymbol{\gamma}} > 0$, $n_+$ is the number of positive examples, $n_-$ is the number of negative examples, $n = n_+ + n_-$
Description:
  for $t \in 1, ..., T$ do
    $\alpha_t \leftarrow \alpha_0/t$;
    $\mathbf{w}_{t-1} = \mathbf{w}_t;\ \boldsymbol{\gamma}_{t-1} = \boldsymbol{\gamma}_t$;
    for $i \in$ randperm$(n)$ do
      – Compute sub-gradients
      $\mathbf{g}_{\mathbf{w}}^t = \begin{cases} \frac{\lambda\mathbf{w}}{n} + \frac{\mathbf{x}_i}{n_+}, & \text{if } y_i = 1 \ \& \ y_i(\mathbf{w}_t^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}_t^\top|\mathbf{x}_i - \mathbf{c}| - 1) \geq 0 \\ \frac{\lambda\mathbf{w}}{n} - \frac{\mathbf{x}_i}{n_-}, & \text{if } y_i = -1 \ \& \ y_i(\mathbf{w}_t^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}_t^\top|\mathbf{x}_i - \mathbf{c}| - 1 - \rho_t) \geq 0 \\ \frac{\lambda\mathbf{w}}{n}, & \text{otherwise.} \end{cases}$
      $\mathbf{g}_{\boldsymbol{\gamma}}^t = \begin{cases} \frac{\mathbf{x}_i}{n_+} - \frac{\mathbf{s}}{n}, & \text{if } y_i = 1 \ \& \ y_i(\mathbf{w}_t^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}_t^\top|\mathbf{x}_i - \mathbf{c}| - 1) \geq 0 \\ -\frac{\mathbf{x}_i}{n_-} - \frac{\mathbf{s}}{n}, & \text{if } y_i = -1 \ \& \ y_i(\mathbf{w}_t^\top(\mathbf{x}_i - \mathbf{c}) + \boldsymbol{\gamma}_t^\top|\mathbf{x}_i - \mathbf{c}| - 1 - \rho_t) \geq 0 \\ -\frac{\mathbf{s}}{n}, & \text{otherwise.} \end{cases}$
      – Update polyhedral cone parameters
      $\mathbf{w}_t \leftarrow \mathbf{w}_t - \alpha_t\mathbf{g}_{\mathbf{w}}^t$
      $\boldsymbol{\gamma}_t \leftarrow \boldsymbol{\gamma}_t - \alpha_t\mathbf{g}_{\boldsymbol{\gamma}}^t$
    end for
    if $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \epsilon_{\mathbf{w}} \ \& \ \|\boldsymbol{\gamma}_t - \boldsymbol{\gamma}_{t-1}\| < \epsilon_{\boldsymbol{\gamma}}$, break
  end for
```

when constructing the extended vector, i.e., $\tilde{\mathbf{x}} \equiv \binom{\mathbf{x}-\mathbf{c}}{||\mathbf{x}-\mathbf{c}||} \in \mathbb{R}^{d+1}$, where $||.||$ denotes the L_2 (the Euclidean) norm of the vector, we obtain a classifier that returns ellipsoidal decision regions with the constraint that the ellipsoid is a sphere that has been elongated in a single direction (i.e., the shape matrix of the ellipse is the identity matrix plus a rank one update). The center of the ellipse is also shifted from $\mathbf{c}$ by an amount related to $\tilde{\mathbf{w}}$. Similarly we can use squares of distances in EPCC to create ellipsoidal decision regions, i.e. we set $\tilde{\mathbf{x}} \equiv \binom{\mathbf{x}-\mathbf{c}}{\mathbf{x}_e} \in \mathbb{R}^{2d}$, where $\mathbf{x}_e = [x_{ei}] \in \mathbb{R}^d,\ x_{ei} = (x_i - c_i)^2$.

2.3 Estimating Optimum Cone Vertex Point

Here, we introduce a method to estimate the best cone vertex position and classifier parameters optimally. To this end, we first write the cone vertex point, $\mathbf{c}$, as linear combination of positive class samples, i.e., $\mathbf{c} = \mathbf{X}_{pos}\boldsymbol{\alpha}$, where $\mathbf{X}_{pos}$ is the matrix whose columns are the positive class samples and $\boldsymbol{\alpha}$ is the unknown vector of linear combination coefficients. In this case, EPCF given in (3) can be written as

$$f_{\mathbf{w},\boldsymbol{\gamma},\boldsymbol{\alpha},b}(\mathbf{x}) = \mathbf{w}^\top(\mathbf{x} - \mathbf{X}_{pos}\boldsymbol{\alpha}) + \boldsymbol{\gamma}^\top|\mathbf{x} - \mathbf{X}_{pos}\boldsymbol{\alpha}| - b. \quad (6)$$

To find the best cone vertex point and the EPCC classifier parameters simultaneously, we need to solve the following optimization problem

$$\begin{aligned} \underset{\mathbf{w},\boldsymbol{\gamma},\boldsymbol{\alpha},b}{\arg\min} \quad & \tfrac{1}{2}\mathbf{w}^\top\mathbf{w} + \tfrac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{\alpha} + C_+\textstyle\sum_i \xi_i + C_-\textstyle\sum_j \xi_j \\ \text{s.t.} \quad & \mathbf{w}^\top(\mathbf{x}_i - \mathbf{X}_{pos}\boldsymbol{\alpha}) + \boldsymbol{\gamma}^\top|\mathbf{x}_i - \mathbf{X}_{pos}\boldsymbol{\alpha}| - b - \xi_i \leq -1, \; i \in I_+, \\ & \mathbf{w}^\top(\mathbf{x}_j - \mathbf{X}_{pos}\boldsymbol{\alpha}) + \boldsymbol{\gamma}^\top|\mathbf{x}_j - \mathbf{X}_{pos}\boldsymbol{\alpha}| - b + \xi_j \geq +1, \; j \in I_-, \\ & \xi_i, \, \xi_j \geq 0. \end{aligned} \quad (7)$$

The same optimization problem can also be used for PCC by setting γ to a scalar value. This optimization problem differs from (4) in the way that there are additional unknown $\boldsymbol{\alpha}$ parameters and the objective function includes a term to regularize these parameters. This optimization problem is non-convex with respect to all parameters, but it becomes convex when we fix some of them. Therefore, we used Alternating Optimization to solve this problem. To this end, we first set the cone vertex to an initial value (e.g., positive class mean), and find the EPCC classifier parameters, $(\mathbf{w}, \boldsymbol{\gamma}, b)$ using linear SVM algorithm. Then, we fix the classifier parameters, and take the derivative of the objective function with respect to $\boldsymbol{\alpha}$, and find the best parameters of $\boldsymbol{\alpha}$ using SG algorithm. Next, the new classifier parameters are found by using new calculated cone vertex position. This procedure is iteratively repeated until there is no big change in parameters. The proposed classifier converges to the optimal cone vertex point in 5-6 iterations. Section 3.7 reports some experimental results on both synthetic and real datasets obtained by using the proposed method. It should be noted that this classifier becomes too slow compared to classical EPCC since we have to solve two large optimization problems in each iteration. Therefore, we did not attempt to estimate the best cone vertex position in the rest of the experiments and we set it to the positive class mean.

2.4 Deep Neural Networks Using Polyhedral Conic Classification

We integrated the proposed polyhedral conic classifier into deep neural networks to learn features and classifier simultaneously [4]. To this end, we used the recent state-of-the-art ResNet-101 architecture. It should be noted that using a single label for an image is generally not appropriate for real-world applications, as the majority of the images can be associated with multiple labels to describe its semantic contents, such as objects, scenes, actions and attributes. Therefore, we formulated the proposed polyhedral conic function loss to handle both multi-class and multi-label classification problems.

Now, let $\tilde{\mathbf{x}}_i \equiv \begin{pmatrix} \mathbf{x}_i - \mathbf{c}_{\mathbf{y}_i} \\ |\mathbf{x}_i - \mathbf{c}_{\mathbf{y}_i}| \end{pmatrix} \in \mathbb{R}^{2d}$, $(i = 1, \ldots, n)$ is the augmented feature vector in the last layer of the network just before the classification layer, and $\mathbf{y}_i \in \{-1, 1\}^m$ is the corresponding label vector with -1 indicating a negative and 1 indicating a positive class. There are m different classes to be classified. For single label case, the label vector $\mathbf{y}$ includes a single 1 term and it may include several 1 terms when an image includes multiple labels.

Assuming that $C^+_{\mathbf{x}_i}$ and $C^-_{\mathbf{x}_i}$ denote the sets of indices with positive and negative labels for $\tilde{\mathbf{x}}_i$, our proposed classifier solves the following optimization problem

$$\min_{\tilde{\mathbf{W}}} \; \sum_{i=1}^{n} \sum_{j=1}^{C^+_{\mathbf{x}_i}} \sum_{k=1}^{C^-_{\mathbf{x}_i}} H(\tilde{\mathbf{w}}_j^\top \tilde{\mathbf{x}}_i - \tilde{\mathbf{w}}_k^\top \tilde{\mathbf{x}}_i), \quad (8)$$

where $H(t) = \max(0, 1 - t)$ is the classical hinge loss, and $\tilde{\mathbf{w}}_j$ corresponds to the j-th column of $2d \times m$ weight matrix $\tilde{\mathbf{W}}$. It should be noted that we omitted the regularization loss term, trace$(\tilde{\mathbf{W}}^\top\tilde{\mathbf{W}})$, since it is already implemented in deep neural networks via weight decay parameter.

The cone vertex $\mathbf{c}_{y_i}$ must be updated in each iteration since the feature representations of samples also change. Updating the cone vertices in each batch using entire training data is impractical, thus we update them in each batch by using the examples of the existing samples in that batch. For binary classification problems, we set the cone vertex to the mean of the positive class samples. For multi-class (label) classification problems, ideally we should compute the cone vertex for each class separately. But, this brings additional computations during testing since the class vertex of each class must be subtracted from the test examples to create augmented test feature vectors separately. To avoid this, we used a single common cone vertex for all classes, and it is simply set to averages of all class samples in the current batch.

3 Experiments

We tested the proposed polyhedral conic classifiers[5] on both synthetic and real datasets for object detection, face verification, open set recognition and classical closed-set multi-class discrimination. We also investigated whether an improvement can be obtained by using optimized cone vertex point described at section 2.3. We compare our results with several other linear and quasi-linear methods like SVM, Kernel SVM using 2nd order polynomial function, 1-Sided Best Fitting Hyperplane Classifier (1S-BFHC) [16], GEPSVM [5], one-class SVM (SVDD) [10], Additive Kernels method [18], Convex Polytope Machine (CPM) [25]. For open set recognition problems, we also compared the proposed methods to 1-vs-Set Machine method of [6]. We could not test against the polyhedral classifier of [20] as this software is not available. For deep neural networks, we compared the proposed deep neural network using polyhedral conic loss function to the one using softmax loss function.

We emphasize out that our polyhedral classifiers are best viewed as drop-in replacements for *linear* SVM, which our classifiers systematically outperform in the tests below regardless of the application and the features used, with only modest increases in memory usage and run time comparing to SVM. Kernel SVMs and similar instance-based methods will typically have sometimes better absolute accuracy but they are usually too slow for practical use in applications of these kinds, except perhaps as the final stages of classifier cascades with faster early stages such as our methods. This applies to training too: in the face detection study below the final training set size is about 250K and kernel SVM algorithms like Sequential Minimal Optimization [36] struggle to handle datasets of this scale. For this reason it was not practical to

4. The software for deep neural network classifier can be reached at https://github.com/hsaglamlar/Deep_EPCC.

5. Our code is available at http://mlcv.ogu.edu.tr/softwarepcc.html

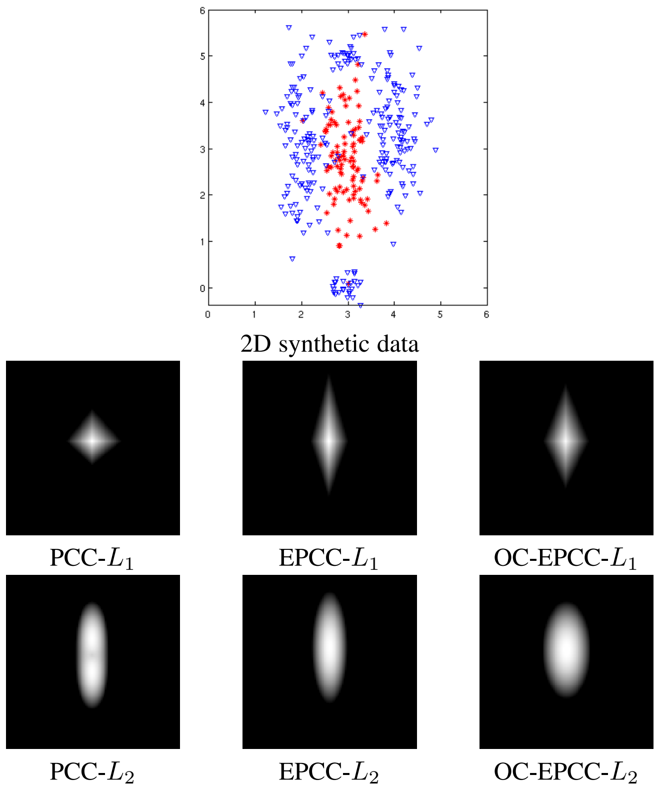


Fig. 4. 2D synthetic data and the decision boundaries returned by the proposed classifiers. Brighter pixels in decision boundaries correspond to higher scores.

include results for kernelized methods in the object detection tests. But, we tested kernel SVMs in other experiments. In addition, we also tested Additive Kernels method [18] that approximates the kernelized methods. We report classification rates or PASCAL VOC style Average Precision (AP) scores [37] for performance assessment. For multi-class problems we used the one-against-rest (OAR) approach as this worked best for all methods.

3.1 Illustrations on Synthetic Data

Fig. 4 illustrates the proposed conic classifiers on a synthetic 2D dataset consisting of random points with the positive class being Gaussian with mean $\binom{3}{3}$ and axis-aligned standard deviation $\binom{0.1}{0.9}$, while the negative class is a mixture of Gaussians with the same standard deviation and several means surrounding the positive one. Quantitatively, Table 1 gives empirical Average Precisions for a 250 positive / 750 negative test set sampled from these distributions. The PCC method using L_2 norm achieves the best result after the statistically-optimal Bayes classifier. One-class EPCC (OC-EPCC) and one-class PCC (OC-PCC) also do very well even though the versions tested here were trained using positive samples alone. Linear SVM fares poorly because the problem is not linearly separable. An Additive Kernel method that explicitly maps the data to an 18-D feature space does better, but not as well as our methods which only use 3 or 4 dimensional embeddings.

TABLE 1
Average Precision (%) on the Synthetic Dataset.

Method	**AP Scores**
Bayes Classifier	**90.89**
PCC-L_2	**90.64**
EPCC-L_2	88.85
EPCC-L_1	86.62
OC-EPCC-L_1	84.87
OC-EPCC-L_2	84.26
PCC-L_1	79.90
Additive Kernels	76.80
SVDD	71.14
GEPSVM	44.25
SVM	22.85

3.2 Object Detection Experiments

3.2.1 *Experiments on Face Detection*

We tested our classifiers on two face detection datasets, 2845 image FDDB (Face Detection Data set and Benchmark) [38], and ESOGU[6], a frontal face detection dataset that includes 667 high-resolution color images with 2042 annotated frontal faces. Both datasets include images that contain faces appearing at a wide range of image positions and scales, and also complex backgrounds, occlusions and illumination variations.

Given the limitations of current publicly-available face detector training sets, we collected 20K sub-images of frontal upright faces from the web for training. The faces were rescaled and cropped to a resolution of 35×28. For the negative set we randomly sampled 10K windows from face-free regions of the images with complex backgrounds. The sub-images were rescaled and cropped to size 35×28 then represented as 620-D LBP+HOG feature vectors.

To allow a direct comparison of methods we trained several sliding window face detectors that were identical except for the (quasi-)linear classifiers used, testing the proposed PCC and EPCC methods, 1S-BFHC hyperplane-fitting classifier, linear SVM, and Additive Kernels. We used spectral clustering to partition the positive samples into three groups and trained a single classifier for each partition. Each initial classifiers were used to scan a set of thousands of images to collect additional hard negatives. Then the classifiers were retrained to create the final detector. The final size of the training set is around 250K. The standard sliding window approach of [39] was used for testing, stepping the detector window by 3 pixels horizontally, 4 vertically, and 1.15 in scale and using greedy non-maximum suppression.

The PASCAL VOC metrics were used to assess accuracy: we report Average Precision (AP), i.e., area under the precision-recall curve. Table 2 gives Average Precision scores for tested classifiers on the FDDB and ESOGU datasets. It also gives the corresponding scores for three publicly-available detectors: the boosted frontal face detector of Kalal *et al.* [40], the short cascade of Cevikalp & Triggs [11], and the OpenCV Viola-Jones detector [41]. The scores of the latter detectors are not strictly comparable because they used different non-publicly-available training sets and multi-stage cascades with nonlinear final stages whereas our detectors used only a single linear stage. For FDDB dataset, our proposed

6. http://mlcvdb.ogu.edu.tr/facedetection.html

TABLE 2
Average Precision (%) for various face detectors on the FDDB and ESOGU Faces datasets.

Method	FDDB	ESOGU
EPCC-L_1	**71.86**	**89.11**
EPCC-L_2	65.33	72.40
PCC-L_1	67.17	78.79
PCC-L_2	54.25	65.30
SVM	37.60	47.66
Additive Kernels	55.70	78.70
1S-BFHC	70.5	80.0
Cevikalp-Triggs [11]	**74.1**	87.4
Kalal *et al*. [40]	66.3	79.7
Viola-Jones [41]	67.6	76.2

EPCC-L_1 method achieved the second best result after cascade detector of [11], whereas the proposed EPCC-L_1 achieved the best result on ESOGU dataset. These results are very promising since they compare favorably to the more complex nonlinear classifiers and they are obtained by using linear (non-kernelized) form of the proposed classifiers. The PCC methods usually produce lower results compared to EPCC. 1S-BFHC achieves the second best result after EPCC-L_1 for both datasets among our trained classifiers. Using L_2 norm with the proposed classifiers significantly reduces the accuracy. SVM classifier remarkably fails and gives the worst performance, because there are many false positives that fall to the positive side of the separating hyperplane far from the positive samples as illustrated in Fig. 1. These false positives have high scores, thus they significantly decrease the accuracy. Using Additive Kernels to provide nonlinear decision boundaries is a significant improvement over linear SVM, but its accuracy remains lower than the proposed methods and 1S-BFHC, suggesting that it does not manage to constrain the positive region as well as they do.

3.2.2 Experiments on Pedestrian Detection

We trained and tested an analogous series of detectors on the INRIA Person dataset [42], again testing linear EPCC, PCC, 1S-BFHC, SVM and Additive Kernels with identical settings for each. We used Felzenszwalb *et al*. [39] latent training methodology, training one symmetric pair of roots without parts. The roots were initialized by applying *k*-means clustering to mirror-image pairs. We used HOG features as in [39]: 8×8 pixel cells with window steps of 8 pixels and pyramid scales spaced by a factor of 1.07. For comparison we cite the published results of Felzenszwalb *et al*. [39] (linear latent SVM over HOG, using one symmetric pair of roots, each with 8 parts – 18 filters in total, and bounding box prediction), Hussain & Triggs [43] (a two stage, linear then quadratic cascade based on single root latent SVM over HOG+LBP+LTP), and Dalal & Triggs [42] (simple linear SVM over HOG without latency, multiple roots or parts).

The resulting accuracies and testing times per image are presented in Table 3. The EPCC-L_1 detector achieves the best results among those trained. Without having parts modeling, it does not quite match the score of the Felzenszwalb multi-root, multi-part detector. However, it does outperform the Hussain & Triggs [43] method despite the Hussain & Triggs's better features and two stages. EPCC-L_2, PCC-L_2, and PCC-L_1 also perform well here. As opposed to the face detection results, there is not a significant difference between accuracies obtained by using L_1 and L_2 norms. Note that despite their gains in accuracy, the run times for EPCC and PCC are very similar to those for SVM (and half of those for [39]), so EPCC is a promising drop-in replacement for linear SVM here. In contrast to the face detection results, Additive Kernels provides little improvement in accuracy over linear SVM even though it is the slowest method tested.

TABLE 3
Average Precision (%) on the INRIA Person dataset.

Method	AP Score (%)	Run Time (s)
EPCC-L_1	**85.6**	1.8
EPCC-L_2	85.0	1.6
PCC-L_1	83.6	1.8
PCC-L_2	84.3	1.5
SVM	80.4	1.6
Additive Kernels	80.9	−−
1S-BFHC	78.5	1.6
Felzenszwalb [39]	**86.9**	3.5
Hussain-Triggs [43]	84.1	–
Dalal-Triggs [42]	75.0	–

TABLE 4
Object Detection Results (%) on MS COCO dataset.

Method	mAP@0.5	mAP@[.5, .95]
EPCC	**37.2**	**18.4**
Fast R-CNN [44] (SVM)	35.1	17.8

3.2.3 Experiments on MS COCO

We experimented on Microsoft COCO [46] object detection dataset released in 2014. MS COCO detection dataset includes 80 object categories with approximately 80k images in the training set and 40k images in the validation set. We evaluate the mAP averaged for IoU (Intersection over Union) $\in [0.5 : 0.05 : 0.95]$ (this is COCO's standard metric denoted by mAP@[.5, .95]) and mAP@0.5 (PASCAL VOC's metric). To test the proposed classifier, we used a multi-stage training approach where a deep neural network is trained to obtain CNN features in the first stage and this is followed by the EPCC training on the extracted features.

We used Fast R-CNN [44] implementation with the setup given in [47] which uses Region Proposals provided by [48]. Region proposals are selective search proposals around 2000 regions per image. The deep neural network is a VGG-16 architecture fine-tuned from a network pre-trained on ImageNet dataset. We used the default parameters for fine-tuning which used a mini batch size of 64, 240K iterations with a learning rate of 0.001 and then for 80K iterations with 0.0001. In the second stage, we extracted FC7 layer's output as CNN features and then trained EPCC classifier. We set the maximum number of negative samples to 90K per class due to memory limitations during EPCC training. In a similar manner, we also trained SVM classifier for comparison as in Fast R-CNN [44]. The results are given in Table 4. The proposed EPCC classifier improves the accuracy around 2% for mAP@0.5 and 0.6% for mAP@[.5, .95] over linear SVM. It should be noted that our Fast R-CNN implementation using SVM achieves lower accuracies compared to the ones given in [47]. The authors of [47]

TABLE 5
Results for PaSC Face Verification Experiments.

	One-Against-One Regime				One-Against-Rest Regime			
	Verification Rates (%)		mAP Scores (%)		Verification Rates (%)		mAP Scores (%)	
Methods	**PaSC control**	**PaSC handheld**	**PaSC control**	**PaSC handheld**	**PaSC control**	**PaSC handheld**	**PaSC control**	**PaSC handheld**
EPCC-L_1	88.94	75.20	**71.00**	57.15	91.47	81.99	68.07	64.47
EPCC-L_2	88.54	74.46	70.22	56.59	**91.63**	**82.13**	**73.82**	**64.49**
PCC-L_1	**89.33**	74.97	70.53	**57.30**	90.59	79.20	72.58	61.97
PCC-L_2	86.17	75.58	67.14	56.42	91.25	80.87	72.97	63.15
SVM	86.68	74.57	69.73	56.98	78.06	71.48	58.26	51.04
Additive Kernels	86.07	70.80	67.96	53.59	88.76	76.34	64.71	59.75
CERML-EG [45]	80.11	**77.37**	–	–	–	–	–	–

use different network parameters during fine-tuning stage since they use 8-GPUs. In contrast, we used the default parameters recommended for a single GPU. The accuracy difference may be due to this factor.

3.3 Experiments on Face Verification

For face verification experiments, we used Point-and-Shoot Face Recognition Challenge (PaSC) dataset [49]. The PaSC dataset includes 2802 videos of 265 people carrying out simple actions. Videos are recorded under two different settings. In our experiments, we used deep CNN features of face images provided by [45]. On PaSC, there are two video face verification experiments: control-to-control and handheld-to-handheld experiments. In both experiments, the target and query sets contain the same set of videos. The task is to verify a claimed identity in the query video by comparing with the associated target video. Since the same 1401 videos served as both the target and query sets, "same video" comparisons are excluded as in [45], and our results are directly comparable to the ones reported in [45] since we use the same CNN features and test protocols.

To test methods, we follow the same testing setup as used in [45]: we first compute the similarities between pair-wise face videos and create a similarity matrix. Then, this matrix is used to create ROC curves and we report the verification rate when false accept rate is 0.01. In addition, we also report the average precision (mAP) scores obtained from Precision-Recall curves. For computing similarity scores, we used two different settings to emphasize the importance of polyhedral separation. In the first setting, we just trained a binary classifier using images of two videos coming from target and query sets, and used the reciprocal of returned margin, $||\tilde{\mathbf{w}}||$, as similarity score between these two video sets. It should be noted that the classes can be easily separated by a linear hyperplane in this setting since pair-wise separations are considered. This setting is called as "one-against-one setting (OAO)" since it is similar to the "one-against-one regime" which is used to extend binary SVM classifiers to multi-class classification. For the second setting, we combine all images in the query set and treat it as negative class and each video in the target set is considered as positive class. Then, we train a binary classifier separating these two classes. The similarity matrix is computed by finding the closest 10 samples from each video in the query set to the separating hyperplane and taking their mean as final distance. This setting is called as "one-against-rest setting (OAR)". In contrast to the first setting, the classes are imbalanced here and it is much harder to separate the positive and combined negative classes with a linear hyperplane anymore. So, we need polyhedral acceptance regions for better performance.

The results are reported in Table 5 and they support our claim that polyhedral acceptance regions are better suited for "one-against-rest setting". More precisely, the proposed EPCC classifier using L_2 norm achieves the best accuracies in "one-against-rest setting", and these accuracies are significantly higher than the results of [45], 80.11% in control set and 77.37% in handheld set. Moreover our best result on handheld dataset also significantly outperforms the recent state-of-the-art result, 80.33%, of [50], and our best result on control dataset is slightly behind the accuracy, 92.06%, in [50]. Combining all video images in "one-against-rest setting" improves the accuracies of the proposed methods compared to the ones obtained for "one-against-one setting". It is because using more negative data helps to return more compact polyhedral acceptance regions for positive classes since the negative class samples surround the positive class regions. In contrast, in "one-against-one setting", the samples of positive and negative classes are balanced and the proposed classifiers return looser acceptance regions, which yield to lower accuracies. As opposed to these results, linear SVM yields higher accuracies for OAO setting since the pair-wise classes are linearly separable whereas the accuracies significantly drop for OAR setting since the classes are no longer separable by a linear hyperplane in this setting.

3.4 Visual Object Classification Experiments

3.4.1 *Experiments on PASCAL VOC 2007 Dataset*

We ran tests on the PASCAL 2007 Visual Object Classification dataset using a popular Convolutional Neural Net feature set. We ran the pre-trained ILSVRC2012 Caffe implementation [51] of the Krizhevsky *et al.* [52] AlexNet CNN on images resized to 256×256, producing 4096-dimensional feature vectors for each of the methods shown. For comparability with the literature and to see the performance differences between tested classifiers better, we used stock ILSVRC features without fine-tuning them on the PASCAL dataset. The results are given in Table 6, as PASCAL VOC Average Precision scores. The proposed methods achieve the best accuracies along with Additive Kernels and KSVM for most of the classes. The best performer is OC-EPCC, trained with samples of both positive and negative classes. It significantly outperforms a linear SVM over the same features, gaining about 4% on average and more than 5% on the classes bottle, bus, chair, dining table, dog, potted-plant, sofa and tv monitor. Additive

TABLE 6
Average Precision scores (%) on PASCAL VOC 2007 classification datasets.

Methods	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	TV Monitor	Average
OC-EPCC-L_1	85.1	79.7	82.9	81.3	36.4	**69.5**	83.2	80.7	**57.7**	61.6	**70.0**	**79.9**	**83.2**	74.0	**90.4**	51.0	73.4	58.6	84.5	66.7	**72.5**
EPCC-L_1	**87.2**	**80.0**	**83.3**	80.9	35.9	66.5	**83.4**	80.9	56.5	59.4	68.7	78.5	82.6	73.8	90.1	49.7	71.3	57.1	86.5	66.6	72.0
PCC-L_1	86.3	79.0	83.0	80.5	35.3	65.8	**83.4**	80.2	56.1	60.3	68.0	77.2	81.8	73.3	89.8	47.9	70.8	55.6	85.9	66.4	71.3
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	68.6
KSVM	83.9	77.3	82.2	**81.8**	**38.7**	**69.5**	81.9	79.6	57.5	60.2	69.8	79.2	79.1	71.2	89.0	**52.6**	**73.8**	**59.3**	84.8	**69.7**	72.1
Additive Kernels	86.6	78.5	83.0	81.2	35.6	68.0	82.0	**81.5**	51.0	**63.1**	65.5	76.2	82.7	**74.9**	88.7	47.3	72.7	54.0	**86.7**	64.2	71.2
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	66.8
GEPSVM	36.2	21.9	45.1	26.4	10.3	27.0	34.1	21.9	29.0	39.9	32.0	22.2	32.0	19.6	53.9	15.4	27.2	14.3	39.0	25.8	28.7
SVDD	65.5	32.4	25.0	26.0	21.5	31.2	37.1	48.7	28.3	23.1	17.7	25.5	39.3	31.8	58.8	12.3	21.2	18.5	59.2	25.5	32.4
Deep EPCC	**96.2**	**91.3**	**91.2**	**89.8**	**63.6**	**84.5**	**89.4**	**91.1**	**68.5**	**84.7**	**81.3**	**90.6**	**90.6**	**89.0**	92.9	**69.7**	**84.2**	**73.7**	91.6	81.4	**84.8**
Deep CNN (softmax)	93.6	89.7	90.9	88.9	60.1	83.1	88.5	91.0	68.4	84.1	79.4	90.1	90.2	88.9	**94.1**	66.6	83.6	73.6	**91.7**	81.0	83.9

Kernels improves results over linear SVM, but it uses a three-times larger feature space. GEPSVM was the worst performer here.

In addition to these experiments, we also tested the proposed ResNet-101 deep CNN classifier (Deep EPCC) using the proposed EPCC loss function and compared it to the same network using softmax loss function. It should be noted that both deep CNN classifiers significantly improve the accuracies over the classifiers using AlexNet CNN features. Moreover, the proposed Deep EPCC classifier beats the one with softmax for 18 classes out of all 20 classes, and it brings around 1% improvement on the average.

3.4.2 Experiments on CIFAR-10 Dataset

Here we use CIFAR-10 dataset images[7]. The CIFAR-10 dataset consists of 60K, 32×32 color images of 10 classes, with 6K images per class. There are 50K training and 10K test samples. We first extracted CNN features for CIFAR-10 dataset images by using the AlexNet as in PASCAL VOC experiments. Then we trained the proposed deep CNN networks using ResNet-101 architecture. The results are given in Table 7. As seen in the table, the proposed Deep EPCC classifier achieves a very high accuracy of 96.68%. It should be noted that this accuracy is very close to the recent state-of-the-art accuracy of 97.28% reported in [53] and it slightly beats the previous best result of 96.53% in [54]. Among the classification methods using the stock CNN features, KSVM achieves the highest accuracy followed by the proposed EPCC-L_1 method. GEPSVM is again the worst performer.

TABLE 7
Classification Rates (%) on the CIFAR-10 dataset.

Method	**AP Score (%)**
EPCC-L_1	75.79
EPCC-L_2	75.22
PCC-L_1	75.62
PCC-L_2	72.28
SVM	71.71
KSVM	**76.45**
Additive Kernels	75.47
1S-BFHC	71.30
GEPSVM	18.23
Deep EPCC	**96.68**
Deep CNN (softmax)	95.00

3.4.3 Experiments on Caltech-256 & CIFAR-100 Datasets

Here we test classifiers on two multi-class visual object classification datasets: Caltech-256 and CIFAR-100 data sets. For Caltech-256, we follow the standard procedure: pick 60 images from each class and split them into 30 for training and 30 for test, then, reverse the role of training and test. Fisher vector (FV) representation is used to represent images and we used the same setup as in [55]. 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 (PCA). 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. For CIFAR-100 dataset, we used 4096-dimensional fine-tuned Convolutional Neural Net (CNN) features. Training data has 50K samples while test data has 10K samples.

The results are summarized in Table 8, in terms of simple classification accuracies. The Additive Kernels gave the best accuracy on Caltech-256. However, the proposed EPCC classifiers and KSVM using 2nd order polynomial kernel function achieved the best accuracies for CIFAR-100 dataset. However, note that Additive Kernels used significantly longer feature vectors than EPCC: 3 times the original input space dimension for Caltech-256, and 5 times for CIFAR-100 dataset. In a similar manner the dimensionality of the new sample space is $\binom{d+2-1}{2}$ when a 2nd order polynomial kernel function is used. CPM achieves the second best accuracy for Caltech-256. Although proposed methods were beaten by Additive Kernels and CPM on Caltech-256

7. Available at https://www.cs.toronto.edu/ kriz/cifar.html

TABLE 8
Classification rates (%) for the multi-class visual object classification experiments.

Methods	**Caltech-256**	**CIFAR-100**
EPCC-L_1	40.1 $\pm$0.6	**86.4**
EPCC-L_2	40.0 $\pm$0.7	**86.4**
PCC-L_1	40.4 $\pm$0.7	86.2
PCC-L_2	38.4 $\pm$0.4	85.4
SVM	37.6 $\pm$0.7	85.8
KSVM	38.8 $\pm$0.7	**86.4**
GEPSVM	13.3$\pm$0.6	74.5
1S-BFHC	38.3 $\pm$1.0	85.6
SVDD	9.9 $\pm$0.2	48.8
Additive Kernels	**42.6** $\pm$0.7	73.3
CPM [25]	41.7 $\pm$3.7	65.9

dataset, they did significantly outperform the remaining (quasi-) linear classifiers that were tested. Note that for Caltech-256, PCC-L_1 significantly outperforms SVM even though it has just one additional feature (of 164k). This shows that it is the positive-class-bounding polyhedral cone geometry that is providing the improvement here, not the features used, and also that our training methods can gracefully handle very large feature vectors.

TABLE 9
Accuracies on the MS-COCO dataset on 80 classes for top-3 highest ranked labels.

Method	**P-C**	**R-C**	**F1-C**
Deep EPCC	60.6	**60.2**	59.0
WARP [56]	59.3	52.5	55.7
CNN+RNN [57]	**66.0**	55.6	**60.4**

3.4.4 *Experiments on Multi-Label Classification*

We have used MS-COCO dataset for multi-label classification. Images in this dataset may have more than one category label, and there are approximately 2.95 object labels per image. To assess the performance, we used the most common precision, recall and F1 measures computed based on the top-3 highest ranked labels as in [56]. Here we only tested the proposed multi-label deep neural network classifier and compared it to the best published results on MS-COCO. The results are given in Table 9. As can be seen in the table, the proposed Deep EPCC method achieves comparable results compared to the most complex methods even though it uses a simple loss function given in (8). For example WARP [56] learns a weight function for each example and CNN+RNN [57] uses two different deep neural network architectures for multi-label classification. Yet, the proposed method significantly outperforms WARP and it is slightly behind CNN+RNN in terms of F1-C scores.

3.5 Experiments on UCI Repository Datasets

We tested the proposed methods on 7 binary and multi-class datasets chosen from UCI repository: Ionosphere, Iris, Letter Recognition (LR), Multiple Features (MF) - pixel averages, Pima Indian Diabetes (PID), Wine, and Wisconsin Diagnostic Breast Cancer (WDBC) datasets. The sizes of the UCI problems are summarized in Table 10. We used 10-fold cross-validation to evaluate the performance. The accuracies are given in Table 11. Since the software of SPLA1 [23], SPLA2 [23], and Polyceptron [24] methods are not available, we report only the published accuracies on some of the datasets tested in those studies. EPCC, PCC and KSVM methods beat other methods by winning 3 out of 7 datasets. The proposed methods typically achieve the best or the second best results for all tested datasets. The proposed classifiers significantly outperform linear SVM especially on Ionosphere and LR datasets (e.g., for LR dataset, the accuracy of EPCC-L_1 is approximately 20% better than the accuracy of SVM which achieves the best performance among other linear rival classifiers). The Polyceptron method of [24] wins for WDBC, while GEPSVM and SVDD methods usually give the worst classification accuracies.

TABLE 10
Datasets from the UCI Repository

Dataset	**# Classes**	**# Examples**	**Dimension**
Ionosphere	2	351	34
Iris	3	150	4
LR	26	20000	16
MF	10	2000	240
PID	2	768	8
Wine	3	178	13
WDBC	2	569	30

3.6 Experiments on Open Set Recognition

3.6.1 *Open Set CIFAR-10 Dataset Recognition*

Here, we used CIFAR-10 dataset to create open set recognition setup. For this setup, we randomly select 3 classes from training data and train the classifiers on the training samples from these classes alone. In contrast, testing uses samples from all 10 classes. We compute AP scores from the Precision-Recall curves for the 3 classes, take the average of these, repeat the whole procedure over 10 trials, and report the final averaged average AP score. The results for CIFAR-10 are summarized in Table 12. As seen from the results, the proposed PCC-L_1 achieves the best accuracy. All PCC/EPCC classifiers significantly outperform all other tested classifiers including KSVM and Additive Kernels. 1-vs-Set Machine method slightly beats linear SVM. GEPSVM and SVDD are the worst performing methods.

3.6.2 *Open Set USPS Digit Recognition*

Next we made experiments on open set recognition environment based on the USPS Digits dataset that contains 9298 16$\times$16 gray-scale images of hand-written digits, with 7291 for training and validation and the remaining 2007 for testing. To make the problem harder we use the raw gray-scale pixel values as features without any pre-processing or feature extraction. As in the previous experiment, for open set recognition we randomly choose three classes and train the methods on the training samples from these classes alone. We use samples from all 10 classes during testing. We compute AP scores as in the previous experiment. We also compared test times of the methods for this dataset. The results are presented in Table 13. The OC-EPCC classifier achieves the best accuracies, followed by PCC-L_2, EPCC-L_1/L_2, PCC-L_1 and KSVM. All of the proposed methods significantly outperform

TABLE 11
Classification rates (%) for the UCI repository datasets.

Methods	Ionosphere	Iris	WDBC	PIMA	Wine	MF	LR
EPCC-L_1	92.0 ±5.5	**98.0** ±3.2	98.0 ±1.6	76.5 ±3.2	97.5 ±3.2	97.2 ±1.3	79.6 ±1.4
EPCC-L_2	92.0 ±5.5	**98.0** ±3.2	97.9 ±1.6	76.9 ±4.2	97.5 ±3.2	97.2 ±1.3	79.3 ±1.4
PCC-L_1	**92.3** ±4.4	96.7 ±4.7	97.4 ±2.1	**77.3** ±2.5	98.1 ±3.0	96.8 ±1.3	68.6 ±0.8
PCC-L_2	89.9 ±6.9	96.7 ±4.7	97.2 ±1.9	70.1 ±4.9	95.0 ±5.8	95.2 ±1.4	63.6 ±1.3
SVM	87.3 ±7.0	94.7 ±4.9	97.7 ±2.0	76.8 ±3.1	96.9 ±4.4	93.9 ±1.1	59.8 ±1.7
KSVM	91.7 ±5.8	**98.0** ±4.5	95.1 ±2.0	72.5 ±5.0	96.3 ±5.3	**98.2** ±0.7	**95.3** ±0.7
GEPSVM	74.8 ±5.8	97.3 ±4.7	89.1 ±5.5	74.7 ±4.4	80.7 ±13.7	53.8 ±4.0	30.5 ±1.1
1S-BFHC	86.8 ±7.2	94.0 ±6.6	97.4 ±2.1	76.0 ±5.0	**98.8** ±2.6	93.8 ±1.5	25.3 ±0.8
SVDD	79.4 ±10.5	91.3 ±7.0	89.8 ±4.8	59.0 ±8.4	90.0 ±8.9	80.1 ±3.5	37.5 ±1.6
Additive Kernels	86.8 ±7.3	96.0 ±4.7	96.3 ±2.3	77.1 ±3.3	96.3 ±4.4	95.1 ±1.0	78.3 ±1.2
CPM [25]	85.2 ±4.9	83.3 ±12.7	91.86 ±2.5	60.7 ±11.3	96.9 ±4.4	96.1 ±1.3	52.9 ±6.0
SPLA1[1] [23]	88.0 ±6.0	-	93.8 ±2.9	76.9 ±0.7	-	-	-
SPLA2[1] [23]	90.6 ±1.2	-	95.8 ±0.4	76.8 ±0.6	-	-	-
Batch Polyceptron[1] [24]	89.7 ±1.3	-	**98.5** ±0.1	-	-	-	-

[1] The results are taken from corresponding article due to lack of software.

TABLE 12
AP scores (%) for the open set CIFAR-10 dataset experiment.

Methods	AP Score (%)
EPCC-L_1	88.85
EPCC-L_2	88.85
PCC-L_1	**90.63**
PCC-L_2	87.90
OC-EPCC_L_1	88.64
OC-EPCC_L_2	88.64
SVM	79.07
KSVM	84.27
Additive Kernels	84.02
CPM [25]	81.05
1-vs-Set Machine	79.11
GEPSVM	36.24
SVDD	35.66

TABLE 13
AP Scores (%) for the open set USPS experiment.

Methods	AP Score (%)	Test Time (s)
OC-EPCC-L_1	**81.98** $\pm$ 13.74	0.44
OC-EPCC-L_2	81.97 $\pm$ 13.70	0.41
EPCC-L_1	79.63 $\pm$ 13.76	0.45
EPCC-L_2	79.63 $\pm$ 13.76	0.48
PCC-L_1	78.26 $\pm$ 13.56	0.23
PCC-L_2	80.21 $\pm$ 11.34	0.24
SVM	69.7 $\pm$ 24.24	0.13
KSVM	76.39 $\pm$ 21.47	6.61
Additive Kernels	72.08 $\pm$ 22.45	1.78
CPM [25]	69.25 $\pm$ 14.24	48.10
1S-BFHC	68.12 $\pm$ 23.15	0.18
1-vs-Set Machine	64.28 $\pm$ 24.09	0.42
GEPSVM	45.51 $\pm$ 25.93	0.01
SVDD	12.54 $\pm$ 8.30	0.01

linear SVM. In terms of testing speed, GEPSVM, SVDD, linear SVM and 1S-BFHC classifiers are the best ones followed by the proposed classifiers. The testing times of the proposed classifiers are comparable to the testing time of linear SVM. More precisely, linear SVM is only 1.4 times faster than PCC classifiers whereas it is approximately 2.5 times faster than EPCC classifiers. KSVM, CPM, and Additive Kernels are the worst performing methods in terms of speed.

3.7 Vertex Point Estimation Experiments

Here we conduct tests on estimating the best cone vertex points by solving the optimization problem (7). To this end, we first test the proposed method on synthetic data to visualize the convergence of the algorithm, and then we perform tests on 7 real binary and multi-class datasets chosen from UCI repository: Iris, Letter Recognition (LR), Multiple Features (MF) - pixel averages, Wine, Glass Identification, Pima Indian Diabetes (PID), and Wisconsin Daignostic Breast Cancer (WDBC) datasets.

For 2D synthetic data, we sampled 250 points uniformly distributed between (2,3) in both directions as positive data. Thus, the positive class region is a square region where the mean point is $\boldsymbol{\mu} = (2.5, 2.5)$. Then we sampled 750 negative samples surrounding the positive class samples as seen in Fig. 5. We initialized the vertex point with $(0, 0)$, and the proposed method successfully converged to the optimal vertex point, $(2.44, 2.46)$ in 6 iterations as seen in Fig. 5.

The results for real data sets using 5-fold cross-validation are summarized in Table 14. Here we compare the EPCC methods using the positive class mean and the optimally estimated vertex point (it is denoted as EPCC-V). For all tests, we initialized the vertex point with positive class mean. As can be seen in the table, the EPCC-V classifier using the optimally estimated vertex points achieves better results than the one using positive class means for some datasets, but the difference is not very significant. Considering the time-consuming optimization of EPCC-V classifier, it is

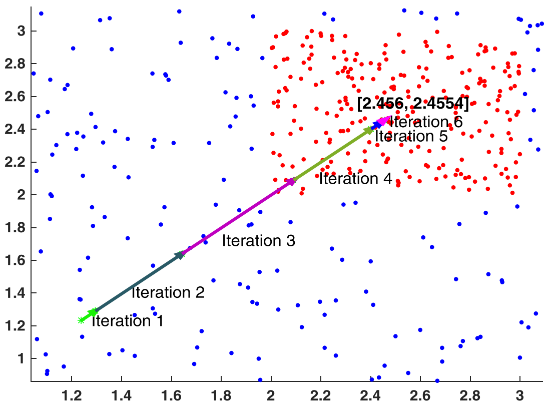


Fig. 5. Visualization of the convergence of the cone vertex point estimation algorithm: Starting from $(0, 0)$, the method converges to the optimal point in 6 iterations.

TABLE 14
Classification Rates (%) on the UCI Datasets.

Method	Iris	Letter	MF	Wine	Glass	WDBC	PIMA
EPCC-L_1-V	**96.67**	79.60	95.30	96.25	**97.62**	**97.91**	76.41
EPPC-L_2-V	96.00	**79.69**	**95.60**	95.63	94.36	97.73	**77.30**
EPCC-L_1	95.99	75.41	95.40	95.63	**97.62**	97.86	76.53
EPCC-L_2	95.99	79.34	**95.60**	96.25	96.19	97.73	76.89

much preferable to set the cone vertex to the mean of all positive training data.

4 Summary and Conclusions

This study argues that in open set object recognition, face verification and sliding window object detection problems, it is advantageous to use asymmetric classifiers that focus on producing compact, well-constrained decision regions for the positive (target object) class. To this end we introduced PCC, EPCC and OC-EPCC, a family of robust scalable maximum margin learning methods whose positive acceptance regions are planar sections through L_1 or L_2 cones. Then, we integrated the polyhedral conic classification loss into the deep neural network classifiers. We also studied finding an optimum cone vertex point and proposed a novel methodology that simultaneously finds the best cone vertex point and classifier parameters. For appropriate parameter settings, the proposed methods give compact, convex acceptance regions that tightly constrain the extent of the positive class. A feature vector augmentation allows PCC and EPCC to be trained using standard linear SVM software, while OC-EPCC is currently trained using an analogous stochastic gradient descent method. We obtained good results on a range of object detection, face verification, open set recognition and classical closed-set discrimination tasks with these methods. The detection and open set recognition results were particularly promising, giving significant improvements across the board against comparable (quasi-)linear classifiers including SVMs and several one-class approaches. Using L_1 norm yielded better results compared to L_2 norm especially for the datasets where the dimensionality is high and the number of samples per class is low compared to the feature size. Overall, we believe that our methods will prove to be useful drop-in replacements for linear discriminants such as SVMs in many current visual object detection and classification tasks. Moreover, the proposed deep neural network classifier using the EPCC loss outperformed the network using softmax loss in all tests. These results comply with the recent studies [27], [28], [29], [30] showing the need for compact class decision boundaries in deep neural network classifiers.

Acknowledgments: This work was funded in part by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number EEEAG-116E080.

References

[1] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
[2] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.
[3] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proc. of Knowledge Discovery and Data Mining*, pages 77–86, 2001.
[4] H. Cevikalp and B. Triggs. Hyperdisk based large margin classifier. *Pattern Recognition*, 46:1523–1531, 2013.
[5] O. L. Mangasarian and E. W. Wild. Multisurface proximal support vcetor machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelliegence*, 28:69–74, 2006.
[6] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1757–1772, 2013.
[7] S. Hussain. *Machine learning methods for visual object detection*. PhD thesis, Laboratoire Jean Kuntzmann, 2011.
[8] A. Satpathy, X. Jiang, and H. L. Eng. Human detection by quadratic classification on subspace of extended histogram of gradients. *IEEE Transactions on Image Processing*, 23:287–297, 2014.
[9] H. Cevikalp and B. Triggs. Polyhedral conic classifiers for visual object detection and classification. In *IEEE Society Conference on Computer Vision and Pattern Recognition*, 2017.
[10] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.
[11] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In *IEEE Society Conference on Computer Vision and Pattern Recognition*, 2012.
[12] H. Cevikalp and B. Triggs. Visual object detection using cascades of binary and one-class classifiers. *International Journal of Computer Vision*, 123:334–349, 2017.
[13] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:2317–2324, 2014.
[14] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult. The extreme value machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:762–768, 2017.
[15] Jayadeva, R. Khemchandani, and S. Chandra. Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelliegence*, 29:905–910, 2007.
[16] 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.

[17] H. Cevikalp. Best fitting hyperplanes for classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1076–1088, 2017.

[18] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:480–492, 2012.

[19] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[20] M. M. Dundar, M. Wolf, S. Lakare, M. Salganicoff, and V. C. Raykar. Polyhedral classifier for target detection: A case study: Colorectal cancer. In *International Conference on Machine Learning*, 2008.

[21] R. N. Gasimov and G. Ozturk. Separation via polyhedral conic functions. *Optimization Methods and Software*, 21:527–540, 2006.

[22] A. M. Bagirov, J. Ugon, D. Webb, G. Ozturk, and R. Kasimbeyli. A novel piece-wise linear classifier based on polyhedral conic and and max-min separabilities. *TOP*, 21:3–24, 2013.

[23] Naresh Manwani and P. S. Sastry. Learning polyhedral classifiers using logistic function. In Masashi Sugiyama and Qiang Yang, editors, *ACML*, volume 13 of *JMLR Proceedings*, pages 17–30. JMLR.org, 2010.

[24] Naresh Manwani and P. S. Sastry. Polyceptron: A polyhedral learning algorithm. *CoRR*, abs/1107.1564, 2011.

[25] Alex Kantchelian, Michael C Tschantz, Ling Huang, Peter L Bartlett, Anthony D Joseph, and J. D. Tygar. Large-margin convex polytope machine. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3248–3256, 2014.

[26] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), September 2010.

[27] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, 2016.

[28] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[29] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: a unified embedding for face recognition and clustering. In *IEEE Society Conference on Computer Vision and Pattern Recognition*, 2015.

[30] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Conference on Learning Representations (ICLR)*, 2015.

[31] K. P. Bennett and E. J. Bredensteiner. Duality and geometry in svm classifiers. In *International Conference on Machine Learning*, 2000.

[32] A. Astorino and M. Gaudioso. Polyhedral separability through successive lp. *Journal of Optimization Theory and Applications*, 112:265–293, 2002.

[33] A. Bagirov. Max-min separability. *Optimization Methods and Software*, 20:271–290, 2005.

[34] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2232, 2009.

[35] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-grdient solver for svm. In *International Conference on Machine Learning*, 2007.

[36] John C. Platt. Fast training of support vector machines using sequential minimal optimization, 1998. Advances in Kernel Methods-Support Vector Learning, Cambridge, MA, MIT Press.

[37] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[38] 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.

[39] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), September 2010.

[40] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *British Machine Vision Conference*, 2008.

[41] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[42] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Society Conference on Computer Vision and Pattern Recognition*, 2005.

[43] S. Hussain and B. Triggs. Feature sets and dimensionality reduction for visual object detection. In *British Machine Vision Conference*, 2010.

[44] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision*, 2015.

[45] Z. Huang, R. Wang, S. Shan, L. Van Gool, and X. Chen. Cross euclidean-to-riemannian metric learning with application to face recognition from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2827–2840, 2018.

[46] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[47] S. Ren, K. He, R. Girschick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Image Processing*, 39:1137–1149, 2017.

[48] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:814–830, 2015.

[49] J. R. Beveridge, P. J. Philiphs, D. S. Bolme, B. A. Draper, G. H. Given, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, and et al. K. W. Bowyer. The challenge of face recognition from digital point-and-shoot cameras. In *BTAS*, 2013.

[50] Y. Rao, J. Lin, J. Lu, and J. Zhou. Learning discriminative aggregations network for video-based face recognition. In *International Conference on Computer Vision*, 2017.

[51] 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.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

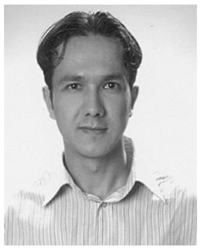
[53] Xavier Gastaldi. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017.

[54] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014.

[55] 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.

[56] Y. Gong, Y. Jia, T.K. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. In *arXiv:1312.4894*, 2014.

[57] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. Cnn-rnn: A unified framework for multi-label image classification. In *IEEE Society Conference on Computer Vision and Pattern Recognition*, 2016.



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 worked as a post-doctoral researcher at Lear Team of INRIA Rhoene-Alpes in France in 2007 and at Rowan University in USA in 2008. 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.

Halil Saglamlar is a doctoral candidate in the Department of Electrical and Electronics Engineering at Eskisehir Osmangazi University, Turkey. He received his B.Sc degree in Department of Electrical and Electronics Engineering at Hacettepe University, Ankara, Turkey (2003) and M.S at Aeronautics and Space Technologies Institute, Istanbul, Turkey (2009). His research interests include machine learning and computer vision.