Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 913 – 928

doi:10.3906/elk-2005-142

The nearest polyhedral convex conic regions for high-dimensional classification

Hakan ÇEVİKALP[1], **Emre ÇİMEN**[2,*], **Gürkan ÖZTÜRK**[2]

[1]Department of Electrical and Electronics Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Turkey

[2]Computational Intelligence and Optimization Laboratory, Department of Industrial Engineering, Faculty of Engineering, Eskişehir Technical University, Eskişehir, Turkey

Received: 21.05.2020 • **Accepted/Published Online:** 24.08.2020 • **Final Version:** 30.03.2021

Abstract: In the nearest-convex-model type classifiers, each class in the training set is approximated with a convex class model, and a test sample is assigned to a class based on the shortest distance from the test sample to these class models. In this paper, we propose new methods for approximating the distances from test samples to the convex regions spanned by training samples of classes. To this end, we approximate each class region with a polyhedral convex conic region by utilizing polyhedral conic functions (PCFs) and its extension, extended PCFs. Then, we derive the necessary formulations for computing the distances from test samples to these new models. We tested the proposed methods on different high-dimensional classification tasks including face, digit, and generic object classification as well as on some lower-dimensional classification problems. The experimental results on different datasets show that the proposed classifiers achieve either the best or comparable results on high-dimensional classification problems compared to other nearest-convex-model classifiers, which shows the superiority of the proposed methods.

Key words: Classification, polyhedral conic region, affine hull, convex hull, convex cone, face recognition

1. Introduction

The nearest subspace-based methods [1, 2] and their generalized extensions using other convex class models [3–7] have been widely used for improving the classification accuracy of the k-nearest neighbor classifiers especially in high-dimensional pattern classification problems. In these methods, each class is approximated with a particular convex class model such as linear/affine subspaces, convex hulls, and convex cones, and then a test sample is assigned to a class based on the shortest distance from the test feature vector to the class models. These nearest-class-model type classifiers significantly improve the classification accuracies of the k-nearest neighbor classifiers that suffer from "hole artifacts" encountered in high-dimensional classification problems with limited number of training samples [4, 8].

Early methods used linear/affine subspaces for approximating classes and very successful classification accuracies were achieved especially for face and hand-written digit recognition problems [1, 9, 10]. In fact, it has been theoretically proved that the set of all images of a Lambertian object as in face images captured under various illumination conditions can be approximated by a low-dimensional linear subspace [11]. Similarly, the coordinate vectors of feature points from a moving rigid object lie in an affine subspace of dimension 3 under affine camera models [12]. More recently, affine hull models are also successfuly used for activity recognition

*Correspondence: ecimen@eskisehir.edu.tr

[13]. Among the subspace methods, class featuring information compression (CLAFIC) [1, 10] (also known as the nearest subspace classifier proposed in [2]) uses linear subspaces to approximate each class and a test sample is assigned to the class yielding the nearest linear subspace distance. The common vector method in [14], on the other hand, uses affine hulls (affine subspaces) to model classes and yields better accuracies than CLAFIC in most applications since affine hull models better localize the class regions compared to the linear subspaces [14–16]. However, both linear and affine subspaces are very loose approximations in the sense that they do not specify where the class is within the linear/affine subspaces. However, they are computationally very efficient (since linear/affine subspace parameters can be computed offline) despite their lack of selectivity and they surprisingly work well in high-dimensional classification problems. In fact, the most of the state-of-the-art classifiers are obtained by introducing additional constraints on linear/affine subspace combination coefficients. For example, both [17] and [18] enforced sparsity on the samples used for creating subspaces in face recognition. Same sparse linear/affine models were also successfully used in bag of words models in the context of visual object classification [19, 20]. In a similar manner, the authors in [21] used regularized affine hull models to represent image sets where L2-norms of affine hull combination coefficients are minimized when computing the smallest distances between affine hulls.

The authors in [5, 22] used a tighter convex hull modeling for approximating classes. It should be noted that the convex hulls are the smallest convex regions containing samples of a particular class and the popular support vector machine (SVM) classifier also uses convex hulls to approximate the regions of positive and negative classes. Although convex hulls tend to produce overtight approximations, they outperform linear/affine subspace models for more general classification problems cast in low-dimensional spaces [23, 24]. However, unlike linear/affine subspaces, their parameters cannot be computed in advance, which makes them computationally inefficient for real-time applications. The nearest convex hull classifier is extended for classification problems where data are represented with tensors in [25].

More recently, new methods have used moderate models that are between affine and convex hulls in terms of tightness. For example, the compact hypersphere model is the basis of the 1-class SVM [7] and it is used for outlier detection. The authors in [4, 26] introduced hyperdisks which are disk-shaped regions formed by the intersection of the affine hull and the bounding hyperspheres. The hyperdisk approximation is still somewhat loose but it encodes both the relevant variables and the region occupied by the class within their subspace. The authors in [6] used convex cone models which are constructed by using conic combinations of class samples for face and pedestrian detection. Convex cones are also used in [27] for image set-based recognition. Zhao et al. [28] used symmetric positive definite models for the nearest-convex class classification.

It should be noted that all methods described above are global methods that use a single convex model for approximating a particular class. However, convex class models can also be used locally by approximating a few samples returned by a selective algorithm (e.g., k-nearest neighbor) as in [3, 8]. Also, we would like to point out that convex class models are widely used for image set classification where the user supplies a set of image feature vectors as query rather than supplying a single query feature sample [16–18, 21] and for improving the clustering accuracy in high-dimensional spaces [29].

In addition, the large-margin-based classifiers also use convex class models to find linear separating hyperplanes [15, 26, 30–32]. The prototypical method of this kind, SVM, finds a linear hyperplane in feature space that maximizes the "margin" between convex hulls of positive and negative classes [30]. Similarly, the authors in [15] find the separating hyperplane that best separates affine hulls of positive and negative classes,

whereas the authors in [26] find the margin classifier that best separates the binary classes approximated with the hyperdisks. These large-margin-based classifiers are different from the nearest-class-model type classifiers since they return linear pairwise decision boundaries. On the other hand, the pairwise decision boundaries returned by the nearest-class-model type classifiers are not linear; they are generically at least quadratic or piecewise quadratic. For example, for affine hulls they are generally hyperboloids and they are typically more flexible than linear decision boundaries returned by the large margin classifiers.

In this paper, we introduce new convex class models for the nearest convex model type classification. To this end, we approximate each class with a polyhedral convex cone (PCC) or extended PCC (EPCC) by utilizing polyhedral conic functions and their extensions. The proposed PCC and EPCC models are very different than the convex cones and they can be seen as a crude approximation of the convex hulls. We first derive the formulations needed to compute the distances from the test samples to the PCC and EPCC models and then compare these models with the existing convex class models in terms of accuracy. The proposed models typically achieve the best accuracies among all tested methods for high-dimensional classification problems showing that the proposed models successfully approximate the true class regions in high-dimensional feature spaces.

2. Related methods

In the nearest-class-model type classifiers, each class in the training set is approximated with a convex class model and the distances from the test sample to each class model are computed. Then, the test sample is assigned to the class with the closest distance. In this section, we first briefly describe the related convex class models and discuss their advantages and limitations. Characteristic properties of the convex class models are briefly summarized in Table 1.

Table 1. Convex class models (SVD, singular value decomposition; QP, quadratic programming; LP, linear programming).

Convex class models	Model parameters	Solver type
Linear subspace	$\mathbf{U}^c$ (computed offline)	SVD
Affine subspace	$\mathbf{U}^c$, μ_c (computed offline)	SVD
Convex hull	$\boldsymbol{\alpha}_c^*$ (computed online during testing time)	QP
Convex cone	$\boldsymbol{\alpha}_c^*$ (computed online during testing time)	QP
Polyhedral convex cone	$\widetilde{\mathbf{w}}_c \equiv \left(\begin{smallmatrix}\mathbf{w}_c\\ \gamma_c\end{smallmatrix}\right)$, b_c (computed online during testing time)	LP

2.1. Linear subspace models

Let us suppose there are C classes in the training set and $\mathbf{x}_i^c$ denotes the i-th sample of class c. A linear subspace class model of a particular class c is the linear subspace spanned by its training samples, i.e.,

$$LS_c = \left\{ \mathbf{x} | \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i^c \mathbf{x}_i^c, \;\; \alpha_i^c \in \mathbb{R} \right\}. \tag{1}$$

To find the minimum distance from a test sample to a linear subspace, one has to solve the following unconstrained optimization problem:

$$\arg\min_{\boldsymbol{\alpha}_c} \;\; \left\| \mathbf{x}_{test} - \mathbf{X}^c \boldsymbol{\alpha}_c \right\|, \tag{2}$$

where $\mathbf{X}^c$ denotes the matrix whose columns are the samples of the c-th class. By taking derivative of the objective function and equating it to zero, it can be shown that minimizing this distance becomes the maximizing $\|\mathbf{U}^c\mathbf{x}_{test}\|$ where $\mathbf{U}^c$ is the matrix whose columns are the orthonormal basis vectors spanning the linear subspace of class c [9, 10]. Numerically, $\mathbf{U}^c$ can be found by applying an eigen-decomposition on correlation matrix $\mathbf{X}^c(\mathbf{X}^c)^\top$ or it can be found as the U-matrix of the "thin" singular value decomposition (SVD) of $\mathbf{X}^c$. Here, "thin" indicates that we take only the columns of U corresponding to "significantly nonzero" singular values. This subspace estimation process is essentially orthogonal least squares fitting. Discarding near-zero singular values corresponds to discarding directions that appear to be predominantly "noise". It should be noted that the length of the test vector does not contribute to the classifier decision, i.e. classifier using linear subspace models is invariant to the test vector length and makes the decision based on the angle between the test vector and the subspace.

Linear subspace modeling is a very loose approximation to a class region and it does not specify where the class samples lie within the linear subspace. Also, linear subspaces can only be used for pattern classification problems where the dimensionality is larger than the number of samples in each class (otherwise the distances from test samples to linear subspaces become zero, which makes the method useless). However, they are very efficient in terms of real-time performance since linear subspace parameters (orthonormal basis vector matrix – $\mathbf{U}^c$) can be computed offline and one has to make simple matrix multiplications to compute the distance, $\|\mathbf{U}^c\mathbf{x}_{test}\|$, during testing. State-of-the-art face recognition methods such as [17] are obtained by enforcing sparsity on linear combination coefficients $\boldsymbol{\alpha}_c$ via L1 norm.

2.2. Affine subspace models

Affine subspace (or affine hull) of a particular class is affine span of its training samples:

$$AS_c = \left\{ \mathbf{x} | \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i^c \mathbf{x}_i^c, \;\; \alpha_i^c \in \mathbb{R}, \;\; \sum_{i=1}^{n_c} \alpha_i^c = 1 \right\}. \tag{3}$$

Affine subspaces can be seen as shifted linear subspaces and they can better localize the class samples compared to the linear subspaces. To get rid of the equality constraint, we can choose any reference point from the class samples, e.g., the mean $\mu_c = (1/n_c)\sum_{i=1}^{n_c} \mathbf{x}_i^c$, and rewrite (3) as:

$$AS_c = \left\{ \mathbf{x} | \mathbf{x} \;=\; \mu_c + \mathbf{U}^c\boldsymbol{\alpha}^c, \;\; \boldsymbol{\alpha}^c \in \mathbb{R}^l \right\}, \tag{4}$$

where $\mathbf{U}^c$ is an orthonormal basis for the directions spanned by the $\left\{\mathbf{x}_1^c - \mu_c, \ldots, \mathbf{x}_{n_c}^c - \mu_c\right\}$, and l is the number of basis vectors. Numerically, $\mathbf{U}^c$ can be found by applying an eigen-decomposition on covariance matrix or by applying SVD to the mean subtracted examples. By following similar steps as in linear subspaces, the distance from a test sample to an affine subspace can be computed by:

$$d(\mathbf{x}_{test}, AS_c) \;=\; \|\mathbf{x}_{test} - \mu_c - \mathbf{P}_c(\mathbf{x}_{test} - \mu_c)\|, \tag{5}$$

where $\mathbf{P}_c \;=\; \mathbf{U}^c(\mathbf{U}^c)^\top$ is the orthogonal projection operator onto the affine subspace of class c. In contrast to the linear subspaces, the distance from a test sample to an affine subspace depends on the length of the test sample. Therefore, affine subspaces distances are not invariant to the scaling of the test feature sample.

As in linear subspaces, affine hulls are only suitable when the dimensionality of the sample space is larger than the number of samples of classes, and the affine subspace parameters can be computed offline so

the distances can be computed in real-time during testing. Affine subspaces surprisingly work well especially in high-dimensional classification problems, e.g., the state-of-the-art common vector and discriminative common vectors [14] use affine subspace distances for classification.

2.3. Convex hull models

The convex hull of a class is the smallest convex set containing its samples and it requires adding additional constraints, $\alpha_i^c \geq 0,\ i = 1, \ldots, n_c$, to (3). Therefore, finding the distance between a test sample and a convex hull requires the solution of the following quadratic programming (QP) problem:

$$\begin{aligned} \arg\min_{\boldsymbol{\alpha}_c} \ & \tfrac{1}{2} \left\| \mathbf{x}_{test} - \mathbf{X}^c \boldsymbol{\alpha}_c \right\| \\ \text{s.t.} \ & \sum_{i=1}^{n_c} \alpha_i^c = 1, \ \ \alpha_i^c \geq 0, \ \ i = 1, \ldots, n_c. \end{aligned} \tag{6}$$

Once the optimal coefficient vector $\boldsymbol{\alpha}_c^*$ is found, $\left\| \mathbf{x}_{test} - \mathbf{X}^c \boldsymbol{\alpha}_c^* \right\|$ determines the distance from the test sample to the convex hull of the class c.

This model makes few assumptions but tends to produce "overtight" approximations for high-dimensional spaces since for convex classes in high dimensions with practical numbers of training examples, almost all of the class volume typically lies outside the convex hull of the samples but they significantly outperform linear/affine spaces for classification problems cast in lower dimensional spaces. However, finding distances is computationally very expensive, which is the major limitation of the convex hulls because the convex hull typically has exponentially many facets so it cannot be stored explicitly as in linear/affine subspaces, instead, we need to solve QP problem of (6) for each test point at run-time.

2.4. Convex cone models

Convex cone of a class c includes conic combinations of its samples, i.e.,

$$CC_c = \left\{ \mathbf{x} | \mathbf{x} = \sum_{i=1}^{n_c} \alpha_i^c \mathbf{x}_i^c, \ \ \alpha_i^c \geq 0 \right\}. \tag{7}$$

The only difference between a convex cone and a convex hull is that the convex cone does not have the equality constraint, $\sum_{i=1}^{n_c} \alpha_i^c = 1$. Thus, convex cones are looser models compared to convex hulls. The distance from a test sample to a convex cone can be found by solving the following quadratic programming (QP) problem:

$$\begin{aligned} \arg\min_{\boldsymbol{\alpha}_c} \ & \tfrac{1}{2} \left\| \mathbf{x}_{test} - \mathbf{X}^c \boldsymbol{\alpha}_c \right\| \\ \text{s.t.} \ & \alpha_i^c \geq 0, \ \ i = 1, \ldots, n_c. \end{aligned} \tag{8}$$

Once the optimal convex cone combination parameters are found, the distance can be computed as $\left\| \mathbf{x}_{test} - \mathbf{X}^c \boldsymbol{\alpha}_c^* \right\|$. As in convex hulls, computing distances is computationally expensive since we need to solve QP problem of (8) for each test point at run-time. It should be noted that the distances to the convex cone models are sensitive to the test vector length, but they are invariant to the scaling of the training class samples.

3. Polyhedral conic functions and polyhedral convex cones

3.1. Preliminaries

In the proposed method, we approximate each class with a polyhedral conic region constructed by using polyhedral conic functions (PCFs). PCFs were first introduced in [33] to separate two arbitrary finite disjoint sets, and its extended version, EPCFs, were proposed in [34, 35] for visual object detection and classification. A graph of a PCF is a polyhedral cone with a sublevel set including an intersection of at most 2^d half spaces as seen in Figure 1. PCFs and EPCFs can be defined as follows:

$$f_{\mathbf{w},\gamma,\mathbf{s},b}(\mathbf{x}) = \mathbf{w}^\top(\mathbf{x}-\mathbf{s}) + \gamma \left\|\mathbf{x}-\mathbf{s}\right\|_1 - b \qquad \text{(PCF)} \tag{9}$$

$$f_{\mathbf{w},\boldsymbol{\gamma},\mathbf{s},b}(\mathbf{x}) = \mathbf{w}^\top(\mathbf{x}-\mathbf{s}) + \boldsymbol{\gamma}^\top|\mathbf{x}-\mathbf{s}| - b, \qquad \text{(EPCF)} \tag{10}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a test point, $\mathbf{s} \in \mathbb{R}^d$ is the cone vertex, $\mathbf{w} \in \mathbb{R}^d$ is a weight vector, and b is an offset. For PCF, $\left\|\mathbf{u}\right\|_1 = \sum_{i=1}^{d} |u_i|$ denotes the vector L1 norm and γ is a corresponding weight, while for EPCF, $|\mathbf{u}| = (|u_1|, \ldots, |u_d|)^\top$ denotes the component-wise modulus and $\boldsymbol{\gamma}$ is a corresponding weight vector. The fact that such functions define a polyhedral cone follows from the following lemma [33].

Lemma 1 *A graph of the function* $\underset{\mathbf{w},\mathbf{s},\gamma,b}{f(\mathbf{x})}$ *defined in (9) is a polyhedral cone with a vertex at* $(\mathbf{s}, -b)$.

The authors in [33, 34, 36, 37] introduced polyhedral conic classifiers that use PCFs and EPCFs with decision regions $f(\mathbf{x}) \leq 0$ for positives and $f(\mathbf{x}) > 0$ for negatives (opposite of the well-known support vector machine classifier decision rule). For PCF with $b > 0$, $\gamma > 0$, $\left\|\mathbf{w}\right\|_\infty < \gamma$ (where $\left\|\mathbf{u}\right\|_\infty = \max(|u_i|)_{i=1,\ldots,d}$ is the ∞ norm) and any τ, the region $f(\mathbf{x}) < \tau$ is convex and compact in $\mathbb{R}^d$ and it contains $\mathbf{s}$. Similarly, for EPCF, $b > 0$, $\boldsymbol{\gamma} > \mathbf{0}$, $|w_i| < \gamma_i$ for all $i = 1, \ldots, d$, and any τ, the region $f(\mathbf{x}) < \tau$ is convex and compact in $\mathbb{R}^d$ and it again contains $\mathbf{s}$. It should be noted that polyhedral conic regions are looser models compared to convex hulls, but very tight models compared to affine subspaces and convex cones.

3.2. Finding distances from test samples to polyhedral convex conic regions

We need to compute the distances from the test samples to the polyhedral conic models to classify test samples during online testing stage as seen in Figure 2. In this section, we derive the necessary formulations needed to compute such distances. To find the distance from a test sample to the polyhedral conic region spanned by samples of class c, we first augment a feature vector $\mathbf{x}$ as $\widetilde{\mathbf{x}} \equiv \begin{pmatrix} \mathbf{x}-\mathbf{s}_c \\ \left\|\mathbf{x}-\mathbf{s}_c\right\|_1 \end{pmatrix} \in \mathbb{R}^{d+1}$ and the weight vector to $\widetilde{\mathbf{w}}_c \equiv \begin{pmatrix} \mathbf{w}_c \\ \gamma_c \end{pmatrix} \in \mathbb{R}^{d+1}$ for PCF. Similarly, for EPCF we augment the feature vector as $\widetilde{\mathbf{x}} \equiv \begin{pmatrix} \mathbf{x}-\mathbf{s}_c \\ |\mathbf{x}-\mathbf{s}_c| \end{pmatrix} \in \mathbb{R}^{2d}$ and the weight vector to $\widetilde{\mathbf{w}}_c \equiv \begin{pmatrix} \mathbf{w}_c \\ \boldsymbol{\gamma}_c \end{pmatrix} \in \mathbb{R}^{2d}$. We set the cone vertex, $\mathbf{s}$, to the class mean as in [34]. Then, the distance from the augmented test sample $\widetilde{\mathbf{x}}_{test}$ to the polyhedral convex region spanned by samples of class c

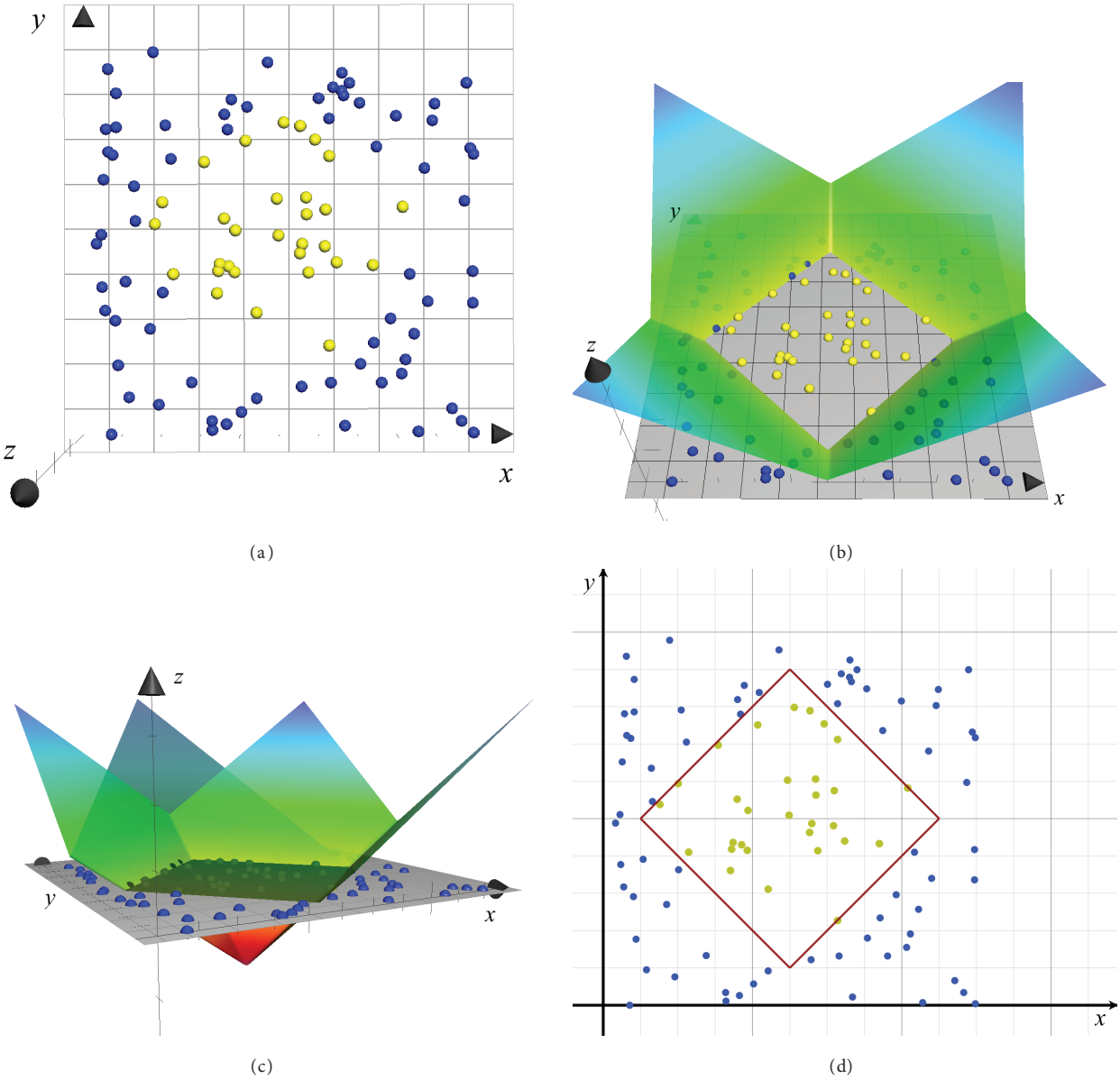

Figure 1. Visualization of PCC classifiers for 2D synthetic data: The positive acceptance regions are "kite-like" octahedroids containing the points for which a linear hyperplane lies above an L_1 cone. (a): 2D positive (yellow) and negative (blue) samples; (b) and (c): view of positive-class acceptance region in 3D; (d): Resulting "kite-like" acceptance region in 2D space.

can be computed by solving the following linear optimization problem:

$$\begin{aligned} \underset{\mathbf{w}_c,\gamma_c(\boldsymbol{\gamma}_c),b_c}{\arg\max} \quad & \left(\widetilde{\mathbf{w}}_c^\top \widetilde{\mathbf{x}}_{test} - b_c\right) \\ \text{s.t.} \quad & \widetilde{\mathbf{w}}_c^\top \widetilde{\mathbf{x}}_{test} - b_c \geq 0, \\ & \widetilde{\mathbf{w}}_c^\top \widetilde{\mathbf{x}}_i^c - b_c \leq 0, \;\; i = 1, \ldots, n_c. \\ & b_c > 0, \;\; \gamma_c > 0 (PCF), \;\; \boldsymbol{\gamma}_c > \mathbf{0} (EPCF). \end{aligned} \tag{11}$$

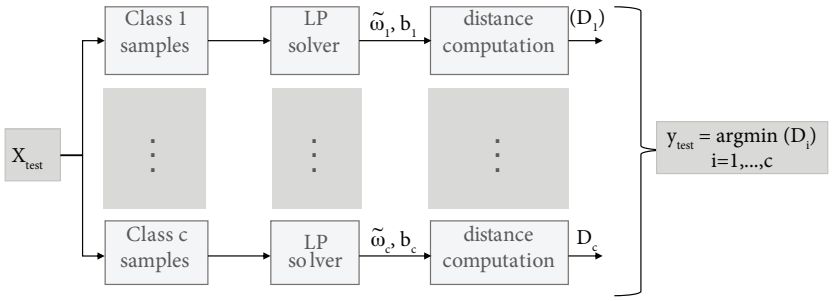

Figure 2. Illustration of the proposed method: The distances from a test sample to the polyhedral conic regions of classes are computed by using a linear programming (LP) solver. Then, the label of the test sample, y_{test}, is assigned to the class giving the smallest distance.

This optimization problem formulates the problem as finding the maximum distance from the test sample to a closed polyhedral conic region that inscribes all samples of class c. The distance is maximized when the parameters $\widetilde{\mathbf{w}}_c, b_c$ of PCF/EPCF return the smallest compact polyhedral region including $c-$th class samples. As a result, we compute the shortest distances from test samples to the most compact PCC and EPCC class models. Note that this is a standard linear optimization problem with inequality constraints and some bounds, so any LP solver can be used to solve this simple problem. Once the optimal PCF and EPCF parameters $(\widetilde{\mathbf{w}}_c, b_c)$ are found, the distance to polyhedral convex conic regions (PCCR) can be computed as:

$$d(\mathbf{x}_{test}, PCCR_c(EPCCR_c)) \;=\; \frac{(\widetilde{\mathbf{w}}_c^\top \widetilde{\mathbf{x}}_{test} - b_c)}{\|\widetilde{\mathbf{w}}_c\|}. \tag{12}$$

Then the test samples are assigned to the class yielding the smallest distance. We call the resulting methods using the polyhedral and extended polyhedral conic regions as PCCR and EPCCR, respectively.

3.3. Comparison to other nearest-convex-model classifiers

The proposed method uses PCC and EPCC models for approximating class regions. These models are slightly looser than the convex hulls, but they are tighter compared to remaining convex class models including linear/affine subspaces and convex cones. Regarding testing time efficiency, the distance computation from test samples to polyhedral conic regions needs to solve an optimization problem during online testing time as in computing convex hull and convex cone distances. Therefore, this procedure is slower compared to the methods using linear/affine subspaces, but it is faster compared to the method using convex hull since it solves a simple LP problem instead of a computationally more expensive QP problem.

4. Experiments

We tested the proposed methods on various datasets with different dimensionalities and training set sizes. The proposed PCCR and EPCCR methods solve the LP problem given (11) for each test sample at run time. We compared our proposed methods to the ones using linear subspaces (LS), affine subspaces (AS), convex hulls (CH), convex cones (CC), and linear hyperspheres (LHS).

4.1. Face recognition

We used the AR Face dataset[1] for face recognition experiments. The AR Face dataset includes frontal views of 126 subjects, with two groups of 13 images per subject recorded in two sessions spaced by 14 days. In our experiments, we randomly selected 20 male and 20 female subjects, aligning, down-scaling, and cropping the images to 105×78 pixels. Raw pixel values are used as features; thus, the dimensionality of the input space, 8190, is much higher than the total number of samples in each class. We randomly selected 13 images of each person for training and the remaining 13 images are used for testing. This is repeated 10 times and the final accuracies are averages of accuracies obtained in 10 trials.

Table 2 reports the classification accuracies and testing times of the classifiers. The best accuracies are obtained by the proposed PCCR and EPCCR methods, and they significantly outperform other methods with exception of AS classifier. Since the dimensionality of the input space is much larger than the number of samples in each class, affine subspace models significantly outperform convex hull and convex cone models as expected. However, it was surprising to see that EPCCR and PCCR methods which compute distances from test samples to polyhedral convex conic regions at run time even outperform affine subspace models. Linear subspaces perform poorly compared to other convex class models except for hypersphere ones. The worst accuracies are obtained by method using linear hyperspheres.

In terms of the speed, linear hypersphere is the fastest one followed by linear/affine subspace classifiers. These classifiers are fast since their parameters are precomputed before testing time. Our proposed classifiers are faster than both CH and CC methods using QP solvers, but slower compared to the methods using linear hypersphere and linear subspaces.

Table 2. Classification Rates (%) on the AR Face Dataset.

Method	Accuracy	Run time (s)
LS	77.3 ± 2.6	4×10^{-3}
AS	94.9 ± 1.5	11×10^{-3}
CH	83.0 ± 2.4	33×10^{-2}
CC	85.1 ± 1.8	14×10^{-2}
LHS	45.6 ± 2.7	16×10^{-4}
PCCR	95.5 ± 1.6	10×10^{-3}
EPCCR	95.4 ± 1.5	15×10^{-3}

4.2. Visual object recognition

We tested the proposed methods on two visual object recognition datasets, Cifar10[2] and Coil100[3] datasets. Cifar10 dataset includes 60K 32×32 small images of 10 objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck as shown in Figure 3. 50K samples are used as training and they are split into 5 batches whereas the remaining 10K samples are used for testing. We used 4096-dimensional CNN features for Cifar10 dataset experiments. To extract CNN features, all images are first resized to 256×256 and then we used Caffe implementation of the AlexNet architecture described in [38] by using the identical setting used for

[1] http://www2.ece.ohio-state.edu/∼aleix/ARdata-base.html
[2] https://www.cs.toronto.edu/∼kriz/cifar.html
[3] http://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php

ILSVRC 2012 classification with the exception that the base learning rate was set to 0.001. AlexNet includes 8 layers, 5 of which are convolutional layers and the remaining 3 are the fully connected layers. ReLU activation functions are used in the network, and soft-max classification loss is used for classification. We used 80% of the full training data for training and the remaining 20% as validation to train the CNN classifier. We used data augmentation for training. The number of iterations is set to 120K.

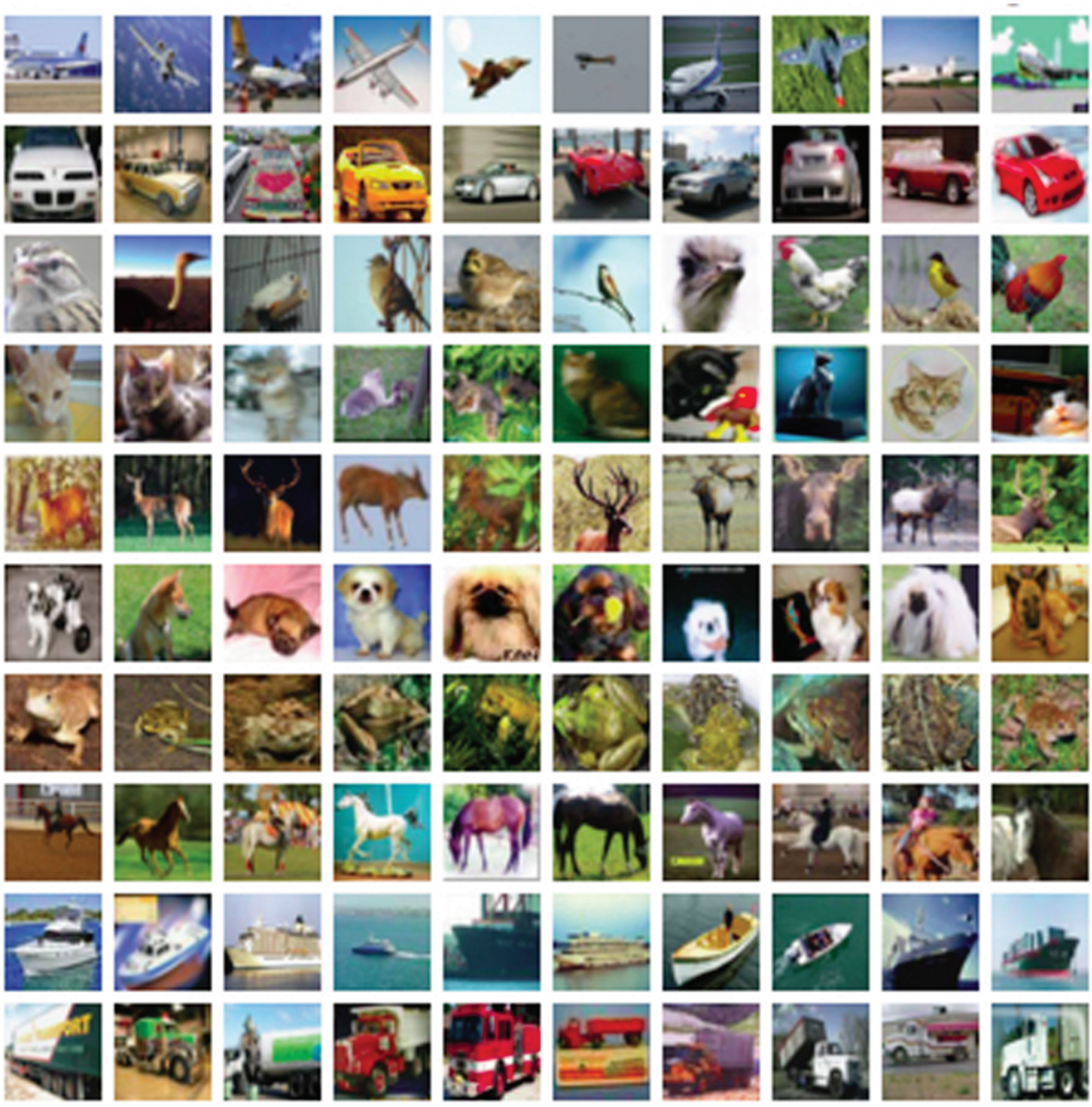

Figure 3. Some selected images from Cifar10 dataset. Cifar10 has 32×32 small images of 10 objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each row corresponds to a class in the figure.

The classification accuracies and testing times are given in Table 3. All methods achieve similar results except for the method using linear hypersphere which yields the worst classification accuracy. The best classification accuracy is obtained by the method using convex cones, and it is closely followed by the proposed EPCCR method. It should be noted that the number of samples per class is slightly larger than the feature dimensionality (feature dimension is 4096, and the number of samples per class is around 5000) for Cifar-10 dataset. As a result, all methods yielded similar accuracies with the exception of linear hypersphere since a linear bounding hypersphere is not a very accurate model for approximating class regions. The AlexNet which is initialized with random weights yielded an accuracy of 79.17%, which is significantly worse than the accuracies of the proposed methods. In terms of testing time, the methods whose parameters are computed offline are again the fastest classifiers. In contrast to the face recognition experiments, CC method is faster compared to the proposed methods despite it solves a QP problem. However, our proposed methods are faster than CH method as before.

The Coil100 dataset includes 100 different objects and 72 views of each object taken at 5-degree-apart orientations. We used gray-scale values of images down-sampled to 64×64, which in turns yields a 4096 dimensional feature space. In our experiments we used 40 classes. We randomly selected 36 samples from each class for training and the remaining 36 samples are used for testing. This is repeated 10 times and the final accuracies given in Table 4 are averages of results obtained in each trial. As in face recognition experiments, the dimensionality of the input space is much higher than the training samples of each class. The best classification accuracies are obtained by the proposed PCCR and EPCCR methods and the CH method using convex hulls. The worst accuracy is again obtained by linear hyperspheres. The number of samples per class is very low compared to the dimensionality and the looser models such as affine and linear subspaces are expected to work better than the tight models. However, since the image samples belong to the same identity for each object class, there is a very little intraclass variation; therefore, the tighter models such as convex hulls and convex cones also worked well for this particular dataset. The testing times are similar to the ones computed for Cifar10 dataset, and the proposed methods are faster than only CH method.

Table 3. Classification rates (%) on the CIFAR10 dataset.

Method	Accuracy	Run time (s)
LS	84.0 ± 0.2	45×10^{-4}
AS	83.8 ± 0.2	18×10^{-3}
CH	83.9 ± 1.1	60.88
CC	84.3 ± 0.3	41×10^{-2}
LHS	80.7 ± 3.6	2×10^{-4}
PCCR	83.9 ± 0.5	10.80
EPCCR	84.2 ± 0.4	11.46

Table 4. Classification rates (%) on the Coil-40 dataset.

Method	Accuracy	Run time (s)
LS	98.8 ± 0.6	28×10^{-4}
AS	99.1 ± 0.6	68×10^{-4}
CH	99.2 ± 0.5	64.62
CC	99.0 ± 0.5	0.37
LHS	76.2 ± 0.9	9×10^{-4}
PCCR	99.2 ± 0.6	17.53
EPCCR	99.2 ± 0.6	43.07

4.3. Hand-written digit recognition

We used USPS dataset for digit recognition. It contains 9298 16×16 gray-scale images of hand-written digits, with 7291 reserved for training and validation and the remaining 2007 for testing. In our experiments, we used raw gray-scale pixel values as features without any preprocessing. It should be noted that unlike other tested datasets, the training set size is larger than the dimensionality of the input space for this dataset.

The classification results are given in Table 5. The best accuracies are obtained by the proposed PCCR and EPCCR methods and they significantly outperform the other tested methods, which shows the superiority of the proposed classifiers. As expected, both methods using linear and affine subspace models perform rather poorly since the dimensionality is much smaller than the samples of each class. The method using linear hypersphere model again produces the worst accuracy. We also tested AlexNet on this dataset. The network is started from random weights as before. It yielded an accuracy of 95.47% which is slightly higher than the accuracies of the proposed methods. In terms of testing time, the proposed methods are faster than only CH method as in visual object classification experiments.

Table 5. Classification rates (%) on the USPS digit dataset.

Method	Accuracy	Run time (s)
LS	41.5	12×10^{-4}
AS	51.7	16×10^{-4}
CH	87.5	37.28
CC	91.7	27×10^{-3}
LHS	19.7	3×10^{-5}
PCCR	94.5	16.68
EPCCR	94.4	30.82

4.4. Experiments on UCI repository datasets

We tested the proposed methods on 5 binary and multiclass datasets chosen from UCI repository: Ionosphere, Iris, Multiple Features (MF) - pixel averages, Pima Indian Diabetes (PID), and Wisconsin Diagnostic Breast Cancer (WDBC) datasets. The sizes of the UCI problems are summarized in Table 6. As opposed to previous datasets, the dimensionality of the sample space is very low for these datasets.

We used 5-fold cross-validation to test the classifiers. The results are given in Table 7. The proposed EPCCR classifier achieves the best accuracy on Ionosphere dataset and significantly outperforms the other classifiers. For the remaining 4 classes, the proposed methods achieve the third best accuracy. CH method is the winner for MF and WDBC datasets, but the difference is not very significant. CC method achieves the best accuracy on Iris dataset, and AS method wins for the PID dataset, and again the performance difference is not very significant. Overall, the results are quite mixed, and there is not a clear winner on these low-dimensional classification problems.

Table 6. Datasets from the UCI Repository

Dataset	# classes	# examples	Dimension
Ionosphere	2	351	34
Iris	3	150	4
MF	10	2000	240
PID	2	768	8
WDBC	2	569	30

Table 7. Classification rates (%) on the UCI datasets.

Method	Ionosphere	Iris	MF	PID	WDBC
LS	70.4 ± 8.8	97.3 ± 2.8	97.0 ± 0.5	65.8 ± 4.8	88.8 ± 1.4
AS	84.6 ± 4.2	95.3 ± 3.8	94.1 ± 0.5	73.3 ± 2.9	84.5 ± 5.7
CH	85.2 ± 5.7	96.0 ± 5.4	98.5 ± 0.6	73.1 ± 4.7	95.3 ± 1.3
CC	75.5 ± 2.6	98.0 ± 3.0	98.4 ± 0.8	64.4 ± 5.8	94.6 ± 2.1
PCCR	92.9 ± 3.6	94.0 ± 4.9	98.1 ± 0.8	69.3 ± 3.4	93.8 ± 2.9
EPCCR	94.6 ± 3.6	96.0 ± 2.8	97.4 ± 0.6	70.7 ± 3.4	93.7 ± 2.5

4.5. Open set recognition

Here, we tested the proposed classifiers on the open set recognition task to assess the proposed models' rejection performance of the samples coming from other unseen classes. To this end, we used Cifar10 dataset. For the open set recognition setup, we randomly select 3 classes from training data and use the training samples coming from these classes alone during distance computation. During testing, we use all test samples coming from 10 classes of Cifar10. For accuracy, we compute AP (averege precision) scores from the precision-recall curves for the 3 classes and the take the mean of these. The precision-recall curves are plotted in Figure 4 and computed mAP scores are given in Table 8. As seen in the table, the proposed models achieve the best accuracies and significantly outperform all the other methods. This clearly shows that the proposed models successfully approximate class regions and reject the samples coming from the unknown classes with a high precision.

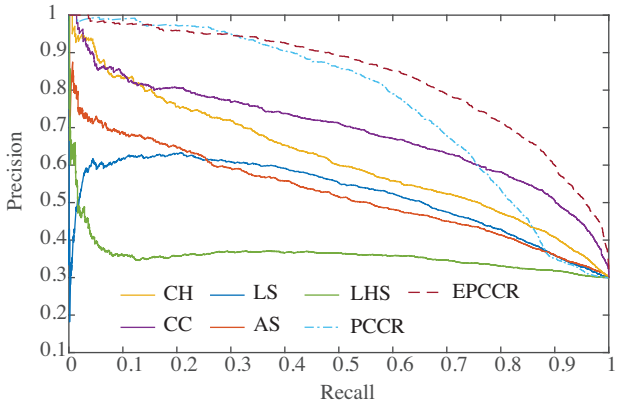

Figure 4. Precision-recall curves of the tested classifiers on open set recognition task.

5. Conclusions

This paper introduces polyhedral convex conic regions (PCCRs) to approximate classes for the nearest-class-model type classification. To this end, we approximated each class region with a polyhedral convex cone and proposed a linear programming (LP)-based method to find the distances from test samples to the polyhedral convex conic regions. The polyhedral conic region models which are used to approximate the class regions are looser than convex hulls, but they are tighter compared to other convex models such as linear/affine

Table 8. AP scores of the tested classifiers.

Convex class models	AUC
LS	0.5189
AS	0.5241
CH	0.6127
CC	0.6902
LHS	0.3576
PCCR	0.7686
EPCCR	0.8370

subspaces and convex cones. We tested the proposed methods on various high-dimensional classification tasks including face, digit, and general object classification as well as on some lower-dimensional datasets. In addition, the proposed methods are tested on open set recognition setting to assess the rejection performance for the test samples coming from unknown classes. The proposed methods typically achieve the best accuracies on high-dimensional problems and significantly outperform the other rival classifiers especially on digit and face recognition problems. The proposed models also yielded the best accuracies for open set recognition. All these results demonstrate that the polyhedral conic regions can successfully estimate the true class regions on classification problems cast in high-dimensional spaces compared to other loose and tight models and they can easily reject the test samples coming from unknown classes that are not seen in the training (gallery) data.

Acknowledgment

This work was funded in part by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number EEEAG-116E080. Emre Çimen and Gürkan Öztürk is partially supported by the Scientific Research Projects commission of Eskisehir Technical University under the grants 19ADP022 and 19ADP081.

References

[1] Watanabe S, Pakvasa N. Subspace method in pattern recognition. In: International Conference on Pattern Recognition; Washington DC, USA; 1973. pp. 25-32.

[2] Liu Y, Ge SS, Li C, You Z. k-ns: A classifier by the distance to the nearest subspace. IEEE Transactions on Neural Networks 2011; 22: 1256-1268.

[3] Cevikalp H, Larlus D, Neamtu M, Triggs B, Jurie F. Manifold based local classifiers: Linear and nonlinear approaches. Journal of Signal Processing Systems 2010; 61: 61-73.

[4] Cevikalp H, Triggs B, Polikar R. Nearest hyperdisk methods for high-dimensional classification. In: The 25th International Conference on Machine Learning; Helsinki, Finland; 2008. pp. 120 - 127.

[5] Nalbantov GI, Groenen PJF, Bioch JC. Nearest convex hull classification. Technical report, Econometric Institute and Erasmus Research Institute of Management, 2007.

[6] Kobayashi T, Otsu N. Cone-restricted subspace methods. In: 19th International Conference on Pattern Recognition; Tampa, FL, USA; 2008.pp.1-25.

[7] Tax DMJ, Duin RPW. Support vector data description. Machine Learning 2004; 54: 45-66.

[8] Vincent P, Bengio Y. K-local hyperplane and convex distance nearest neighbor algorithms. In: Advances in Neural Information Processing Systems; Vancouver, Canada; 2001. pp. 1-20.

[9] Oja E. Subspace Methods of Pattern Recognition. New York, NY, USA: Research Studies Press, 1983.

[10] Laaksonen J. Subspace classifiers in recognition of handwritten digits. Ph.D. thesis, Helsinki University of Technology, 1997.

[11] Ho J, Yang M, Lim J, Lee K, Kriegman D. Clustering appearances of objects under varying illumination conditions. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR); Madison, WI, USA; 2003.pp.1-20.

[12] Costeira J, Kanade T. A multibody factorization method for independently moving objects. International Journal of Computer Vision 1998; 29: 159-179.

[13] Cheema MS, Eweiwi A, Bauckhage C. High dimensional low samples size activity recognition using geometric classifiers. Digital Signal Processing 2015; 42: 61-69.

[14] Gulmezoglu M, Dzhafarov V, Keskin M, Barkana A. A novel approach to isolated word recognition. IEEE Transactions on Speech and Audio Processing 1999; 7(6):620-628.

[15] Suykens JAK, Vandewalle J. Least squares support vector machine classifiers. Neural Processing Letters 1999; 9: 293-300.

[16] Cevikalp H, Triggs B. Face recognition based on image sets. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR); San Francisco, CA, USA; 2010. pp. 2567-2573.

[17] Wright J, Yang A, Ganes A, Sastry S, Ma Y. Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 2009; 31: 210-227.

[18] Hu Y, Mian AS, Owens R. Face recognition using sparse approximated nearest points between image sets. IEEE Transactions on Pattern Analysis and Machine Intelligence 2012; 34(3): 1992-2004.

[19] Bourreau Y, Bach F, LeCun Y, Ponce J. Learning mid-level features for recognition. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR); San Francisco, CA, USA; 2010. pp. 2559-2566.

[20] Yang J, Yu K, Gong Y, Huang TS. Linear spatial pyramid matching using sparse coding for image classification. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR); Miami, FL, USA; 2009. pp. 1794-1801.

[21] Yang M, Zhu P, Gool LV, Zhang L. Face recognition based on regularized nearest points between image sets. In: 10th IEEE International Conference on Automatic Face and Gesture Recognition; Shanghai, China; 2013. pp. 1-20.

[22] Nemirko AP. Lightweight nearest convex hull classifier. Pattern Recognition and Image Analysis 2019; 3: 360-365.

[23] Zhou X, Shi Y. Nearest neighbor convex hull classification method for face recognition. In: The International Conference on Computational Science; Louisiana, USA; 2009. pp. 570-577.

[24] Takahashi T, Kudo M, Nakamura A. Construction of convex hull classifiers in high dimensions. Pattern Recognition Letters 2011; 32(16): 2224-2230.

[25] Cheng Z, Wang R. Nearest neighbor convex hull tensor classification for gear intelligent fault diagnosis based on multi-sensor signals. IEEE Access 2019; 7: 140781-140793.

[26] Cevikalp H, Triggs B. Hyperdisk based large margin classifier. Pattern Recognition 2013; 46: 1523-1531.

[27] Sogi N, Nakayama T, Fukui K. A method based on convex cone model for image-set classification with cnn features. In: International Joint Conference on Neural Networks (IJCNN); Rio de Janeiro, Brazil; 2018. pp. 1-25.

[28] Zhao K, Wiliem A, Chen S, Lovell BC. Convex class model on symmetric positive definite manifolds. Image and Vision Computing 2019; 87: 57-67.

[29] Cevikalp H. High-dimensional data clustering by using local affine/convex hulls. Pattern Recognition Letters 2019; 128: 427-432.

[30] Bennett KP, Bredensteiner EJ. Duality and geometry in svm classifiers. In: 17th International Conference on Machine Learning (ICML); Stanford, CA, USA; 2000.pp.1-20.

[31] Cevikalp H, Triggs B. Large margin classifiers based on convex class models. In: International Conference on Computer Vision Workshops (ICCVW); Kyoto, Japan; 2009. pp. 101-108.

[32] Zhu R, Wang Z, Sogi N, Fukui K, Xue JH. A novel separating hyperplane classification framework to unify nearest-class-model methods for high-dmensional data. IEEE Transactions on Neural Networks and Learning Systems 2020; 1: 1-11.

[33] Gasimov RN, Ozturk G. Separation via polyhedral conic functions. Optimization Methods and Software 2006; 21: 527-540.

[34] Cevikalp H, Triggs B. Polyhedral conic classifiers for visual object detection and classification. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Honolulu, HI, USA; 2017. pp. 4114-4122.

[35] Cevikalp H, Saglamlar H. Polyhedral conic classifiers for computer vision applications and open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 2019; 1-15.

[36] Ozturk G, Cimen E. Polyhedral conic kernel-like functions for SVMs. Turkish Journal of Electrical Engineering and Computer Sciences 2019; 27: 1172-1180.

[37] Cimen E, Ozturk G, Gerek ON. Incremental conic functions algorithm for large scale classification problems. Digital Signal Processing 2018; 77: 187-194.

[38] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: The 26th Annual Conference on Neural Information Processing Systems (NIPS); Harrahs and Harveys, USA; 2012.pp.1-20.