L1-norm Kernel PCA

Cheolmin Kim, Diego Klabjan

Abstract—We present the first model and algorithm for L1-norm kernel PCA. While L2-norm kernel PCA has been widely studied, there has been no work on L1-norm kernel PCA. For this non-convex and non-smooth problem, we offer geometric understandings through reformulations and present an efficient algorithm where the kernel trick is applicable. To attest the efficiency of the algorithm, we provide a convergence analysis including linear rate of convergence. Moreover, we prove that the output of our algorithm is a local optimal solution to the L1-norm kernel PCA problem. We also numerically show its robustness when extracting principal components in the presence of influential outliers, as well as its runtime comparability to L2-norm kernel PCA. Lastly, we introduce its application to outlier detection and show that the L1-norm kernel PCA based model outperforms especially for high dimensional data.

Index Terms—Principal Component Analysis, Robustness, Kernel, Outlier Detection.

1 INTRODUCTION

P
RINCIPAL Component Analysis (PCA) is one of the most popular dimensionality reduction techniques [1]. Given a large set of possibly correlated features, it attempts to find a small set of features (principal components) that retain as much information as possible. To generate such new dimensions, it linearly transforms original features by multiplying loading vectors in a way that newly generated features are orthogonal and have the largest variances.

In traditional PCA, variances are measured using the L2-norm. This has a nice property that although the problem itself is non-convex, the optimal solution can be easily found through matrix factorization. With this property, together with its easy interpretability, PCA has been extensively used in a variety of applications. However, despite of its success, it still has some limitations. First, since it generates new dimensions through a linear combination of features, it is not able to capture non-linear relationships between features. Second, as it uses the L2-norm for measuring variance, its solutions tend to be substantially affected by influential outliers. To overcome these limitations, the following two approaches have been proposed.

Kernel PCA The idea of kernel PCA is to map original features into a high-dimensional feature space, and perform PCA in that high-dimensional feature space [2]. With non-linear mappings, we can capture non-linear relationships among features, and this computation can be done efficiently using the kernel trick. With the kernel trick, computations of principal components can be done without an explicitly mapping.

L1-norm PCA To alleviate the effects of influential observations, L1-norm PCA uses the L1-norm instead of the L2-norm to measure variances. The L1-norm is more advantageous than the L2-norm when there are outliers having large feature values since it is less influenced by them. By utilizing this property, more robust results can be obtained through the L1-norm based formulation in the presence of influential outliers.

In this paper, we combine these two approaches for the variance maximization version of L1-norm PCA (which is not the same as minimizing the reconstruction error with respect to the L1-norm). In other words, we tackle a kernel version of L1-norm PCA. Unlike L2-norm kernel PCA, the kernel version of L1-norm PCA is a hard problem in that it is not only non-convex but also non-smooth. However, through a reformulation, we make it a geometrically interpretable problem where the goal is to minimize the L2-norm of a vector subject to a linear constraint involving the L1-norm terms. For this reformulated problem, we present a “fixed point” type algorithm that iteratively computes a -1,1 weight for each data point based on the kernel matrix and previous weights. We show that the kernel trick is applicable to this algorithm. Moreover, we prove the efficiency of our algorithm through a convergence analysis. We show that our algorithm converges in a finite number of steps and the objective values decrease with a linear rate. Lastly, we computationally investigate the robustness of our algorithm and illustrate its use for outlier detection.

Our work has the following contributions.

1. We are the first to present a model and an algorithm for L1-norm kernel PCA. While L2-norm kernel PCA has been widely used, a kernel version of L1-norm PCA has never been studied before. In this work, we show that the kernel trick which made L2-norm kernel PCA successful is also applicable for L1-norm kernel PCA.
2. We provide a rate of convergence analysis for our L1-norm kernel PCA algorithm. Although many algorithms have been proposed for L1-norm PCA, none of them provided a rate of convergence analysis. The work shows that our algorithm achieves a linear rate of convergence by exploiting the structure of the problem.
3. We introduce a methodology based on L1-norm kernel PCA for outlier detection.

In what follows, we always refer to the variance maximization version of L1-norm PCA and we assume that every variable in the input data is standardized with a mean.
of 0 and standard deviation of 1. This paper is organized as follows. Section 2 reviews recent works on $L_1$-norm PCA, and points out how our work is distinguishable from them. Section 3 covers various formulations of the kernel version of $L_1$-norm PCA. Through reformulations, we offer an understanding of the problem and based on these understandings we present our algorithm in Section 4. Section 5 gives a convergence analysis for our algorithm. Lastly, we show the robustness of our algorithm and its application to outlier detection in Section 6.

2 Related Work

$L_1$-norm PCA has been shown to be NP-hard in [3] and [4]. Nevertheless, an algorithm finding a global optimal solution is proposed in [3]. Utilizing the auxiliary-unit-vector technique [5], it computes a global optimal solution with complexity $O(nrp+p-1)$ where $n$ is the number of observations, $r$ is the rank of the data matrix, and $p$ is the desired number of principal components. Assuming $r$ and $p$ are fixed, the runtime of this algorithm is polynomial in $n$. However, if $n$, $p$, $r$ are large its computation time can be prohibitive. Rather than finding a global optimal solution which is intractable for general problems, our work focuses on developing an efficient algorithm finding a local optimal solution.

Recognizing the hardness of $L_1$-norm PCA, an approximation algorithm is presented in [4] based on the known Nesterov’s theorem [6]. In this work, $L_1$-norm PCA is relaxed to a semi-definite programming (SDP) problem and alternatively the SDP relaxation is considered. After solving the relaxed problem, it generates a random vector and uses a randomized rounding to produce a feasible solution. This randomized algorithm is a $\sqrt{2/\pi}$-approximate algorithm in expectation. To achieve this approximation guarantee with high probability, it performs multiple times of randomized roundings and takes the one having the best objective value. Instead of providing an approximation guarantee by solving a relaxed problem, our work directly considers the $L_1$-norm kernel PCA problem, and develops an efficient algorithm finding a local optimal solution.

Another approach using a known mathematical programming model is introduced in [7]. Specifically, it proposes an iterative algorithm that solves a mixed integer programming (MIP) problem in each iteration. Given an orthonormal matrix of loading vectors, it perturbs the matrix slightly in a way that the resulting matrix yields the largest objective value. After perturbation, it uses singular value decomposition to recover orthogonality. The algorithm is completely different from the one proposed herein, the objective values of the iterates do not necessarily improve over iterations. As opposed to it, our work shows monotone convergence of the objective values as well as a linear rate of convergence to a local optimal solution.

On the other hand, a simple numerical algorithm finding a local optimal solution is proposed in [8], and its extended version that finds multiple loading vectors at the same time is presented in [9]. In the former work, the optimal solution is assumed to have a certain form, and parameters involved in that form are updated at each iteration improving the objective values. For a linear kernel, our algorithm has the same form as this algorithm. However, while the algorithm in [9] is derived without any justification, we provide an understanding behind the algorithm. Moreover, we prove a rate of convergence analysis, and introduce a kernel version of it.

3 Kernel-based $L_1$-norm PCA Formulations

We consider $L_1$-norm PCA in a high-dimensional feature space $F$. Suppose we map data vectors $a_i \in \mathbb{R}^d$, $i=1,\ldots,n$ into a feature space $F$ by a possibly non-linear mapping $\Phi : \mathbb{R}^d \to F$. Assuming $|\Phi(a_i)^T\Phi(a_j)| < \infty$ for every $i, j=1,\ldots,n$, the kernel version of $L_1$-norm PCA can be formulated as follows.

$$\begin{align*}
\max_{x \in F} \quad & f(x) = \sum_{i=1}^{n} |\Phi(a_i)^T x| \\
\text{subject to} \quad & \|x\|_2 = 1
\end{align*}$$

As shown in (1), we only consider extracting the first loading vector. This assumption is justifiable since subsequent loading vectors can be found by iteratively running the same algorithm. Specifically, each time a new loading vector $x$ is obtained, we update the kernel matrix $K$ defined by $K_{ij} = \Phi(a_i)^T\Phi(a_j)$ by orthogonally projecting $\Phi(a_i)$, $i=1,\ldots,n$ onto the space orthogonal to the most recently obtained loading vector, and then run the same algorithm on the updated kernel matrix $K$.

The problem (1) has a convex non-smooth objective function to maximize. Moreover, the feasibility set is non-convex. To better understand this problem and derive an efficient algorithm, we reformulate (1) in the following way.

$$\begin{align*}
\min_{x \in F} \quad & g(x) = \|x\|_2 \\
\text{subject to} \quad & \sum_{i=1}^{n} |\Phi(a_i)^T x| = 1
\end{align*}$$

To prove the equivalence of two formulations, we show that an optimal solution of one formulation can be derived from an optimal solution of the other formulation.

**Proposition 1.** The following holds.

a) If $x^*$ is optimal to (1), then $\hat{x}^* = (1/n) \sum_{i=1}^{n} |\Phi(a_i)^T x^*| x^*$ is an optimal solution to (2).

b) If $y^*$ is optimal to (2), then $\hat{y}^* = y^*/\|y^*\|_2$ is an optimal solution to (1).

**Proof.** a) It is easy to check that $\hat{x}^*$ is feasible to (2). If $\hat{x}^*$ is not optimal to (2), there exists $z$ such that $\|z\|_2 < \|\hat{x}^*\|_2$. From its feasibility, it is obvious that $\sum_{i=1}^{n} |\Phi(a_i)^T z| = 1$ holds. Then, for $w = z/\|z\|_2$, we have

$$f(w) = \sum_{i=1}^{n} |\Phi(a_i)^T w| = (1/\|z\|_2) \sum_{i=1}^{n} |\Phi(a_i)^T z| = 1/\|z\|_2.$$

In the same way, we have

$$f(x^*) = 1/\|\hat{x}^*\|_2$$

due to $x^* = \hat{x}^*/\|\hat{x}^*\|_2$. From $\|z\|_2 < \|\hat{x}^*\|_2$, we have

$$f(x^*) = 1/\|\hat{x}^*\|_2 < f(w) = 1/\|z\|_2.$$
must hold, which contradicts the assumption that $x^*$ is an optimal solution of (1).

b) Again, it is easy to check $\hat{y}^*$ is feasible to (1). If $\hat{y}^*$ is not optimal to (1), then there exists $w$ such that

$$\sum_{i=1}^{n} |\Phi(a_i)^T \hat{y}^*| < \sum_{i=1}^{n} |\Phi(a_i)^T w|.$$

Since $\|w\|_2 = 1$, for $z = (1/\sum_{i=1}^{n} |\Phi(a_i)^T w|) w$, we have

$$g(z) = (1/\sum_{i=1}^{n} |\Phi(a_i)^T w|) \|w\|_2 = 1/\sum_{i=1}^{n} |\Phi(a_i)^T w|.$$

On the other hand,

$$y^* = (1/\sum_{i=1}^{n} |\Phi(a_i)^T y^*|) y^*$$

follows from $\hat{y}^* = y^*/\|y^*\|_2$ resulting in

$$g(y^*) = (1/\sum_{i=1}^{n} |\Phi(a_i)^T \hat{y}^*|) > g(z) = 1/\sum_{i=1}^{n} |\Phi(a_i)^T w|.$$

This contradicts the assumption that $y^*$ is optimal to (2). □

To understand formulation (2), we first examine the constraint set $\{x | \sum_{i=1}^{n} |\Phi(a_i)^T x| = 1\}$. Geometrically, this constraint set is symmetric with respect to the origin and represents a boundary of a polytope $P = \{x | \sum_{i=1}^{n} |\Phi(a_i)^T x| \leq 1\}$.

It is easy to check that $P$ is a polytope, since it can be represented by a finite set of linear constraints as

$$P = \{x | (\sum_{i=1}^{n} \Phi(a_i) c_i) x \leq 1 \text{ where } c_i \in \{-1, 1\}, i = 1, \ldots, n\}.$$

Therefore, formulation (2) is to find the closest point to the origin from the boundary of the polytope $P$. The following proposition shows an optimal solution of $x^*$ must be perpendicular to one of the faces.

**Proposition 2.** An optimal solution $x^*$ is perpendicular to the face which it lies on.

**Proof.** Suppose that an optimal solution of (2) is $x^*$. Assuming $c^*_i = \text{sgn}(\Phi(a_i)^T x^*)$, $i = 1, \ldots, n$, consider the face

$$F = \{x | (\sum_{i=1}^{n} \Phi(a_i) c^*_i) x = 1\} \cap P.$$

If $x^*$ is not perpendicular to face $F$, then

$$w = \sum_{i=1}^{n} \Phi(a_i) c^*_i / \sqrt{\sum_{i=1}^{n} \Phi(a_i) c^*_i} \|2\|$$

is the closest point to the origin from

$$\{x | (\sum_{i=1}^{n} \Phi(a_i) c^*_i) x = 1\}$$

having $\|w\|_2 < \|x^*\|_2$. Now, let us define its scalar multiple

$$z = (1/\sum_{i=1}^{n} |\Phi(a_i)^T w|) w.$$

By construction, $z$ is a feasible solution to (2) and has the objective value of

$$g(z) = (1/\sum_{i=1}^{n} |\Phi(a_i)^T w|) \|w\|_2.$$

From

$$\sum_{i=1}^{n} |\Phi(a_i)^T (\sum_{j=1}^{n} \Phi(a_j) c^*_j)| = \sum_{i=1}^{n} |\Phi(a_i)^T c^*_i| \|2\|$$

we have

$$\sum_{i=1}^{n} \Phi(a_i)^T w = \frac{\sum_{i=1}^{n} \Phi(a_i)^T (\sum_{j=1}^{n} \Phi(a_j) c^*_j)}{\sum_{i=1}^{n} |\Phi(a_i)^T c^*_i| \|2\|} \geq 1.$$

As a result,

$$g(z) = (1/\sum_{i=1}^{n} |\Phi(a_i)^T w|) \|w\|_2$$

follows. This contradicts the assumption that $x^*$ is an optimal solution to (2). Therefore, the optimal solution must be perpendicular to face $F$. □

From Proposition 2 we can easily derive the following Corollary 1.

**Corollary 1.** An optimal solution $x^*$ of (2) must have the form of $x^* = (1/\sum_{i=1}^{n} |\Phi(a_i)^T y^*|) y^*$ where $y^* = \sum_{i=1}^{n} \Phi(a_i) c^*_i$ and $c^*_i = \text{sgn}(\Phi(a_i)^T y^*)$, $i = 1, \ldots, n$.

The form that $x^*$ is a scalar multiple of $y^*$ is assumed in [8] for the linear kernel case without any justification but by Corollary 1 it follows that it is essentially the right form for the optimal solution. Moreover, from

$$\|x^*\|_2 = \|y^*\|_2 / \sum_{i=1}^{n} \Phi(a_i)^T y^* = \|y^*\|_2 / \sum_{i=1}^{n} c^*_i \Phi(a_i)^T y^*$$

we can further show that the optimal solution of formulation (2) can be derived from the optimal solution of the following binary problem.

$$\text{maximize} \sum_{i=1}^{n} \Phi(a_i) c^*_i \|2\|.$$  \hspace{1cm} (4)

**Proposition 3.** Let an optimal solution of binary formulation (4) be $c^* = [c^*_1, \ldots, c^*_n]^T$. Then, $y^* = \sum_{i=1}^{n} \Phi(a_i) c^*_i$ satisfies $c^*_i = \text{sgn}(\Phi(a_i)^T y^*)$, $i = 1, \ldots, n$. 
sgn(\(\Phi(a_i)^T y^*\)), for \(i = 1, \ldots, n\). Therefore, it follows that \(x^* = (1/ \sum_{i=1}^n |\Phi(a_i)^T y^*|) y^*\) is an optimal solution for formulation \((2)\).

**Proof.** Since \(c^* = [c_1^*, \ldots, c_n^*]^T\) is an optimal solution of \((4)\), flipping the sign of any \(c_i^*\), \(i = 1, \ldots, n\) must not improve the objective value, \(\| \sum_{i=1}^n \Phi(a_i)c_i^* \|_2^2\).

To deduce a contradiction, let us assume that there exists some \(J(\neq \emptyset) \subseteq \{1, \ldots, n\}\) such that \(c_J^* = -\text{sgn}(\Phi(a_i)^T y^*)\) for \(j \in J\).

Then, for any \(j \in J\), flipping the sign of \(c_j^*\) gives
\[
\| y^* - 2\Phi(a_j)c_j^* \|_2^2 = \| y\|_2^2 - 4y^T (\Phi(a_j)c_j^*) + 4\| \Phi(a_j) \|_2^2
\]
\[
= \| y\|_2^2 + 4y^T (\Phi(a_j)) + 4\| \Phi(a_j) \|_2^2
\]
\[
> \| y\|_2^2 = \| \sum_{i=1}^n \Phi(a_i)c_i^* \|_2^2
\]
which contradicts the assumption that \(c^*\) is an optimal solution to \((4)\). Therefore, \(y^* = \sum_{i=1}^n \Phi(a_i)c_i^*\) must satisfy
\[
c_i^* = \text{sgn}(\Phi(a_i)^T y^*)\]
for \(i = 1, \ldots, n\).

Since \(c^*\) maximizes \(\| \sum_{i=1}^n \Phi(a_i)c_i^* \|_2^2\),
\[
x^* = (1/ \sum_{i=1}^n |\Phi(a_i)^T y^*|) y^*
\]
is a minimizer for \((2)\) due to Corollary \(1\) and \(3\). \(\square\)

The following result has been shown in \(3\) for the linear kernel case but here we generalize it.

**Corollary 2.** Formulation \((2)\) is equivalent to formulation \((4)\).

**Proof.** Due to Corollary \(1\) and \(3\), formulation \((2)\) comes down to
\[
\text{maximize } c \in \{-1, 1\}^n \| \sum_{i=1}^n \Phi(a_i)c_i \|_2^2
\]
subject to
\[
y = \sum_{i=1}^n \Phi(a_i)c_i
\]
\[
c_i = \text{sgn}(\Phi(a_i)^T y), i = 1, \ldots, n
\]
Since an optimal solution \(c^*\) to \((4)\) satisfies the constraints by Proposition \(3\), the two formulations are essentially the same. \(\square\)

Interestingly, the binary formulation \((4)\) is NP-hard. Since expanding \(\| \sum_{i=1}^n \Phi(a_i)c_i \|_2^2\) gives
\[
\sum_{i=1}^n \sum_{j=1}^n \Phi(a_i)^T \Phi(a_j) + \sum_{i<j} (-\Phi(a_i)^T \Phi(a_j))(c_i - c_j)^2,
\]
assuming \(w_{ij} = (-\Phi(a_i)^T \Phi(a_j))\) be the weight of the complete graph having \(n\) nodes \(\{1, \ldots, n\}\), we can show the equivalence of the quadratic binary program \((4)\) and the max-cut problem.

## 4 Algorithm

In this section, we develop an efficient algorithm that finds a local optimal solution to problem \((2)\) based on the findings in Section \(3\). Before giving the details of the algorithm, we first provide an idea behind the algorithm.

The main idea of the algorithm is to move along the boundary of \(P\) so that the \(L_2\)-norm of the iterate decreases. Figure 1 graphically shows a step of Algorithm 1. Starting with iterate \(x^k\), we first identify hyperplane \(h^k\) which current iterate \(x^k\) lies on. After identifying the equation of \(h^k\), we find the closest point to the origin from \(h^k\), which we denote by \(z^k\). After that, we obtain \(x^{k+1}\) by projecting \(z^k\) to the constraint set by multiplying it by an appropriate scalar. We repeat this process until iterate \(x^k\) converges.

![Fig. 1. Geometric interpretation of the algorithm](image)

Next, we develop an algorithm based on the above idea. Let \(K_{ij} = \Phi(a_i)^T \Phi(a_j)\). From Corollary \(1\) we know that the optimal solution \(x^*\) has the form of
\[
x^* = (1/ \sum_{i=1}^n |\Phi(a_i)^T y^*|) y^* = (1/ \sum_{i=1}^n c_i\Phi(a_i)^T y^* y^*)
\]
\[
= (1/ \| c^* \|_2^2) \sum_{i=1}^n \Phi(a_i)c_i^*.\]
Utilizing the fact that the optimal solution \(x^*\) is characterized by the sign vector \(c^*\), we characterize the initial iterate \(x^0\) with the sign vector \(c^0\) as
\[
x^0 = (1/ \| c^0 \|_2^2) \sum_{i=1}^n \Phi(a_i)c_i^0.
\]
With \(y^k = \sum_{i=1}^n \Phi(a_i)c_i^k\), the equation of the hyperplane \(h^k\) is represented by
\[
(y^k)^T (x - x^k) = 0.
\]
The closest point \(z^k\) to the origin among the points in the hyperplane \(h^k\) has the form of \(z^k = sy^k\). By plugging \(z^k = sy^k\) into \((y^k)^T (x - x^k) = 0\), we have
\[
s = \frac{(y^k)^T (x^k)}{(y^k)^T (y^k)}, \quad z^k = \frac{(y^k)^T (x^k)}{(y^k)^T (y^k)} y^k.
We multiply \( z^k \) by \((1/ \sum_{i=1}^n |\Phi(a_i)^T z^k|)\) to make it feasible and thereby get
\[
x^{k+1} = (1/ \sum_{i=1}^n |\Phi(a_i)^T z^k|)z^k.
\] (5)
Utilizing
\[
(y^k)^T(x^k) = \sum_{i=1}^n \Phi(a_i)^T x^k c^k_i = \sum_{i=1}^n |\Phi(a_i)^T x^k| = 1,
\] (6)
we get the followings.
\[
y^k = \sum_{i=1}^n \Phi(a_i)c^k_i, \quad \forall i \in \{1,...,n\}
\] (7)
\[
z^k = \frac{\|y^k\|_2}{2}, \quad \forall i \in \{1,...,n\}
\] (8)
\[
x^{k+1} = (1/ \sum_{i=1}^n |\Phi(a_i)^T y^k|)y^k.
\] (9)
By plugging (7) into (9), \(x^{k+1}\) can be represented as
\[
x^{k+1} = (1/ \sum_{i=1}^n |\Phi(a_i)^T y^k|)y^k = (1/ (c^k)^T K c^k) \sum_{i=1}^n \Phi(a_i)c^k_i.
\] This implies that we only need to update \(c^k\) at each iteration. From
\[
c^k_{i+1} = \text{sgn}((\Phi(a_i)^T x^{k+1}) = \text{sgn}((\Phi(a_i)^T y^k)
\]
\[
= \text{sgn}\left(\sum_{j=1}^n c^k_j K_{i,j}\right)
\]
we only require to compute
\[
c^k_{i+1} = \text{sgn}\left(\sum_{j=1}^n c^k_j K_{i,j}\right)
\]
at each iteration. From
\[
\|x^{k+1} - x^k\|_2^2 = 0 \iff (c^k - c^{k+1})^T K (c^k - c^{k+1}) = 0,
\]
we get the following termination criteria:
\[
(c^k - c^{k+1})^T K (c^k - c^{k+1}) = 0,
\]
resulting in Algorithm 1.

Algorithm 1 \(L_2\)-norm Kernel PCA

**Input:** data vectors \(a_i\), kernel matrix \(K_{i,j} = \Phi(a_i)^T \Phi(a_j)\),
starting sign vector \(c^0\)

**while** \((c^k - c^{k+1})^T K (c^k - c^{k+1}) > 0\) **do**

Compute \(c^k_{i+1} = \text{sgn}\left(\sum_{j=1}^n c^k_j K_{i,j}\right)\), \(i = 1,...,n\)

\(k \leftarrow k + 1\)

**end while**

After getting the final \(c^*\) from Algorithm 1 we can compute principal scores without explicit mapping \(\Phi(a_i)\).

For example, the principal component of \(i^{th}\) observation can be computed by
\[
\frac{\Phi(a_i)^T x^*}{\|x^*\|_2} = \frac{\Phi(a_i)^T y^*}{\|y^*\|_2}
\]
\[
= \sum_{i=1}^n \frac{\Phi(a_i)^T \Phi(a_j)c^*_j}{\sqrt{\sum_{i=1}^n \sum_{j=1}^n \Phi(a_i)^T \Phi(a_j)c^*_i c^*_j}}
\]
\[
= \frac{\sum_{i=1}^n K_{i,j}c^*_j}{\sqrt{\sum_{i=1}^n \sum_{j=1}^n K_{i,j}c^*_i c^*_j}}.
\]

We can also proceed to find more principal components without explicit mapping \(\Phi(a_i)\). As computing a loading vector and principal components only require the kernel matrix \(K\), we only need to update the kernel matrix \(K\) each time a new loading vector is found. We can update the kernel matrix without explicit mapping \(\Phi(a_i)\) by
\[
\tilde{K}_{i,j} = \left(\Phi(a_i) - \frac{\Phi(a_i)^T x^*}{\|x^*\|_2^2} x^*\right)^T \left(\Phi(a_j) - \frac{\Phi(a_j)^T x^*}{\|x^*\|_2^2} x^*\right)
\]
\[
= \Phi(a_i)^T \Phi(a_j) - \frac{\Phi(a_i)^T x^* \Phi(a_j)^T x^*}{\|x^*\|_2^2}
\]
\[
= \frac{\sum_{k=1}^n K_{i,k}c^*_k}{\left(\sum_{i=1}^n \sum_{j=1}^n K_{i,j}c^*_i c^*_j\right)}.
\]

5 Convergence Analysis

In this section, we provide a convergence analysis of Algorithm 1. We first prove finite convergence, and then provide a rate of convergence analysis.

Before proving the convergence of the algorithm, we first show that the sequence \(|\|x^k\|_2^2\|\) is non-increasing.

**Lemma 1.** We have \(|\|x^k\|_2^2^ - \|z^k\|_2^2 \leq |\|x^k\|_2^2 - \|x^k\|_2^2| = 0\).

**Proof.** Inequality \(|\|z^k\|_2^2 - \|x^k\|_2^2| \leq \|x^k\|_2^2 - \|y^k\|_2^2|\) follows from
\[
|\|x^k\|_2^2 - \|z^k\|_2^2| = \|x^k\|_2^2 - \frac{1}{\|y^k\|_2^2}
\]
\[
= \|x^k\|_2^2 - \frac{(\|y^k\|_2^2)}{\|y^k\|_2^2}
\]
\[
= \|x^k\|_2^2 - (\|y^k\|_2^2) \geq 0.
\]

Here, the second equality is from [6] and the last inequality holds by Cauchy-Schwarz inequality where the equality
holds if $x^k$ is a scalar multiple of $y^k$.
Next, we have
\[
\sum_{i=1}^{n} \left| \Phi(a_i)^T z^k \right| = \sum_{i=1}^{n} \left| \Phi(a_i)^T (y^k) \right| (y^k)^T (y^k) = 1
\]
\[
= \frac{1}{(y^k)^T (y^k)} \sum_{i=1}^{n} \left| \Phi(a_i)^T y^k \right|
\]
\[
= \sum_{i=1}^{n} \Phi(a_i)^T (\sum_{j=1}^{n} \Phi(a_j) c_j^k) = \frac{1}{(y^k)^T (y^k)} \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_i) \Phi(a_j) c_i^k c_j^k
\]
\[
= \frac{1}{(y^k)^T (y^k)} \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_i) T \Phi(a_j) c_i^k c_j^k
\]
\[
= \frac{1}{(y^k)^T (y^k)} \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_i) \Phi(a_j) c_i^k c_j^k
\]
\[
= \frac{1}{(y^k)^T (y^k)} \sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_i) T \Phi(a_j) c_i^k c_j^k
\]
\[
\geq 1. \quad (11)
\]
Finally, $\|x^{k+1}\|_2 \leq \|z^k\|_2$ follows from
\[
\|x^{k+1}\|_2 = (1/\sum_{i=1}^{n} \left| \Phi(a_i)^T z^k \right|)^2 \|z^k\|_2 \leq \|z^k\|_2^2.
\]

**Lemma 2.** If $\|x^k\|_2 = \|x^{k+1}\|_2$, we have $x^k = \frac{y^k}{\|y^k\|^2}_2$, $y^k = x^k$, and $x^k = x^{k+1}$.

**Proof.** From Lemma 1, we have $\|z_k\|_2 = \|x^k\|_2$. Then, from (10), $x^k$ is a scalar multiple of $y^k$. Assuming $x^k = ry^k$, $r = \frac{1}{\|y^k\|^2}_2$ follows from (6) resulting in
\[
x^k = \frac{y^k}{\|y^k\|^2}_2.
\]
We can show $y^k = \frac{y^k}{\|y^k\|^2}_2$ in the same manner. As a result, $x^k = z^k$ holds by (6). From $x^k = z^k$, we have
\[
x^k+1 = (1/\sum_{i=1}^{n} \left| \Phi(a_i)^T z^k \right|) z^k = (1/\sum_{i=1}^{n} \left| \Phi(a_i)^T x^k \right|) x^k = x^k.
\]

**Theorem 1.** The sequence $\{x^k\}$ converges in a finite number of steps.

**Proof.** Suppose the sequence $\{x^k\}$ does not converge. As vector $x^k$ is solely determined by $c^k \in \{-1, +1\}^n$, the number of possible $x^k$ is finite. Therefore, if the sequence $\{x^k\}$ does not converge, then some vectors appear more than once in the sequence $\{x^k\}$.
Without loss of generality, let $x^l = x^{l+m}$. By Lemma 1, we have
\[
\|x^{l+m}\|_2 = \|x^l\|_2 \geq \|x^{l+1}\|_2 \geq \ldots \geq \|x^1\|_2.
\]
forcing us to have $\|x^1\|_2 = \|x^2\|_2 = \ldots = \|x^{l+m}\|_2$. Now, by Lemma 2, $x^1 = x^2 = \ldots = x^{l+m}$ must hold, which contradicts the assumption that the sequence $\{x^k\}$ does not converge. In other words, the algorithm stops at iteration $l+1$. Therefore, the sequence $\{x^k\}$ generated by Algorithm 1 converges in a finite number of steps.

Next, we prove that the sequence $\{x^k\}$ generated by Algorithm 1 converges with a linear rate.

**Theorem 2.** Let Algorithm 1 start from $x^0$ and terminate with $x^*$ at iteration $k^* + 1$. Then we have $\|x^k\|_2 - \|x^*\|_2 \leq \rho^k (\|x^0\|_2 - \|x^*\|_2)$ where $\rho < 1$ for all $k < k^*$.

**Proof.** From (5), we have
\[
\|x^k\|_2 = (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) |z_{k-1}|_2.
\]
Since $|z_{k-1}|_2 \leq \|x^{k-1}\|_2$ by Lemma 1, we have
\[
\|x^k\|_2 \leq (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^{k-1}\|_2. \quad (12)
\]
Now, we show
\[
\|x^k\|_2 - \|x^*\|_2 \leq (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^{k-1}\|_2 - \|x^*\|_2.
\]
Inequality (12) implies
\[
\|x^k\|_2 - \|x^*\|_2 \leq (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^{k-1}\|_2 - \|x^*\|_2. \quad (13)
\]
Subtracting $\|x^*\|_2$ gives,
\[
\|x^k\|_2 - \|x^*\|_2 \leq (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^{k-1}\|_2 - \|x^*\|_2.
\]
Here,
\[
(1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^{k-1}\|_2 - \|x^*\|_2
\]
\[
= (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) (\|x^{k-1}\|_2 - (\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) \|x^*\|_2)
\]
\[
\leq (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{k-1}\|) (\|x^{k-1}\|_2 - \|x^*\|_2)
\]
follows resulting in (13) where the last inequality follows from (11).
In order to show the convergence rate statement, we observe that from (13) we have
\[
\|x^k\|_2 - \|x^*\|_2 \leq \|x^0\|_2 - \|x^*\|_2. \quad (14)
\]
If $(1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{j-1}\|) = 1$, we have
\[
x^j = (1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{j-1}\|) z_{j-1} = z_{j-1}.
\]
This indicates that $x^{j-1}$ and $y^j$ lie on the same hyperplane $H^{j-1}$ resulting in $y^{j-1} = y^j$ and $x^j = x^{j+1}$. Therefore, unless $j - 1 \geq k^* - 1$, we have
\[
(1/\sum_{i=1}^{n} \|\Phi(a_i)^T z_{j-1}\|) < 1.
\]
Let $c \in \{-1, 1\}^n$ and
\[
\rho(c) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \Phi(a_i)^T \Phi(a_j)c_ic_j}{\sum_{i=1}^{n} |\sum_{j=1}^{n} \Phi(a_i)^T \Phi(a_j)c_ic_j|}.
\]
Assuming $\rho = \max_c \rho(c)$ subject to $\rho(c) < 1$, we have
\[
(1/\sum_{i=1}^{n} |\Phi(a_i)^T x_k|) = \left(\frac{\|y^k\|_2^2}{\sum_{i=1}^{n} |\Phi(a_i)^T y^k|}\right) = \rho(x^k) \leq \rho < 1
\]
for $k = 0, \ldots, k^* - 2$. Here, the first equality follows from [3] and the second one follows [7]. Together with [14], this gives the desired result.

As shown by Theorem 2, no matter where the algorithm starts, its objective value decreases with a linear rate. Now, we show that we can construct a local optimal solution to (1) by scaling the output of Algorithm 1.

**Theorem 3.** Let the output of Algorithm 1 be $x^*$. Then, $\hat{x}^* = x^*/\|x^*\|_2$ is a local optimal solution to (1).

**Proof.** It is easy to see that $\hat{x}^*$ is feasible. Since $x^*$ is the output of Algorithm 1, $y^* = \frac{x^*}{\|x^*\|_2}$ holds from Lemma 2.

Next, consider $L(\lambda, x) = \sum_{i=1}^{n} |\Phi(a_i)^T x| - \lambda(\|x\|_2^2 - 1)$. From
\[
\nabla_x L(\lambda, x) = \sum_{i=1}^{n} \text{sgn}(\Phi(a_i)^T x) \Phi(a_i) - 2\lambda x,
\]
we have
\[
\nabla_x L(\lambda, \hat{x}^*) = \sum_{i=1}^{n} \text{sgn}(\Phi(a_i)^T \hat{x}^*) \Phi(a_i) - 2\lambda \hat{x}^* = \sum_{i=1}^{n} \text{sgn}(\Phi(a_i)^T x^*) \Phi(a_i) - 2\lambda \hat{x}^* = y^* - 2\lambda \hat{x}^* = \frac{x^*}{\|x^*\|_2^2} - 2\lambda \hat{x}^* = \left(\frac{1}{\|x^*\|_2^2} - 2\lambda\right) \hat{x}^*.\]

Therefore, with $\lambda^* = \frac{1}{2\|x^*\|_2^2}$, we have $\nabla_x L(\lambda^*, \hat{x}^*) = 0$ meaning that $(\lambda^*, \hat{x}^*)$ satisfies the first-order necessary conditions. Moreover, from $\nabla_x L(\lambda^*, \hat{x}^*) = -2\lambda^* I < 0$, the second-order sufficient condition is also satisfied. Therefore, $\hat{x}^*$ is a local optimal solution to (1). \hfill \square

## 6 Experimental Results

To measure the effectiveness of $L_2$-norm kernel PCA, we test Algorithm 1 together with the $L_2$-norm kernel PCA algorithm [2] on datasets having influential outliers to see how well each algorithm extracts principal components in such a noisy setting. We also investigate how such principal components obtained in a noisy setting can be used for outlier detection. Lastly, we present a runtime comparison to $L_2$-norm kernel PCA.

### 6.1 Robust Extraction of PCs

To measure robustness, we first run the kernel PCA algorithms on datasets having outliers (noisy datasets) to obtain the loading vectors. Next, we compare how much variation in outlier-excluded datasets (normal datasets) is explained by loading vectors obtained from noisy datasets. For this purpose, we generate synthetic datasets that have influential outliers, so that the loading vectors obtained by applying kernel PCA algorithms on noisy and normal datasets are much different from each other.

For synthetic data generation, we first construct a $1000 \times 50$ data matrix with the rank of 10 following the data generation procedure in [10]. While the largest size in [10] is $300 \times 50$, we choose the size of $1000 \times 50$ to consider larger datasets. After that, we corrupt $r\%$ of observations by adding random noise resulting in a noisy dataset. We call the normal dataset the noisy dataset without the corrupted observations. For each value of $r \in \{5, 10, 15, 20, 25, 30\}$, we generate 10 instances.

Let $K, \hat{K}$ denote the kernel matrix of normal and noisy datasets, respectively. Let $x_1, \ldots, x_p$ be the $p$ loading vectors obtained by running an $L_2$-norm kernel PCA algorithm on $K$, and let $\hat{x}_1, \ldots, \hat{x}_p$ be the loading vectors obtained by running either $L_1$-norm or $L_2$-norm kernel PCA algorithm on $\hat{K}$. Then, assuming that the normal dataset is standardized,
\[
\sum_{j=1}^{p} \sum_{i=1}^{n} \left(\Phi(a_i)^T \hat{x}_j\right)^2 = \sum_{j=1}^{p} \hat{x}_j^T K \hat{x}_j
\]
represents the amount of variation of the normal dataset explained by the $p$ loading vectors obtained by running $L_2$-norm kernel PCA algorithm on $K$, and let $\hat{x}_1, \ldots, \hat{x}_p$ be the loading vectors obtained by running either $L_1$-norm or $L_2$-norm kernel PCA algorithm on $\hat{K}$. Then, assuming that the normal dataset is standardized,
\[
\sum_{j=1}^{p} \sum_{i=1}^{n} \left(\Phi(a_i)^T \hat{x}_j\right)^2 = \sum_{j=1}^{p} x_j^T K \hat{x}_j
\]
represents the amount of variation of the normal dataset explained by the $p$ loading vectors obtained by running $L_1$-norm kernel PCA algorithm on $K$.

This metric represents how well the loading vectors obtained from a noisy dataset explain the variation of the normal dataset with respect to the $L_2$-norm. In other words, it measures the robustness of kernel PCA algorithms in the presence of outliers. If the value is closer to $I$, the algorithm is more robust to outliers. Using this metric, we compare the robustness of the $L_1$-norm and $L_2$-norm kernel PCA algorithms. For each value of $r$, we compute the metrics with $p = 4$ for the ten datasets and average them. We arbitrarily choose $p = 4$ since the result is consistent regardless of the choice of $p$. Figure 2 shows the results for the linear kernel and Figure 3 shows the results for the Gaussian kernel with width $\sigma$ varying from 10 to 25.

As shown in Figure 2, when the linear kernel is used, $L_1$-norm kernel PCA outperforms $L_2$-norm kernel PCA for all values of $r$ demonstrating its robustness with respect to the presence of outliers. While the loading vectors from $L_2$-norm kernel PCA explain around 90% of the variation in the normal datasets, those from $L_1$-norm kernel PCA explain nearly 95% of it. As the percentage of corrupted
observations (r%) increases, the total explained variation is decreasing for both \( L_1 \)-norm and \( L_2 \)-norm kernel PCA but the gaps between two models remain the same.

When the Gaussian kernel is used, the results are slightly different depending on the values of width parameter \( \sigma \) as shown in Figure 3. If \( \sigma \) is large, the influences of outliers are easily captured in the kernel matrix so that similar patterns appear as in the case of the linear kernel. On the other hand, if \( \sigma \) and \( r \) are small, the effects of outliers located far away from the normal datasets are relatively small, and thus performing \( L_2 \)-norm kernel PCA with or without outliers gives almost the same results. As a result, \( L_2 \)-norm kernel PCA has a higher total explained variation. This makes sense in such cases where the influences of outliers are not significant because our metric uses the \( L_2 \)-norm for measuring explained variation of the normal datasets, and the variance with respect to the \( L_2 \)-norm is maximized in \( L_2 \)-norm kernel PCA. However, even with small \( \sigma \), \( L_1 \)-norm kernel PCA starts to outperform as \( r \) grows.

### 6.2 Outlier Detection

\( L_2 \)-norm PCA has been shown to be effective for anomaly detection [11]. The idea is to extract loading vectors using datasets consisting of only normal samples, and to use these loading vectors for developing a detection model. Specifically, the boundary of normal samples is derived from the loading vectors and used to discriminate normal and abnormal samples.

We extend this principle to outlier detection, i.e. its unsupervised counterpart. In the outlier detection setting, sample labels are not given when the model is built. Therefore, it is not possible to build a detection model solely based on normal samples. Given this context, we apply \( L_1 \)-norm kernel PCA on the entire dataset (with outliers) to extract loading vectors and use them to characterize the boundary. Since \( L_1 \)-norm kernel PCA is robust to extracting loading vectors as shown in Section 6.1, we expect that these loading vectors would better construct the boundary of normal samples. We compare the performance of this algorithm to \( L_2 \)-norm kernel PCA based model and other outlier detection models [12] [13].

#### 6.2.1 A Toy Example

Before presenting experimental results, we first illustrate the advantage of using \( L_1 \)-norm PCA for outlier detection using a two-dimensional toy example. Figure 4 shows the distribution of normal samples and outliers in the toy example. In the figure, normal samples are distributed forming a linear pattern and outliers are distributed exhibiting two different patterns. For example, two rightmost red points are outliers due to their scales while they follow the linear pattern. On the other hand, six red points located in the middle are outliers since they do not follow the pattern.

In this example, if the first loading vector exactly matches the linear pattern, outliers can be easily detected using two principal components. The two rightmost outliers will be detected due to large first principal components and the six middle outliers will be detected due to large second principal components. However, it is hard for the first loading vector to match the linear pattern due to outliers. In this setting, \( L_1 \)-norm PCA with the first loading vector less deviating from the linear pattern as shown in Figure 4 can be useful compared to \( L_2 \)-norm PCA.

Figure 5 shows the PCA results using both \( L_1 \)-norm PCA and \( L_2 \)-norm PCA on the toy example. In the figure, the x-axis (y-axis) represents the first (second) principal component divided by its sample standard deviation. As shown in the figure, the two outliers having large scales are easily separated in both cases. However, while the six outliers can be clearly discriminated by the second principal components obtained by \( L_1 \)-norm PCA, there exists some overlaps between normal samples and outliers in the ranges of second principal components obtained by \( L_2 \)-norm PCA. As a result, some outliers appear closer to the origin than the farthest normal samples making it hard to construct the normal boundary. On the other hand, all normal samples are clearly separated from the outliers in the \( L_1 \)-norm PCA result as shown in the left plot of Figure 5, demonstrating an advantage of using the \( L_1 \)-norm PCA based model in outlier detection.

#### 6.2.2 Real-world Datasets

For outlier detection, we use datasets from the UCI Machine Learning Repository [14] and ODDS Library [15], see Table 1.

| Data set   | # samples | # features | # outliers |
|------------|-----------|------------|------------|
| WBC        | 378       | 30         | 21 (7.6%)  |
| Ionosphere | 351       | 33         | 126 (36%)  |
| BreastW    | 683       | 9          | 239 (35%)  |
| Cardio     | 1831      | 21         | 176 (9.6%) |
| Musk       | 3062      | 166        | 97 (3.2%)  |
| Mnist      | 7603      | 100        | 700 (9.2%) |

In this experiment, we basically use the same detection rule as in [11] where it is applied for anomaly detection. Assuming that \( Y \in \mathbb{R}^{n \times d} \) represents the projected data points based on \( p \) loading vectors and \( \lambda_j \) is the variance of \( j^{th} \) principal component, we use the following rule to detect outliers:

```markdown
Y^T \lambda_j \cdot Y < \text{threshold}
```
Figure 3. Robust Extraction of PCs (Gaussian Kernel with σ ranging from 10 to 25)

Figure 4. The toy example

Classify $i^{th}$ sample as an outlier, if
$$\sum_{\{j: \lambda_j \geq \alpha\}} \frac{Y_i^2_j}{\lambda_j} > c. \quad (15)$$

The metric appearing on the left-hand side of (15) represents the squared Euclidean distance to the origin in the scaled reduced principal component space consisting of principal components whose variance is greater than or equal to α. So, our decision rule can be understood as a circular boundary as illustrated in Figure 5. Since we are assuming the outlier detection setting, sample labels are unknown at the stage of building a model. Therefore, we compute precision and recall with varying $c$ and measure the performance of a model using AUC under the precision-recall curve. We compare AUC of the $L_1$-norm PCA based model to those of the $L_2$-norm PCA based model and two other popular outlier detection models, Local Outlier Factor (LOF) [12] and Isolation Forest (iForest) [13].

Considering that principal components having small variance contain minor information, we choose principal components whose sample variance is greater than or equal to α. We select the largest α such that
$$0.8 \times \sum_{j=1}^{d} \lambda_j \leq \sum_{\{j: \lambda_j \geq \alpha\}} \lambda_j,$$
where $d$ is the number of features. In other words, we select top principal components which explain more than 80% of the variation in a dataset. We develop models using two kernel functions: Linear and Gaussian. For the width parameter $\sigma$ of the Gaussian kernel, we set it to $d$. On the other hand, we set the number of nearest neighbors to 10 in LOF, and set the number of trees, the size of subsample and the number of rounds to 100, 256, 10, respectively in iForest since these parameter values are the most widely used. Table 2 shows AUC of the six different models: the $L_1$-norm linear kernel based model, the $L_2$-norm linear kernel based model, the $L_1$-norm Gaussian kernel based model, the $L_2$-norm Gaussian kernel based model, LOF, iForest.

Table 2 shows that detection models built upon $L_1$-norm
kernel PCA are better than those build upon $L_2$-norm kernel PCA. The numbers in bold present the highest AUC cases (there can be several similar top performances). If outliers are not influential, PCA outcomes of $L_1$-norm kernel PCA and $L_2$-norm kernel PCA are almost the same. Therefore, the two models give very similar AUC values as in Breastw and Mnist. On the other hand, if outliers are influential, $L_1$-norm kernel PCA extracts loading vectors in a more robust manner, and thereby better characterize the boundary of normal samples. In such a case, the detection model from $L_1$-norm kernel PCA outperforms that from $L_2$-norm kernel PCA as in WBC, Ionosphere, Cardio, and Musk. Moreover, introducing Gaussian kernel improves the performance of detection models for two datasets: Breastw and Cardio. This result is important since it justifies the use of a kernel in $L_1$-norm PCA. Comparison to two popular outlier detection models shows that the $L_1$-norm kernel PCA based detection models often outperform in detecting outliers. They are always better than LOF, and outperform iForest in four out of the six datasets. They especially work very well when $d$ is large demonstrating that they should be used for density-based models for high-dimensional datasets.

6.3 Runtime Comparison

Lastly, we compare the runtime of $L_1$-norm and $L_2$-norm kernel PCA using the six real-world datasets presented in Table 1. For each dataset, we apply $L_1$-norm and $L_2$-norm kernel PCA and measure the time taken to get all principal components.

As shown in Table 3 the runtime grows more than linearly as the number of samples $n$ increases. This is because the time to compute the kernel matrix increases quadratically with respect to $n$. Although $L_1$-norm kernel PCA always takes slightly longer than $L_2$-norm kernel PCA, the differences do not seem significant. This result is consistent with the linear convergence rate result of Algorithm 1. In conclusion, $L_1$-norm kernel PCA obtains more robust results with a comparable computational time.

7 Conclusion

In this work, we propose a model and algorithm for the variance maximization version of $L_1$-norm kernel PCA. For this non-convex and non-smooth problem, we first reformulate it so that it can be geometrically understood. Based on geometric understandings, we derive an algorithm under which the kernel trick is applicable. After that, we provide a convergence analysis. We prove that our algorithm converges in a finite number of steps as well as that the rate of convergence is linear. Moreover, we prove that the output of our algorithm satisfies local optimality conditions. Computational experiments demonstrate the robustness of $L_1$-norm kernel PCA and the runtime comparison shows that $L_1$-norm kernel PCA obtains robust results with a comparable computational time. Its application to outlier detection yields great results. The $L_1$-norm kernel PCA based models are not only better than the $L_2$-norm kernel PCA based models but also produce competitive results compared to LOF and iForest, and outperform when the dimension of a dataset is high.
TABLE 3
Runtime Comparison

| Data Set | L1-norm (Linear) | L2-norm (Linear) | L1-norm (Gaussian) | L2-norm (Gaussian) |
|----------|-----------------|-----------------|-----------------|-----------------|
| WBC      | 0.29            | 0.19            | 0.34            | 0.24            |
| Ionosphere | 0.12          | 0.08            | 0.17            | 0.13            |
| Breastw  | 0.72            | 0.68            | 0.88            | 0.85            |
| Cardio   | 13.94           | 11.91           | 14.34           | 13.21           |
| Musk     | 112.93          | 46.39           | 118.34          | 50.28           |
| Mnist    | 881.20          | 795.38          | 824.53          | 744.64          |

REFERENCES

[1] I. Jolliffe, Principal Component Analysis. Wiley Online Library, 2002.
[2] B. Schölkopf, A. Smola, and K.-R. Müller, “Kernel Principal Component Analysis,” in International Conference on Artificial Neural Networks, 1997, pp. 583–588.
[3] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados, “Optimal Algorithms for $L_1$-subspace Signal Processing,” IEEE Transactions on Signal Processing, vol. 62, no. 19, pp. 5046–5058, 2014.
[4] M. McCoy and J. A. Tropp, “Two Proposals for Robust PCA using Semidefinite Programming,” Electronic Journal of Statistics, vol. 5, pp. 1123–1160, 2011.
[5] G. N. Karystinos and A. P. Liavas, “Efficient Computation of the Binary Vector That Maximizes a Rank-Deficient Quadratic Form,” IEEE Transactions on Information Theory, vol. 56, no. 7, pp. 3581–3593, 2010.
[6] Y. Nesterov, “Semidefinite Relaxation and Nonconvex Quadratic Optimization,” Optimization Methods and Software, vol. 9, no. 1-3, pp. 141–160, 1998.
[7] Y. W. Park, “Optimization for Regression, PCA, and SVM: Optimality and Scalability,” Ph.D. dissertation, Northwestern University, 2015.
[8] N. Kwak, “Principal Component Analysis based on $L_1$-norm Maximization,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 9, pp. 1672–1680, 2008.
[9] F. Nie, H. Huang, C. Ding, D. Luo, and H. Wang, “Robust Principal Component Analysis with Non-Greedy $L_1$-Norm Maximization,” in Proceedings of the International Joint Conference on Artificial Intelligence, vol. 22, no. 1, 2011, pp. 1433–1438.
[10] Y. W. Park and D. Klabjan, “Iteratively Reweighted Least Squares Algorithms for $L_1$-Norm Principal Component Analysis,” in IEEE International Conference on Data Mining, 2016, pp. 430–438.
[11] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, “A Novel Anomaly Detection Scheme Based on Principal Component Classifier,” in IEEE International Conference on Data Mining, 2003, pp. 172–179.
[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in ACM SIGMOD Record, vol. 29, no. 2, 2000, pp. 93–104.
[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in IEEE International Conference on Data Mining, 2008, pp. 413–422.
[14] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: [http://archive.ics.uci.edu/ml]
[15] S. Rayana, “ODDS library,” 2016. [Online]. Available: [http://odds.cs.stonybrook.edu]

Cheolmin Kim is a Ph.D student in Industrial Engineering and Management Science at Northwestern University. He received his B.S in Industrial Engineering and B.A in Economics from Seoul National University. His research interests are at the interaction of optimization and machine learning. He is interested in designing a new machine learning model, developing an training algorithm and analyzing the efficiency of the algorithm from an optimization perspective.

Diego Klabjan is a professor at Northwestern University, Department of Industrial Engineering and Management Sciences. He is also Founding Director, Master of Science in Analytics. After obtaining his doctorate from the School of Industrial and Systems Engineering of the Georgia Institute of Technology in 1999 in Algorithms, Combinatorics, and Optimization, in the same year he joined the University of Illinois at Urbana-Champaign. In 2007 he became an associate professor at Northwestern and in 2012 he was promoted to a full professor. His research is focused on machine learning, deep learning and analytics with concentration in finance, transportation, sport, and bioinformatics. Professor Klabjan has led projects with large companies such as Intel, Baxter, Allstate, AbbVie, FedEx Express, General Motors, United Continental, and many others, and he is also assisting numerous start-ups with their analytics needs. He is also a founder of Opex Analytics LLC.