Principal Component Analysis by L_p-norm Maximization

Nojun Kwak, *Member, IEEE*

Abstract

This paper proposes several principal component analysis methods based on L_p-norm optimization techniques. In doing so, the objective function is defined using the L_p-norm with an arbitrary p value, and the gradient of the objective function is computed on the basis of the fact that the number of training samples is finite. In the first part, an easier problem of extracting only one feature is dealt with. In this case, principal components are searched for either by a gradient ascent method or by a Lagrangian multiplier method. When more than one feature is needed, features can be extracted one by one greedily, based on the proposed method. Secondly, a more difficult problem is tackled that simultaneously extracts more than one feature. The proposed methods are shown to find a local optimal solution. In addition, they are easy to implement without significantly increasing computational complexity. Lastly, the proposed methods are applied to several datasets with different values of p, and their performances are compared with those of conventional PCA methods.

Index Terms

PCA-Lp, Lp-norm, optimization, principal component analysis, gradient.

I. Introduction

Principal component analysis (PCA) [1], a well-known tool for data analysis and pattern recognition, is extensively used for visualizing data and reducing its dimensionality. PCA searches a set of projections to maximize the variance of the projected data or to minimize the reconstruction error. These projections constitute a low-dimensional linear subspace that enables us to effectively capture the data structure in the original input space.

Conventional PCA, which is based on the L_2-norm (L2-PCA), is optimal in the sense of mean squared reconstruction error when the data is distributed according to a Gaussian

N. Kwak is an associate professor at the Department of Transdisciplinary Studies, Seoul National University, KOREA.

distribution. However, it is prone to outliers because outliers with large norms dominate the objective function owing to the use of the L_2-norm. To overcome this problem, objective functions based on alternative norms have been explored [2] – [10].

In [5], [6] and [7], each component of the reconstruction error was assumed to follow a Laplacian distribution instead of a Gaussian and L_1-norm PCA (L1-PCA), which tries to minimize the L_1 reconstruction error, was proposed. In order to obtain a solution of L1-PCA, a heuristic estimate for the general L_1 problem was used in [5], whereas methods based on the weighted median and convex programming were proposed in [6] and [7]. Despite the robustness of the proposed L1-PCA methods, they are computationally expensive because of the application of linear or quadratic programming. More importantly, they are not invariant to rotations because the L_1-norm is measured in the input space.

In [8], the same L_1-norm optimization problem was tackled by successively decreasing the dimension of the feature space. Unlike other methods, this method finds a global optimal solution. However, it is time-consuming because it needs to solve a number of linear programs.

In [4], Ding *et al.* proposed R1-PCA, which combines the merits of conventional PCA and L1-PCA. Unlike L1-PCA, it is rotation-invariant, and it successfully suppresses the effect of outliers as L1-PCA does. However, this method is highly dependent on the dimension m of a subspace to be found. For example, the projection vector obtained when $m = 1$ may not be in a subspace obtained when $m = 2$. Moreover, because it is an iterative algorithm based on the successive use of the power method [11], it consumes considerable time to achieve convergence, especially when the input dimension is high.

Unlike the above methods, which try to minimize the reconstruction error in the input space, a greedy method that maximizes the L_1-norm in the feature space is presented in [9] in order to achieve robust and rotation-invariant PCA. To distinguish it from L1-PCA, which minimizes the L1-norm-based reconstruction error, the method proposed in [9] was named PCA-L1. The PCA-L1 algorithm is intuitive, simple, and easy to implement, and it was shown to find a locally maximal solution. Studies on its tensor and supervised versions are described in [12], [13], and [14]. It has also been extended to a non-greedy version in [10] in order to simultaneously find m projections by iterative application of singular value decomposition (SVD) [11].

In this paper, PCA-L1 [9] is generalized to PCA-Lp, which maximizes L_p-norm-based dispersion in the feature space with an arbitrary $p > 0$ value, and it is shown that conventional L2-PCA and PCA-L1 are special cases of the proposed PCA-Lp with appropriate p values

($p = 2$ and $p = 1$, respectively). The proposed methods are realized using a gradient ascent method or a Lagrangian multiplier method, and they are shown to find a local optimal solution. In addition, as an extension of [10], a non-greedy version of PCA-Lp is proposed, which simultaneously finds m principal components that maximize the L_p-norm. Like PCA-L1, PCA-Lp is intuitive, simple, and easy to implement.

The remainder of this paper is organized as follows. In Section II, the problem is formulated. New algorithms for the L_p-norm optimization problem with $m = 1$ and $m > 1$ are presented and their local optimality is proven in Section III and IV respectively. The proposed methods are applied to several pattern recognition problems, and their performances are compared with those of other conventional PCA methods in Section V. Finally, the conclusions are stated in Section VI.

II. Problem Formulation

Let $X = [x_1, \cdots, x_N] \in \Re^{d\times N}$ be a data matrix, where d and N denote the dimension and number of data respectively. Without loss of generality, we can assume that the data has zero mean, i.e., $\sum_{i=1}^{N} x_i = 0$.

Conventional L2-PCA can be formulated as finding $m(< d)$ orthonormal projection vectors $W \in \Re^{d\times m}, W^T W = I_m$, such that the following total scatter or variance after projection is maximized.

$$F_2(W) = \frac{1}{2}\sum_{i=1}^{N} ||W^T x_i||_2^2 = \frac{1}{2} tr(W^T S W). \tag{1}$$

Here, I_m is the $m \times m$ identity matrix, $S = XX^T$ is the scatter matrix of X, $||\cdot||_2$ is the L_2-norm of a vector, and $tr(\cdot)$ is a trace operator on a square matrix. The global optimal solution of (1) can be found by solving $SW = W\Lambda$, where Λ is a diagonal matrix containing eigenvalues of S.

Because the total scatter (1) is dominated by samples with large norms, the following alternative objective function based on L_1-norm (instead of L_2-norm) dispersion was introduced in [9].

$$F_1(W) = \sum_{i=1}^{N} ||W^T x_i||_1 = \sum_{i=1}^{N}\sum_{j=1}^{m} |w_j^T x_i|. \tag{2}$$

Here, w_j is the j-th column of W and $||\cdot||_1$ denotes the L_1-norm of a vector. In [9], the problem was reduced to a special case of $m = 1$, and a local optimal solution was found using PCA-L1. Then, a greedy algorithm was introduced to find a series of projection vectors by taking the reconstruction errors as new data to which PCA-L1 was applied successively. On the other hand, in [10], for a given m, (2) was directly maximized by iterative application of SVD.

In the previous approaches, the norm of a vector $W^T x_i$ was defined using an integer value ($p = 2$ or $p = 1$). However, it can also be defined using an arbitrary non-integer p value. With the use of an arbitrary L_p-norm, PCA is expected to better fit more datasets with non-Gaussian distributions.

The problem formulation in this paper is as follows. Consider the following L_p-norm ($p > 0$) maximization problem with the constraint $W^T W = I_m$.

$$F_p(W) = \frac{1}{p} \sum_{i=1}^{N} ||W^T x_i||_p^p = \frac{1}{p} \sum_{i=1}^{N} \sum_{j=1}^{m} |w_j^T x_i|^p. \tag{3}$$

Here, $W = [w_1, \cdots, w_m] \in \Re^{d \times m}$ is a projection matrix. Note that the objective function (3) is identical to (1) when $p = 2$. Likewise, when $p = 1$, it coincides with (2).

The above optimization problem is difficult to solve when $m > 1$. Therefore, for now, we restrict our attention to only those cases in which $m = 1$. In case more than one projection vectors are required, i.e., $m > 1$, a greedy algorithm can be used, as in [9]. On the other hand, in Section IV, (3) is tackled directly and a non-greedy solution is obtained for general cases of m.

III. Algorithm: PCA-Lp ($m = 1$)

A. Solution - Gradient Ascent

Let $m = 1$. Then, (3) becomes

$$\begin{aligned} w^\star = \operatorname*{argmax}_w F_p(w) = \operatorname*{argmax}_w \frac{1}{p} \sum_{i=1}^{N} |w^T x_i|^p \\ \text{subject to } w^T w = 1. \end{aligned} \tag{4}$$

This optimization problem can be solved by taking the gradient of $F_p(w)$ with respect to w. However, the gradient may not be well defined owing to the absolute value operation. This technical difficulty can be alleviated by introducing the sign function as follows:

$$s(a) = \begin{cases} 1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0. \end{cases}$$

With the sign function, $F_p(w)$ in (4) can be rewritten as

$$F_p(w) = \frac{1}{p}\sum_{i=1}^{N}[s(a_i)a_i]^p, \tag{5}$$

where $a_i \triangleq w^T x_i$.

Taking the gradient of $F_p(w)$ with respect to w, we get

$$\begin{aligned}\nabla_w = \frac{dF_p(w)}{dw} &= \sum_{i=1}^{N}\frac{dF_p(w)}{da_i}\frac{da_i}{dw}\\ &= \sum_{i=1}^{N}[s(a_i)a_i]^{p-1}[s'(a_i)a_i + s(a_i)]x_i\\ &= \sum_{i=1}^{N} s'(a_i)s^{p-1}(a_i)a_i^p x_i + \sum_{i=1}^{N} s^p(a_i)a_i^{p-1}x_i\\ &= 2\sum_{i=1}^{N}\delta(a_i)s^{p-1}(a_i)a_i^p x_i + \sum_{i=1}^{N} s(a_i)|a_i|^{p-1}x_i,\end{aligned} \tag{6}$$

where $\delta(\cdot)$ in the last equality is the Dirac delta function.

The first term vanishes if $a_i(= w^T x_i) \neq 0$ for all x_i ($i = 1, \cdots, N$), and we get

$$\nabla_w = \sum_{i=1}^{N} s(w^T x_i)|w^T x_i|^{p-1}x_i. \tag{7}$$

Moreover, when $p > 1$, even for the singular points (w's) where $w^T x_i = 0$ for some x_i's, the first term vanishes and the gradient is well defined, which becomes (7).

An exceptional case is when $p \leq 1$. In this case, the gradient is not well defined for w's if there exist some x_i's where $w^T x_i = 0$. However, because of a finite number of samples, this singularity condition can be technically avoided by slightly moving w, the operating point of the gradient ∇_w.

Then, the steepest gradient method can be applied to obtain the projection that maximizes the objective function (4). The overall optimization procedure is represented as PCA-Lp(G) in the following:

Algorithm 1-1. PCA-Lp(G) (Input: X, p, Output: $w^\star$)

1) Initialization: $t \leftarrow 0$. Set $w(0)$ such that $||w(0)||_2 = 1$.
2) Singularity check (applies only if $p \leq 1$)
   - If $\exists i$, such that $w^T(t)x_i = 0$, $w(t) \leftarrow (w(t) + \delta)/||w(t) + \delta||_2$. Here, δ is a small random vector.

3) Computation of gradient ∇_w using (7).
4) Gradient search: $w(t+1) \leftarrow w(t) + \alpha \nabla_w$, where α is the learning rate.
5) Normalization:
   - $t \leftarrow t+1$.
   - $w(t) \leftarrow w(t)/||w(t)||_2$.
6) Convergence check
   - If $||w(t) - w(t-1)||_2 > \epsilon$, goto Step 2.
   - Else, $w^\star \leftarrow w(t)$. Stop iteration.

Because PCA-Lp(G) is based on the steepest gradient method, it finds a local optimal solution that is dependent on the initial projection vector $w(0)$. Therefore, the choice of $w(0)$ can be critical to the performance of PCA-Lp(G). Naturally, a good candidate for $w(0)$ may be the solution of conventional L2-PCA. Another choice can be the direction of the sample with the largest norm. For simplicity, in this paper, unless specified explicitly, the initial projection vector $w(0)$ is set to the direction of the sample with the largest norm.

For the steepest gradient method, when the learning rate α is very high, it is difficult to ensure convergence; on the other hand, very small values of α result in slow convergence. In this paper, unless specified explicitly, α is set to $0.1/N$, where N is the number of training samples.

Note that the singularity check (Step 2) is only necessary when $p \leq 1$. Further, note that the normalization step (Step 5) ensures the constraint $||w||_2 = 1$.

With the constraint $||w||_2 = 1$, the search space of the gradient is restricted only to the orthogonal direction of the weight vector w, which can be easily obtained by

$$\begin{aligned} \nabla_w^\perp &= \nabla_w - w(w^T \nabla_w) = (I_d - ww^T)\nabla_w \\ &= (I_d - ww^T) \sum_{i=1}^{N} s(w^T x_i)|w^T x_i|^{p-1} x_i. \end{aligned} \tag{8}$$

Therefore, in the above PCA-Lp(G) procedure, $\nabla_w^\perp$ can be used instead of ∇_w in Step 4. In this case, $\nabla_w^\perp$ can be directly used to check for convergence. Henceforth, $\nabla_w^\perp$ will be referred to as the *gradient orthogonal*.

B. *Interpretation - force*

The gradient orthogonal $\nabla_w^\perp$ can be considered as the net force exerted on a bar w. Rewriting the gradient orthogonal, we get

$$\nabla_w^\perp = \sum_{i=1}^N c_i v_i = \sum_{i=1}^N f_i, \tag{9}$$

where $c_i = s(w^T x_i)|w^T x_i|^{p-1}$ and $v_i = x_i - w(w^T x_i)$. In this interpretation, each sample x_i exerts an orthogonal directional force $f_i = c_i v_i$ on w, as shown in Fig. 1. In the figure, the origin o is fixed, and w is free to move according to the exerted forces until the net force becomes 0. The length a_i in the figure corresponds to $|w^T x_i|$. If the inner product $w^T x_i$ is positive (e.g., x_1), the point x_i pulls the bar w towards itself. On the other hand, if $w^T x_i$ is negative (e.g., x_2 and x_3), x_i pulls $-w$ towards itself, which is equivalent to pushing w in the opposite direction. The magnitude of the force f_i due to the sample point x_i is $|f_i| = a_i^{p-1}|v_i|$.

Consider conventional PCA with $p = 2$. In this case, the magnitude of the force due to x_i becomes $|f_i| = a_i|v_i|$. If the magnitude a_i of the inner product between w and x_i is large and $|v_i|$, the distance of x_i from the bar w, is large, the effect of the sample point x_i is increased.

On the other hand, when $p = 1$, $|f_i| = |v_i|$. In this case, a_i does not contribute to the force, and the only important factor is the distance of the sample point from the bar w. Considering that outliers are normally far from the origin and therefore, have large a_i and $|v_i|$ with high probability, we can expect PCA-L1 to be more robust to outliers than conventional PCA-L2.

In addition, if $p < 1$, a_i negatively affects the force, and the effect of outliers is further reduced. In this case, $|f_i| = \frac{|v_i|}{a_i^{1-p}} = (\frac{|v_i|}{a_i})^{1-p}|v_i|^p = |\tan\theta|^{1-p}|v_i|^p$, where θ is the angle between w and x_i. If $x_i \perp w$, the point exerts an infinite force on the bar w. For this reason, we have included the singularity check (Step 2) in the PCA-Lp(G) algorithm.

In the extreme case of $p \to 0$, $|f_i| \to |v_i|/a_i = |\tan\theta|$. However, note that the problem (4) is not defined at all for $p = 0$ because 0^0 is not well defined.

C. *Alternative Solution - Lagrangian*

Here, instead of a gradient ascent method, an alternative method based on the Lagrangian is derived to solve (4).

Consider the Lagrangian of the constrained optimization problem (4):

$$L(w, \lambda) = F_p(w) + \lambda(w^T w - 1). \tag{10}$$

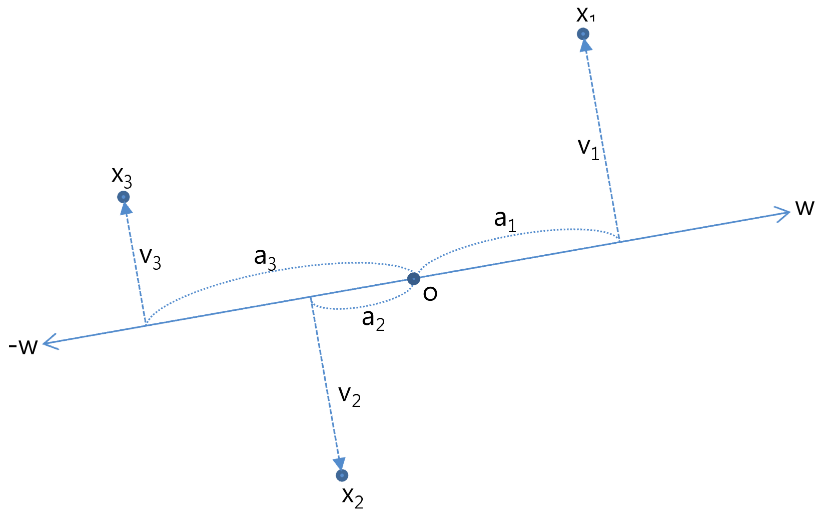


Fig. 1. Illustration of gradient as force

Then, the necessary condition for the optimal solution of (4) can be obtained by setting the derivative of the Lagrangian to zero, i.e.,

$$\frac{dL(w,\lambda)}{dw} = \nabla_w + \lambda w = 0. \tag{11}$$

This condition indicates that w and ∇_w should be parallel (i.e., $w \parallel \nabla_w$) at the optimal w. From the constraint $||w||_2 = 1$, w can be directly updated as $w \leftarrow \frac{\nabla_w}{||\nabla_w||_2}$.

Indeed, (11) is true not only for the maximum points but also for the minimum points. However, in the vicinity of a minimum point, the gradient direction diverges from the minimum point. On the other hand, in the neighborhood of a maximum point, the gradient converges to the maximum point. Therefore, we can expect iterative application of the update rule to find a maximum point, but not a minimum point.

Before we move on to the introduction of an alternative PCA-Lp algorithm, we show that the iterative method $w \leftarrow \frac{\nabla_w}{||\nabla_w||_2}$ is indeed non-decreasing for $p \geq 1$.

Theorem 1: Let $||w||_2 = 1$ and $w' = \frac{\nabla_w}{||\nabla_w||_2}$. Then, for $p \geq 1$, $F_p(w') \geq F_p(w)$.

Proof See Appendix.

Using this property, we can obtain an alternative algorithm that solves the PCA-Lp problem as follows:

Algorithm 1-2. PCA-Lp(L) (Input: X, p**, Output:** $w^\star$**)**

```
1) Initialization: $t \leftarrow 0$. Set $w(0)$ such that $||w(0)||_2 = 1$.
2) Singularity check (applies only if $p \leq 1$)
    • If $\exists i$, such that $w^T(t)x_i = 0$, $w(t) \leftarrow (w(t) + \delta)/||w(t) + \delta||_2$. Here, $\delta$ is a small random vector.
3) Computation of gradient $\nabla_w$ using (7).
4) Projection vector update:
    • $t \leftarrow t + 1$.
    • $w(t) \leftarrow \nabla_w/||\nabla_w||_2$.
5) Convergence check
    • If $||w(t) - w(t-1)||_2 > \epsilon$, goto Step 2.
    • Else, $w^\star \leftarrow w(t)$. Stop iteration.
```

Because the objective function (4) is upper bounded and every iteration of PCA-Lp(L) increases the objective function according to *Theorem 1*, we can show that PCA-Lp(L) finds a local maximal solution as PCA-Lp(G).

The solution of PCA-Lp(L) depends on the initial projection vector $w(0)$. However, in the next subsection, we will show that the performance of PCA-Lp(L) is not as significantly affected by the initialization as that of PCA-Lp(G).

Note that in this alternative version, Step 4 replaces the steepest gradient search (Step 4 and 5 in PCA-Lp(G)). Indeed, PCA-Lp(L) is equivalent to PCA-Lp(G) when $\alpha = \infty$. Both PCA-Lp(G) and PCA-Lp(L) may fall into a local maximum point. However, because PCA-Lp(L) does not make use of a low learning rate α, it is normally faster than PCA-Lp(G) as long as the level set of $F_p(w)$ is smooth. Although PCA-Lp(L) was not shown to be non-decreasing for $p < 1$ in *Theorem 1*, it works well, even for small values of p, in practice.

Like PCA-L1, both PCA-Lp(G) and PCA-Lp(L) have computational complexity $\mathcal{O}(Nd) \times n_{it}$, where n_{it} is the number of iterations for convergence. Clearly, the number of iterations does not depend on the dimension d of the input space, but on the number of samples N. Therefore, PCA-Lp can be applied to problems with a large number input variables without significantly increasing the computational complexity.

D. Comparison of the two algorithms

To investigate the local maximality and convergence properties of PCA-Lp(G) and PCA-Lp(L), we consider the following example.

Ex. 1 Consider the zero-mean two-dimensional data matrix

$$X = \begin{bmatrix} -0.8 & 0.2 & 1.2 & -3.8 & 3.2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}$$

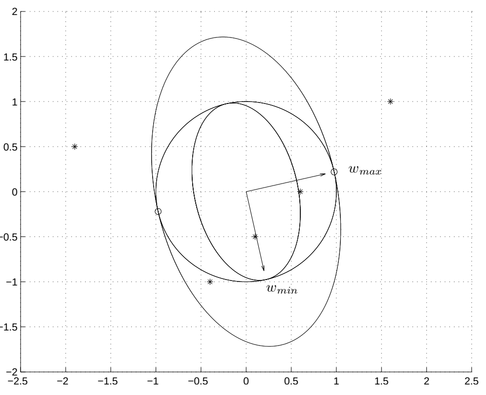
with 5 samples.

Figure 2 shows level sets of $F_p(w)$ for this example with different values of p. In the figures, the inner contour corresponds to the level set $\{w : F_p(w) = c_{min}\}$, whereas the outer contour corresponds to the level set $\{w : F_p(w) = c_{max}\}$. Here, c_{min} and c_{max} are the minimum and maximum of $F_p(w)$ with the constraint $||w||_2 = 1$. Because $F_p(cw) = c^p F_p(w)$ for $c \geq 0$, the inner and outer contours are similar. The circles between the contours correspond to points of $||w||_2 = 1$; the minimum projection vectors w_{min} and the maximum projection vectors w_{max} are also shown in the figures. The points denoted by $\circ$ on the circles are the local maximal points. As expected, at these points (w's), the gradient direction ∇_w, which is orthogonal to the tangent of the contour, is parallel to w. Further, the scaled versions of five data points are shown in the figures by $*$.

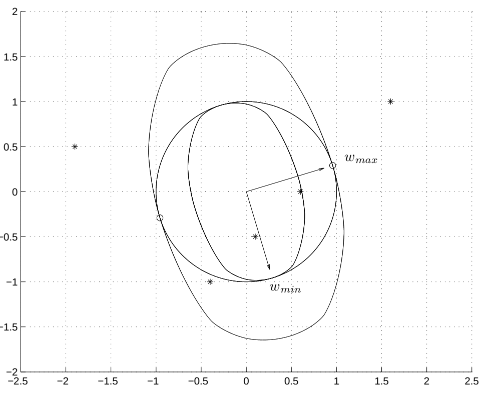
Note that when $p = 2$ where the level set is an ellipsoid, the local maximum corresponds to the global maximum. Likewise, when $p = 1.5$, there is no local maximal point. However, for other cases, there exist local but non-global maximal points. Further, as p decreases, the level sets become less smooth.

For this dataset, we compared the performance of PCA-Lp(G) and PCA-Lp(L) with the initial projection vector being $w(0) = [\cos(\theta), \sin(\theta)]^T$, where θ was varied from $0°$ to $180°$ in steps of $0.1°$. In PCA-Lp(G), the learning factor α was set to 0.02. The stopping tolerance ϵ and the maximum number of iterations were 10^{-10} and $1,000$ respectively for both algorithms.

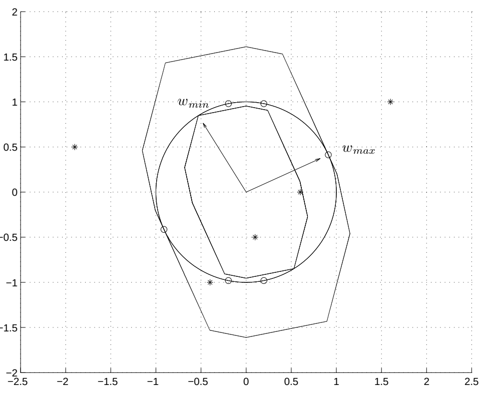
Table I compares PCA-Lp(G) and PCA-Lp(L) in terms of the success rates of finding the global maximum and the average numbers of iterations for various values of p. An experiment is considered as successful if the final value of $F_p(w)$ is the global maximum. The success rate is obtained by dividing the number of successful experiments by the total number of experiments (1,800). In Table I, the numbers in parentheses indicate the standard deviations of the experiments.



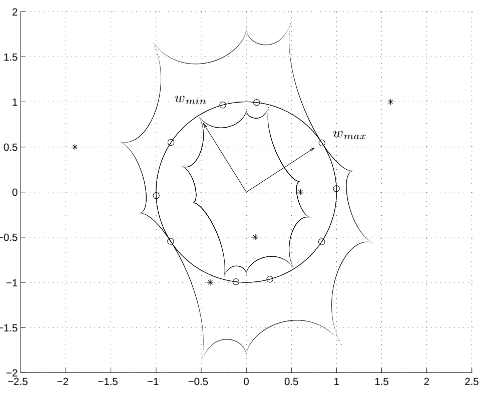
(a) $p = 2.0$



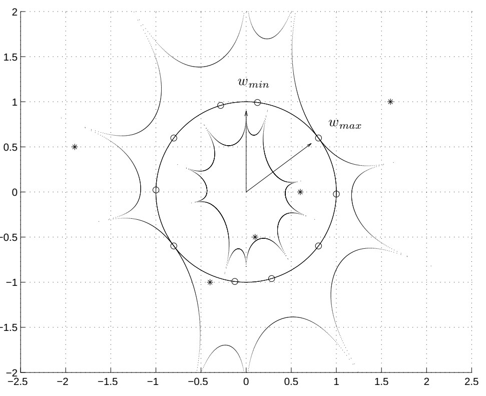
(b) $p = 1.5$



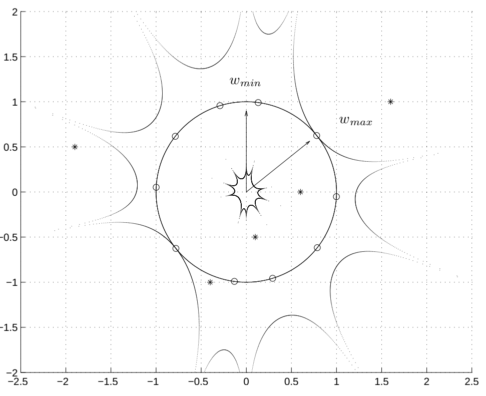
(c) $p = 1.0$



(d) $p = 0.5$



(e) $p = 0.25$



(f) $p = 0.1$

Fig. 2. Level sets of $F_p(w)$ for Ex. 1. The points denoted by $\circ$ on the circle are the local maximal points. The global maximum and minimum points are also indicated. The points denoted as $*$ are the five data points with a scaling of 0.5.

TABLE I

COMPARISON OF PCA-LP(G) AND PCA-LP(L) FOR VARIOUS VALUES OF p. TOTAL 1,800 INITIAL PROJECTION VECTORS $w(0)$'S WERE TESTED TO CHECK WHETHER THE CORRESPONDING FINAL PROJECTION VECTORS CONSTITUTE THE GLOBAL OPTIMAL SOLUTION. THE SUCCESS RATE IS THE PERCENTAGE THAT RESULTED IN THE GLOBAL OPTIMAL SOLUTION. THE AVERAGE NUMBERS OF ITERATIONS AS WELL AS THE STANDARD DEVIATIONS ARE LISTED.

p	PCA-Lp(G)		PCA-Lp(L)	
	success rate (%)	number of iterations	success rate (%)	number of iterations
0.1	39.33	57.82 (30.62)	22.06	1000 (0)
0.25	37.33	59.39 (28.19)	100	177.46 (14.03)
0.5	35.78	69.28 (27.23)	100	39.64 (4.45)
1	74.00	135.51 (19.90)	74.00	2.59 (0.80)
1.5	100	113.38 (11.11)	100	19.51 (1.96)
2	100	80.40 (5.74)	100	21.82 (1.43)

In the table, we can see that PCA-Lp(L) is generally better in finding a global maximal point when p is not so small. In terms of the number of iterations, PCA-Lp(L) is faster than PCA-Lp(G) for an appropriately large p value. In particular, when $p = 1$, it takes less than 3 iterations on average to converge. PCA-Lp(L) requires so many iterations for convergence when p is small because the contour is very spiky for a small p value and the gradient direction changes abruptly near the spiky points (see Fig. 2(d), (e) and (f)).

The success rate of PCA-Lp(L) is exactly 100%, except when $p = 0.1$ and $p = 1$. When $p = 0.1$, the number of iterations indicates that the algorithm did not converge within the predefined maximum number of iterations. On the other hand, when $p = 1$, the average number of iterations is very small (2.59). Fig. 2(c) shows that the level set of $p = 1$ consists of only straight lines. Considering that the gradient is orthogonal to the tangent of the level set contour, the number of gradient directions is finite when $p = 1$. Hence, depending on the initial projection vector $w(0)$, PCA-Lp(L) with $p = 1$ has a good chance to be stuck to one of these gradient directions, which is a local optimal solution. Therefore, the success rate of PCA-Lp(L) when $p = 1$ is only 74% with a small number of iterations.

Unlike the case of $p = 1$, the other values of p do not contain straight lines in their level set contours in Fig. 2. Therefore, if $p \neq 1$, different directions of w result in different ∇_w, and the algorithm checks different w for its optimality as it iterates. If $p > 1$, the level set contours are smooth and PCA-Lp(L) gradually updates its projection vector w like PCA-Lp(G). In this case, as can be seen in Fig. 2(a) and (b), there are not many local optimal solutions, and both PCA-Lp(L) and PCA-Lp(G) have a good chance to find the global optimal solution regardless of the initial projection vector $w(0)$.

On the other hand, when $p < 1$, as can be seen in Fig. 2(d), (e), and (f), the level set contours are very spiky and the projection vectors at consecutive iterations (i.e., $w(t)$ and $w(t{+}1)$) differ radically. Hence, if $p < 1$, $w(0)$ does not affect the performance of PCA-Lp(L) significantly, whereas the performance of PCA-Lp(G) is more dependent on the initialization. Therefore, the success rate of PCA-Lp(G) is lower than that of PCA-Lp(L) when $p = 0.25$ and $p = 0.5$.

Combining the above analysis of both cases when $p > 1$ and $p < 1$, we can conjecture that if $p \neq 1$, PCA-Lp(L) has a good chance of finding a global optimal solution regardless of the initial projection vector $w(0)$.

As a conclusion from this simple example, we recommend that the readers use PCA-Lp(L) unless p is very small, in which case, convergence may be a problem.

E. Relationship with conventional PCAs

1) PCA-L2: Consider conventional PCA, which optimizes the objective function (4) with $p = 2$. Then, the gradient (7) becomes

$$\nabla_w = \sum_{i=1}^{N} x_i x_i^T w = Sw, \tag{12}$$

where $S = \sum_{i=1}^{N} x_i x_i^T = XX^T$ is the scatter matrix of X.

Conventional PCA-L2, which optimizes (1), is obtained by eigenvalue decomposition of S, and the optimal solution is the eigenvector corresponding to the largest eigenvalue.

In the following, we will show the equivalence of eigenvectors of S and the points where the gradient orthogonal $\nabla_w^\perp$ is zero.

Theorem 2: Let $p = 2$. In this case, for the eigenvectors w's of S, the gradient orthogonal $\nabla_w^\perp$ is zero. In addition, if $\nabla_w^\perp = 0$ at some point w, then w is an eigenvector of S.

Proof See Appendix.

Ex. 2 To visualize $\nabla_w^\perp = 0$ at the eigenvectors of S, consider the following example. Let $d = 2$ and $N = 100$. Suppose that each sample is generated according to a Gaussian distribution with zero mean and sample covariance matrix $S = [105, -43; -43, 35]$.

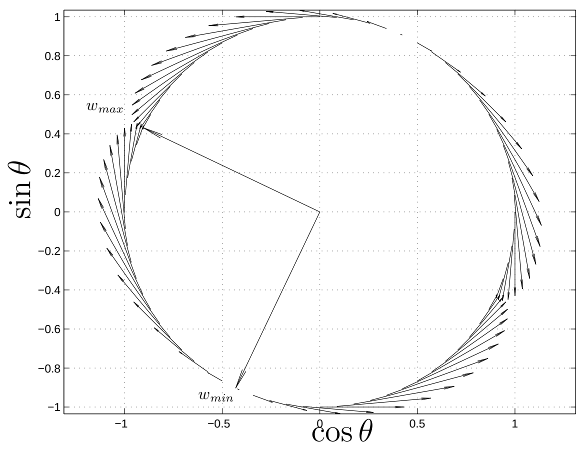


Fig. 3. 2-D example. An arrow starting from a point on the circle w indicates $\nabla_w^\perp$. As expected, $\nabla_w^\perp$'s computed at the two eigenvectors w_{max} and w_{min} are zero. Near w_{max}, $\nabla_w^\perp$ converges to w_{max}, while it diverges from w_{min}.

For this example, we set $w = [\cos\theta, \sin\theta]^T$ and computed $\nabla_w^\perp$ for various values of θ, as shown in Fig. 3. In the figure, the magnitude of $\nabla_w^\perp$ is scaled appropriately, and we can see that $\nabla_w^\perp$ is orthogonal to the corresponding w.

When we solve the eigenvalue decomposition problem on S, the resultant eigenvalues are $\lambda_1 = 125.30$ and $\lambda_2 = 14.75$, and the corresponding eigenvectors are $w_1 = [-0.903, 0.431]^T$ and $w_2 = [-0.431, -0.903]^T$, respectively. These two eigenvectors are shown in Fig. 3.

As expected, $\nabla_w^\perp$'s computed at the two eigenvectors were zero. Note that the directions of $\nabla_w^\perp$'s around w_1 are towards w_1 while those around w_2 are away from w_2.

Ex. 3 Let us consider another example with three-dimensional input space. Suppose that $d = 3$, $N = 100$, and each sample is generated according to a Gaussian distribution with zero mean and sample covariance matrix $S = [295, 64, -99; 64, 85, -7; -99, -7, 39]$.

The three eigenvectors of S are $w_1 = [-0.923, -0.234, 0.305]^T$, $w_2 = [0.153, -0.952, -0.266]^T$, and $w_3 = [0.353, -0.199, 0.914]^T$. To visualize $\nabla_w^\perp$, we parameterized w by the latitude ϕ and the longitude θ as $w = [\cos\phi\cos\theta, \cos\phi\sin\theta, \sin\phi]^T$. Then, we computed $\nabla_w^\perp$ at various points from $\theta \in \{-180°, \cdots, 180°\}$ and $\phi \in \{0°, \cdots, 90°\}$. The gradient orthogonals are shown in Fig. 4.

In the figure, the $\circ$ symbols denote the three eigenvectors where the leftmost one is w_1, rightmost is w_2 and top is w_3. From the figure, we can see that the gradient orthogonal near the $\circ$ symbols are very small in magnitude. In addition, we can also see that the gradient

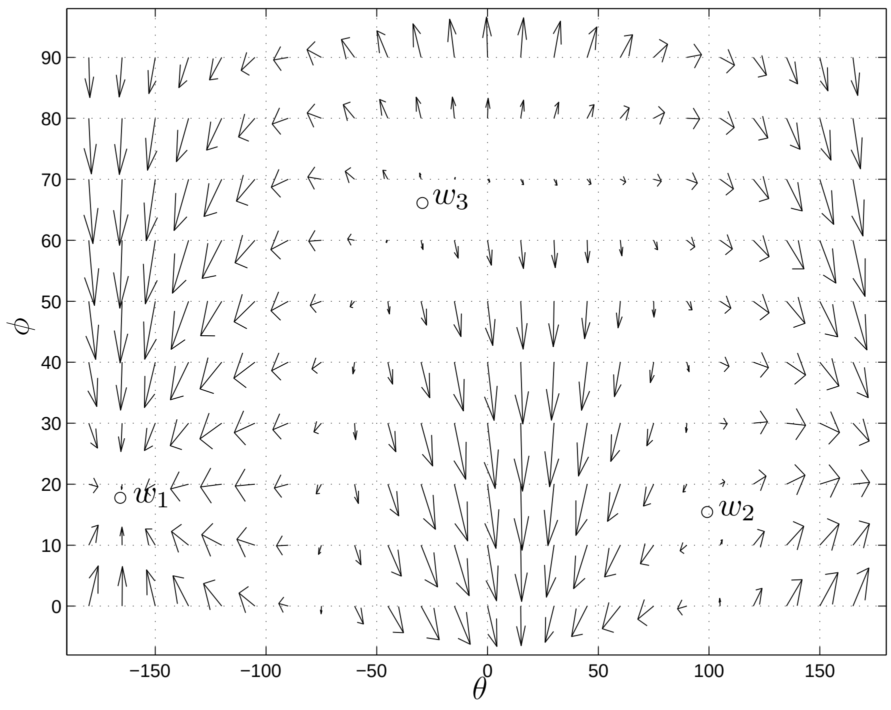


Fig. 4. 3-D example. The three eigenvectors are denoted as $\circ$ where the gradient orthogonals are zero. Near the global maximal point w_1, $\nabla_w^\perp$ converges to w_1, while near the global minimal point w_3, it diverges from w_3. w_2 acts as a saddle point.

orthogonals near w_1 are oriented towards w_1, while those near w_3 are oriented away from w_3. On the other hand, w_2 acts like a saddle point.

We have seen that the proposed gradient search method can be applied to implement the conventional PCA-L2 algorithm; however, because the conventional method directly finds the optimal solution using eigenvalue decomposition, the proposed method is slower and is not recommended when $p = 2$.

2) PCA-L1: Consider the case $p = 1$. In this case, the optimization problem (4) has already been solved in [9]. With $p = 1$, the gradient becomes $\nabla_w = \sum_{i=1}^N s(w^T x_i)x_i$. Replacing ∇_w with this value, we can easily check that PCA-L1 in [9] is exactly the same as PCA-Lp(L) with $p = 1$.

IV. Multiple feature extraction ($m > 1$)

A. Greedy solution

In this section, the PCA-Lp algorithm is extended to multiple feature extraction problems. The proposed method can be easily extended to extract an arbitrary number of features by applying the same procedure greedily to the remainder of the projected samples as follows:

Algorithm 2-1. Greedy PCA-Lp (Input: X, p, m, Output: $W^\star$)

```
1) Let $w_0 = 0$ and $X_0 = X$.
2) For $i = 1 \cdots m$
    a) Set $X_i = (I_d - w_{i-1}w_{i-1}^T)X_{i-1}$.
    b) Apply PCA-Lp(G) or PCA-Lp(L) on $X_i$. i.e., $w_i = \textit{PCA-Lp}(X_i, p)$
3) Output $W^\star = [w_1, \cdots, w_m]$.
```

It is easy to show that the above greedy algorithm provides orthogonal projection vectors and it is expected that these projections suboptimally maximize L_p dispersion. The merit of the greedy method is that the output projection vectors do not change with different values of m. However, the greedy algorithm does not provide an optimal solution to (3). Therefore, in the following, problem (3) is tackled directly and a local optimal solution is provided.

B. Non-greedy solution

Consider the constrained optimization problem (3). For a given $W = [w_1, \cdots, w_m] \in \Re^{d\times m}(m \le d)$, where w_i denotes the i-th projection vector, the gradient is

$$\nabla_W = \frac{dF_p(W)}{dW} = [\nabla_{w_1}, \cdots, \nabla_{w_m}]. \tag{13}$$

Here, ∇_{w_i} is given by (7).

The Lagrangian for the constrained optimization problem (3) can be introduced for the case of $m > 1$ as follows:

$$L(W, \Gamma_m) = F_p(W) + \Gamma_m : (W^T W - I_m). \tag{14}$$

Here, the Lagrange multiplier $\Gamma_m \in \Re^{m\times m}$ is a symmetric matrix and $A : B$ denotes the Frobenius inner product of two matrices A and B with the same dimension, i.e., $A : B = \sum_i \sum_j A_{ij}B_{ij} = tr(A^T B)$.

Then, the necessary condition for the optimal solution of (3) can be obtained by setting the derivative of the Lagrangian to zero, i.e.,

$$\frac{dL(W,\Gamma_m)}{dW} = \nabla_W + 2W\Gamma_m = 0. \tag{15}$$

However, unlike the previous case with a simple constraint on the norm ($||w||_2 = 1$), the orthonormality constraint ($W^TW = I_m$) is harder to meet and the iterative update $W \leftarrow \frac{\nabla_W}{||\nabla_W||}$ cannot be used since Γ_m is not a scalar. Instead, we try to make W as close as possible to ∇_W in each iteration by solving the following optimization problem:

$$\begin{aligned} W' = \underset{Q}{\operatorname{argmax}}\, G(Q) = \underset{Q}{\operatorname{argmax}}\, tr(Q^T\nabla_W) \\ \text{subject to } Q^TQ = I_m. \end{aligned} \tag{16}$$

In the following theorem, we show that the solution to the above optimization problem (16) always improves the objective function $F_p(W)$.

Theorem 3: Let W' be the solution to (16). Then, for $p \geq 1$, $F_p(W') \geq F_p(W)$.

Proof See Appendix.

Now, the solution to (16) can be obtained by the same procedure of non-greedy PCA-L1 described in [10].

Theorem 4: Suppose that the SVD of ∇_W is $\nabla_W = U\Lambda V^T$, where $U \in \Re^{d\times d}$ and $V \in \Re^{m\times m}$ are unitary matrices and $\Lambda \in \Re^{d\times m}$ is a matrix whose non-diagonal elements are identically zero. Then, the solution of (16) is $W' = U[I_m|\mathbf{0}]^TV^T$, where $\mathbf{0} \in \Re^{m\times(d-m)}$ is the zero matrix.

Proof See Appendix.

Finally, the non-greedy PCA-Lp is obtained as follows:

Algorithm 2-2. Non-greedy PCA-Lp (Input: X, p, m, Output: $W^\star$)

```
1) Initialization: $t \leftarrow 0$. Set $W(0)$ such that $W(0)^TW(0) = I_m$.
2) Computation of gradient $\nabla_W$ using (13).
3) SVD of $\nabla_W$: $\nabla_W = U\Lambda V^T$.
4) Projection matrix update:
    • $t \leftarrow t + 1$.
```

- $W(t) \leftarrow U[I_m|\mathbf{0}]^T V^T$.

5) Convergence check
   - If $||W(t) - W(t-1)||_F > \epsilon$, goto Step 2.
     Here, $||A||_F$ denotes the Frobenius norm of a matrix A which is defined as $||A||_F = \sqrt{A : A} = \sqrt{tr(A^T A)}$.
   - Else, $W^\star \leftarrow W(t)$. Stop iteration.

To differentiate the greedy and non-greedy versions of PCA-Lp, henceforth, the former and the latter will be denoted by G-PCA-Lp and NG-PCA-Lp, respectively. Note that NG-PCA-Lp can be regarded as an extension of NG-PCA-L1 [10] for an arbitrary p value.

V. Experimental Results

In this section, we applied the proposed PCA-Lp algorithms with various values of p to several pattern recognition problems and compared the performances with those of R1-PCA [4] and L2-PCA. The experimental settings of R1-PCA were exactly the same as those in [4]. The maximum number of iterations for R1-PCA was set to 50. For G-PCA-Lp, PCA-Lp(L) was used instead of PCA-Lp(G) to extract successive weight vectors. The stopping tolerance ϵ and the maximum number of iterations were 10^{-10} and 100, respectively, for PCA-Lp(L). In Step 5 of NG-PCA-Lp, the Frobenius norm was used with ϵ value of 10^{-10} and the maximum number of iterations was 100. In all the experiments, the initial projection vector of PCA-Lp(L) was set to the sample with the largest L2-norm, i.e., $\boldsymbol{w}(0) = \text{argmax}_{\boldsymbol{x}_i} ||\boldsymbol{x}_i||_2$. On the other hand, in NG-PCA-Lp, the initial projection matrix was set to the solution of L2-PCA.

A. UCI dataset

We applied PCA-Lp to several datasets in UCI machine learning repositories [15], and compared the classification rates with those of L2-PCA and R1-PCA.

Table II shows a brief summary of the datasets used in this paper. Most of these datasets have been previously used in [9].

1) Dataset 1 to 12: As in [9], for each dataset except “Letter” (dataset 13), we performed 10-fold cross validation (CV) 10 times and computed the average classification rate. Before training, each input variable in the training set was normalized to have zero mean and unit

TABLE II

UCI DATASETS USED IN THE EXPERIMENTS

Dataset ID	Data set	No. of variables (d)	No. of classes	No. of instances
1	Australian	14	2	690
2	Balance	4	3	625
3	Breast cancer	9	2	683
4	Dermatology	34	6	358
5	Heart disease	13	2	297
6	Ionosphere	33	2	351
7	Iris	4	3	150
8	Liver	6	2	345
9	Sonar	60	2	208
10	Vehicle	18	4	846
11	Waveform	21	3	4999
12	Yeast	8	10	1484
13	Letter	16	26	20000

variance. The variables in the test set were also normalized using the means and variances of the training set. As a classifier, one nearest neighborhood (1-NN) classifier was used.

Table III shows the classification rates of each dataset using one feature ($m = 1$) along with the corresponding standard deviations in the parentheses. For each dataset, the best classification rate is written in bold face. In the figure, we can see that the best PCA for different classification problems depends on the dataset used. However, as can be seen in the last row of the table, PCA-Lp with $p = 1.0$ and $p = 1.5$ is slightly better than L2-PCA and R1-PCA, on average. In this experiment, because $m = 1$, we can expect that the results of G-PCA-Lp and NG-PCA-Lp are exactly the same. However, they are slightly different in the table owing to the different initial projection vectors and stopping criteria. This phenomenon is more clear for $p = 0.5$, where a relatively large number of local optimal solutions exist (see Fig. 2).

To show the statistical significance of the performance difference between PCA-Lp and R1-PCA, two versions of one tailed Welch's T-test [16] were performed on Table III. The null (H_0) and alternative (H_A) hypotheses for each statistical test are as follows:

- Test 1: Best PCA-Lp vs. R1-PCA
  - H_0: The best classification rate of PCA-Lp methods and that of R1-PCA are the

TABLE III

CLASSIFICATION RATES OF UCI DATASETS ($m = 1$). FOR EACH DATASET, 10-FOLD CV WAS PERFORMED 10 TIMES. STANDARD DEVIATIONS ARE IN PARENTHESES. BOLD-FACED LETTERS ARE THE BEST CLASSIFICATION RATES.

	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
Dataset ID	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
1	77.14 (1.66)	77.84 (0.66)	77.14 (0.93)	**78.28 (1.63)**	77.84 (0.66)	77.14 (0.93)	75.01 (1.70)	75.72 (1.03)
2	53.25 (5.44)	**54.99 (4.23)**	52.38 (3.53)	53.52 (4.31)	54.85 (4.04)	52.99 (3.61)	51.09 (6.11)	49.97 (4.57)
3	**96.41 (0.52)**	96.25 (0.48)	96.00 (0.45)	**96.41 (0.52)**	96.25 (0.48)	96.00 (0.45)	95.93 (0.32)	96.05 (0.39)
4	51.54 (2.47)	**53.65 (1.91)**	52.84 (1.77)	52.17 (2.17)	**53.65 (1.91)**	52.84 (1.77)	52.43 (1.33)	52.90 (2.51)
5	59.94 (3.06)	66.40 (1.47)	**70.31 (0.93)**	62.15 (2.43)	66.40 (1.47)	**70.31 (0.93)**	69.92 (1.45)	70.03 (1.80)
6	73.37 (1.90)	72.93 (3.54)	72.83 (1.83)	72.66 (2.08)	**74.51 (1.86)**	72.93 (3.54)	73.43 (2.37)	72.83 (1.83)
7	88.73 (1.46)	88.87 (2.04)	91.47 (1.29)	88.07 (1.27)	88.87 (2.04)	**91.47 (1.29)**	90.33 (1.27)	89.47 (1.50)
8	68.83 (1.57)	69.26 (2.12)	69.74 (1.12)	68.95 (1.27)	69.26 (2.12)	69.74 (1.12)	**69.77 (1.91)**	68.86 (2.24)
9	51.30 (4.22)	54.47 (3.62)	55.24 (2.71)	53.89 (2.48)	54.47 (3.62)	**55.34 (2.90)**	53.46 (3.18)	52.02 (4.22)
10	35.82 (1.04)	37.25 (1.37)	**37.58 (1.40)**	36.17 (1.86)	37.25 (1.37)	**37.58 (1.40)**	36.18 (1.28)	36.44 (1.28)
11	52.52 (0.79)	52.42 (0.71)	52.34 (0.73)	52.59 (0.89)	52.42 (0.71)	52.34 (0.73)	**52.87 (0.72)**	52.86 (0.61)
12	32.92 (1.77)	32.35 (1.23)	32.46 (1.44)	33.15 (1.00)	32.35 (1.23)	32.46 (1.44)	**33.40 (1.22)**	32.72 (1.15)
average	61.81	63.06	63.36	62.33	63.18	**63.43**	62.81	62.48

same.

 – H_A: The best PCA-Lp outperforms R1-PCA

- Test 2: Worst PCA-Lp vs. R1-PCA
  – H_0: The worst classification rate of PCA-Lp methods and that of R1-PCA are the same.
  – H_A: The worst PCA-Lp is worse than R1-PCA.

The computed T-value, degree of freedom (DoF), and the corresponding 95% target value $T_{95\%}$ are shown in Table IV. For each test, if the T-value is greater than $T_{95\%}$, the null hypothesis H_0 is rejected with 95% confidence; thus, the alternative hypothesis H_A is adopted.

Loosely speaking, if the hypothesis H_0 is accepted, we can say that the performance of the two compared methods are not significantly different. On the other hand, if H_A is accepted, we can say that one is better than the other.

Test 1 shows that the best PCA-Lp is better than R1-PCA for seven datasets out of twelve, while Test 2 shows that R1-PCA is better than the worst PCA-Lp for only one dataset (Heart disease). For the remaining four datasets (datasets 4, 8, 11, and 12), the performances of PCA-Lp and R1-PCA are not significantly different.

Figure 5 shows the average correct classification rates of some of the datasets with various numbers of extracted features. In the parentheses in the legends, 'G' and 'N' denote G-

TABLE IV

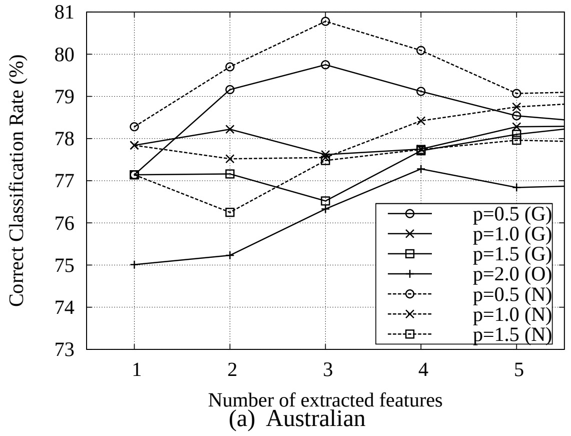
ONE TAILED T-TESTS ON UCI DATASETS BASED ON TABLE III ($m = 1$). TEST 1 IS THE COMPARISON BETWEEN THE BEST CLASSIFICATION RATE OF PCA-LP METHODS AND THAT OF R1-PCA, WHILE TEST 2 COMPARES THE WORST CLASSIFICATION RATE OF PCA-LP'S AND THAT OF R1-PCA. FOR BOTH TESTS, IF THE T-VALUE IS GREATER (SMALLER) THAN $T_{95\%}$, H_A (H_0) IS ACCEPTED.

	Test 1: Best PCA-Lp vs. R1-PCA				Test 2: Worst PCA-Lp vs. R1-PCA			
Dataset ID	T-value	DoF	$T_{95\%}$	Accepted Hypo.	T-value	DoF	$T_{95\%}$	Accepted Hypo.
1	4.20	15	1.75	H_A	1.12	15	1.75	H_0
2	2.55	18	1.73	H_A	-0.46	17	1.73	H_0
3	1.75	17	1.73	H_A	0.75	17	1.73	H_0
4	0.75	17	1.73	H_0	0.69	18	1.73	H_0
5	0.43	13	1.77	H_0	8.98	15	1.75	H_A
6	2.04	18	1.73	H_A	0.19	18	1.73	H_0
7	3.19	18	1.73	H_A	1.11	18	1.73	H_0
8	0.97	18	1.73	H_0	0.03	16	1.74	H_0
9	2.05	16	1.74	H_A	0.38	18	1.73	H_0
10	1.90	18	1.73	H_A	1.18	17	1.73	H_0
11	0.03	18	1.73	H_0	1.72	17	1.73	H_0
12	1.28	18	1.73	H_0	0.69	18	1.73	H_0

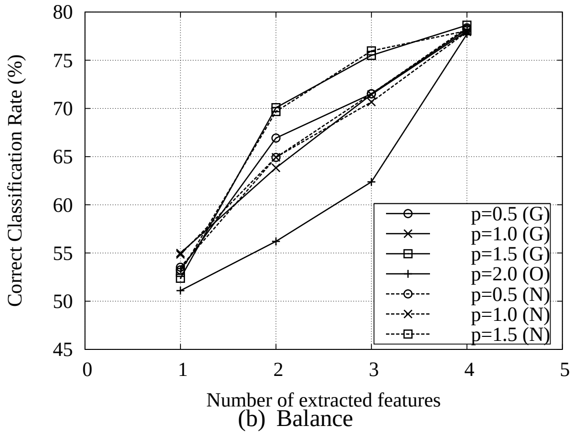
PCA-Lp and NG-PCA-Lp, respectively. L2-PCA is denoted by 'O' and it corresponds to $p = 2.0$.

For each of the UCI datasets, the number of extracted features m is varied from one to the dimension of the original input space d, and the average classification rates of $m = 1 \cdots d$ are reported in Table V. In the table, we can see that the classification rates of PCA-Lp with $p < 2$ are generally better than those of L2-PCA and R1-PCA though they are highly dependent on the specific dataset used. This phenomenon is clearly seen from the average classification rates of all the twelve datasets in the last row of the table. The performances of G-PCA-Lp and NG-PCA-Lp do not differ much in most cases. However, note that the features of NG-PCA-Lp change with the number of extracted features m, while those of G-PCA-Lp are independent of m.

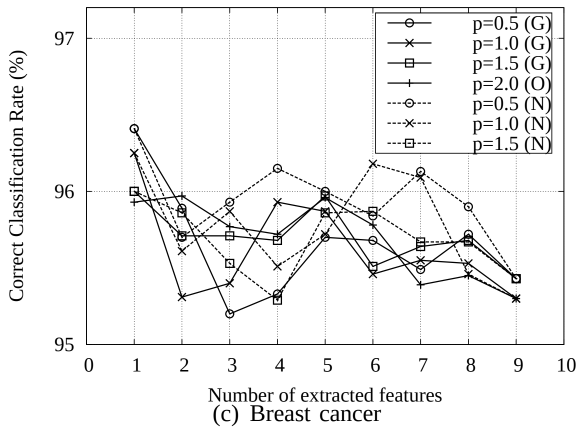
Table VI shows the average classification rates of the twelve datasets with a fixed number of extracted features m. The last row of the table also shows the averages of the best classification rates up to half of the total number of input variables ($d/2$). For all the cases, PCA-Lp with $p = 1.0$ and $p = 1.5$ outperforms L2-PCA and R1-PCA. PCA-Lp with $p = 0.5$ is also better than the conventional methods, except when $m = 1$. The averages of the best classification rates up to $d/2$ are highest with NG-PCA-Lp ($p = 1.5$), and both versions of PCA-Lp show



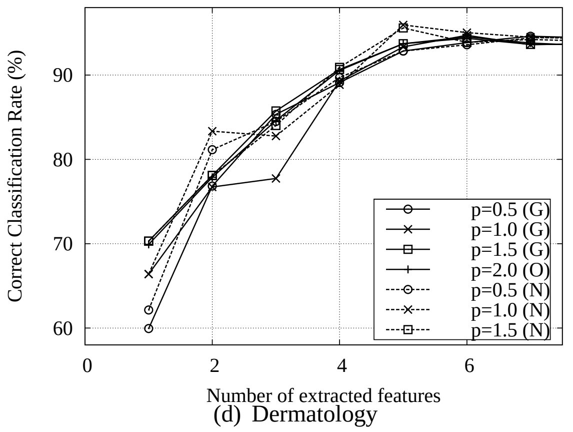
(a) Australian



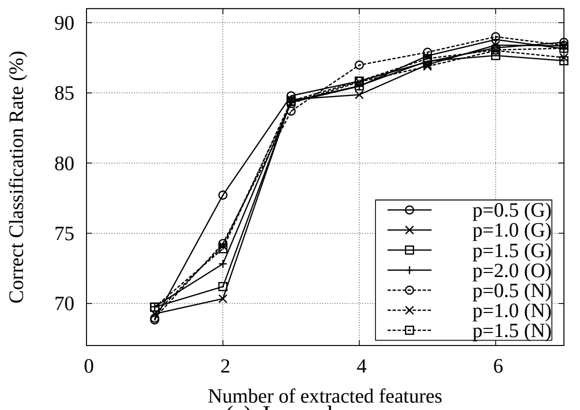
(b) Balance



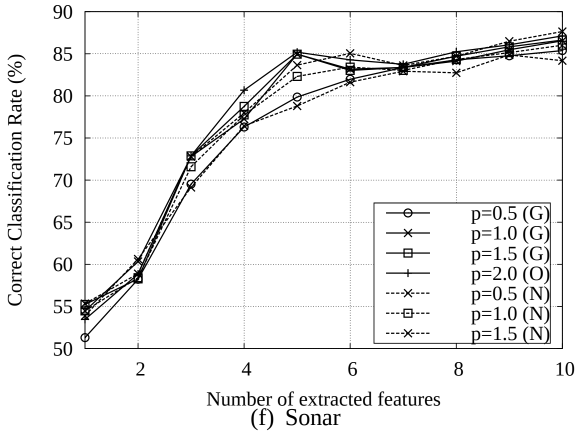
(c) Breast cancer



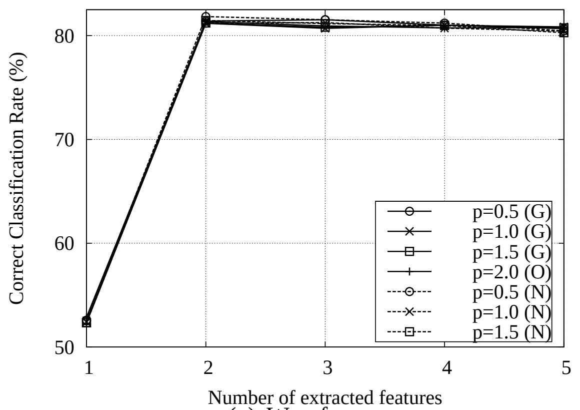
(d) Dermatology



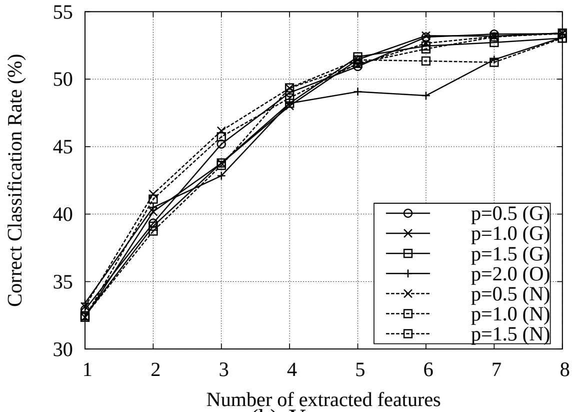
(e) Ionosphere



(f) Sonar



(g) Waveform



(h) Yeast

Fig. 5. Correct classification rates for some UCI datasets with various numbers of extracted features m.

TABLE V

AVERAGE CLASSIFICATION RATES OF UCI DATASETS (AVERAGE OF $m = 1 \cdots d$). THE LAST ROW IS THE AVERAGE OF THE TWELVE DATASETS. THE BEST CLASSIFICATION RATES ARE DENOTED IN BOLD FACE.

	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
Dataset ID	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
1	78.97	78.41	78.11	**79.53**	78.96	78.19	77.24	77.85
2	67.47	67.07	69.15	67.08	67.09	**69.17**	61.84	63.18
3	95.65	95.62	95.70	**95.94**	95.77	95.68	95.69	95.79
4	55.45	55.83	55.79	55.48	55.72	54.93	**55.96**	55.37
5	92.50	92.41	92.61	92.60	**92.81**	92.63	92.50	92.49
6	**76.27**	75.90	75.65	75.64	75.80	75.48	75.72	75.76
7	91.12	91.17	91.65	91.15	91.37	**91.67**	91.48	89.99
8	86.86	87.02	86.96	86.60	**87.09**	87.05	86.96	86.79
9	84.32	84.72	85.25	84.33	84.53	**85.30**	85.26	84.52
10	62.84	63.43	62.85	63.57	**63.70**	63.55	63.28	62.33
11	77.52	**77.65**	77.61	77.58	77.61	77.63	77.55	77.56
12	47.15	46.95	46.68	**47.55**	47.22	46.40	45.91	46.06
average	76.34	76.35	**76.50**	76.42	76.47	76.48	75.78	75.64

TABLE VI

AVERAGE CLASSIFICATION RATES OF UCI DATASETS WITH FIXED m ($m = 1 \cdots 3$). THE LAST ROW SHOWS THE AVERAGES OF THE BEST CLASSIFICATION RATES UP TO HALF OF THE TOTAL NUMBER OF INPUT VARIABLES (d).

	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
m	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
1	61.81	63.06	63.36	62.33	63.18	**63.43**	62.81	62.48
2	69.94	69.45	69.56	**70.17**	70.04	69.41	68.64	68.26
3	74.05	73.37	73.88	73.93	73.96	**74.44**	73.37	72.93
Best up to $m = d/2$	77.82	77.42	78.12	77.60	77.64	**78.22**	76.88	76.83

higher averages than L2-PCA and R1-PCA.

2) Dataset 13 (Letter): The ‘Letter’ dataset is composed of 26 classes of the English alphabet. Each class contains around 750 samples ranging from 734 to 813, a total of 20,000 samples. Because the number of samples is large with only 16 input variables, the samples are close to each other in the feature space and the 1-NN classifier does not work well. Actually, the correct classification rate of the 1-NN classifier applied to the features extracted by L2-PCA was only around 4 to 5%. The situation is almost the same for PCA-Lp and R1-PCA.

Therefore, we used the *nearest to the subspace classifier* to compare the performances of various PCAs as follows.

For each class, the first 300 examples were chosen to form the training sets, while the others were used as test data. For each class c, m principal components (PCs) $W_c \in \Re^{16\times m}$ and their mean vectors $m_c \in \Re^{16}$ were obtained using the corresponding training set with various versions of PCA. Then, once a test sample x is given, the reconstruction error is computed using the PCs and the mean vector corresponding to each class. Finally, the test sample is classified as the class with the minimum reconstruction error as follows.

$$\begin{aligned} c^\star &= \underset{c}{\operatorname{argmin}}\, ||x - \hat{x}_c|| \\ \text{where } \hat{x}_c &= m_c + W_c W_c^T (x - m_c). \end{aligned} \tag{17}$$

Table VII (a) shows the classification rates of various versions of PCA when the number of extracted features m varies from 1 to 7. Before applying PCAs, each input variable was normalized to have zero mean and unit variance. For each m, the best classification rate is denoted in bold face. In the table, we can see that the classification rates of both versions of PCA-Lp do not differ by more than 1%, except for the three cases of $p = 0.5$ with m = 5, 6, and 7. The best performances are obtained by R1-PCA, L2-PCA, and PCA-Lp with $p = 1.5$ depending on m. Their performance differences are less than 1% for all the cases. The performance of PCA-Lp with $p = 1.0$ is slightly worse than the three best ones. PCA-Lp with $p = 0.5$ works worst for all m. Because PCA-Lp with large p values ($p = 1.5$ and 2.0) is better than that with smaller p values ($p = 1.0$ and 0.5), it can be conjectured that the distribution of the dataset is quite close to a Gaussian distribution.

To show that PCA-Lp is more robust to outliers, the original training dataset was modified to contain spot noise. More specifically, after normalizing each input variable to have zero mean and unit variance, for each training sample, 1% of the input variables on average were randomly replaced with a value of 15 or -15. Then, the *nearest to the subspace classifier* is applied to classify the test data.

Table VII (b) shows the classification rates using the spot-noised training data. In the table, the best classification rates were obtained when PCA-Lp with $p = 0.5$ was used, except when $m = 1$, where PCA-Lp with $p = 1.0$ performed best. Comparing the performance of NG-PCA-Lp and G-PCA-Lp, G-PCA-Lp was better than NG-PCA-Lp, especially for large m. For this noised dataset, L2-PCA performed worst regardless of m. This phenomenon was expected as a number of outliers were deliberately introduced. R1-PCA worked relatively

TABLE VII

CLASSIFICATION RATES ON LETTER DATASET WITH VARIOUS m ($m = 1 \cdots 7$).

(a) Original dataset (without noise)

	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
m	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
1	61.56	62.45	62.70	60.68	62.45	62.70	62.80	**62.86**
2	66.47	67.83	**68.16**	66.97	67.39	67.99	67.46	67.67
3	71.81	72.39	**73.11**	71.39	72.52	73.03	72.87	73.07
4	75.35	77.14	77.90	74.82	77.08	77.66	78.01	**78.16**
5	76.83	79.25	79.67	75.48	79.34	**80.28**	79.38	80.08
6	77.84	79.98	**80.80**	76.02	80.33	80.68	80.48	80.73
7	78.68	80.42	**80.85**	74.78	80.02	80.63	80.69	80.61

(b) Dataset with 1% of spot noise

	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
m	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
1	57.26	**60.01**	59.01	59.02	**60.01**	59.01	56.00	59.66
2	63.34	61.95	61.01	**63.70**	61.48	61.18	56.52	62.45
3	**66.31**	66.09	63.76	64.71	64.83	63.75	58.32	63.39
4	**69.77**	67.16	64.23	66.54	66.58	64.70	59.75	66.26
5	**70.30**	68.02	63.43	67.47	66.52	63.39	57.59	65.84
6	**70.52**	67.54	62.65	67.68	65.00	62.20	56.44	66.53
7	**70.66**	66.63	61.17	66.27	64.97	60.12	54.51	65.12

well but not as well as PCA-Lp with $p = 0.5$.

B. USPS dataset

Here, we applied the proposed PCA-Lp with various values of p to the USPS dataset [17] and compared the classification rates with those of conventional L2-PCA and R1-PCA. The USPS dataset consists of 16×16 handwritten digits. As in the experiments on the "Letter" dataset described in the previous subsection, the *nearest to the subspace classifier* was used. Each of the ten digits contains 1,100 examples from which we chose the first 300 examples to form the training sets, while the other 800 were used as test data. For each digit $c \in \{0, \cdots, 9\}$, 30 principal components (PCs) $W_c \in \Re^{256\times 30}$ and their mean vectors $m_c \in \Re^{256}$ are obtained using the corresponding training set with various versions of PCA. Then, once a test sample x is given, the reconstruction error was computed using the PCs and the mean vector corresponding to each digit. Finally, the test sample is classified as the

TABLE VIII

CORRECT CLASSIFICATION RATES ON USPS DATA WITH GAUSSIAN NOISE

noise level	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
σ	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
0	94.88	95.52	95.51	94.62	95.38	**95.65**	95.46	95.30
10	94.90	95.53	95.52	94.62	95.37	**95.65**	95.45	95.30
20	94.97	95.57	95.55	94.58	95.35	**95.63**	95.46	95.30
30	94.92	95.53	95.46	94.58	95.37	**95.61**	95.40	95.25
40	94.81	95.48	95.43	94.46	95.23	**95.60**	95.33	95.15
50	94.70	95.25	95.31	94.55	95.28	**95.35**	95.21	95.21

TABLE IX

CORRECT CLASSIFICATION RATES ON USPS DATA WITH SALT AND PEPPER NOISE

noise level	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
n	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
0	94.88	95.52	95.51	94.62	95.38	**95.65**	95.46	95.30
0.1	94.67	95.07	94.93	93.95	**95.11**	**95.11**	94.92	94.61
0.2	93.20	93.20	93.32	92.82	**93.63**	93.35	93.21	92.51
0.3	90.36	90.45	89.96	**90.47**	89.88	90.40	89.97	90.33
0.4	86.16	85.21	85.06	**87.28**	86.18	85.35	84.98	85.23
0.5	80.75	78.27	78.70	**81.13**	79.10	78.95	77.43	78.73

digit with the minimum reconstruction error.

For this dataset, two types of noise are added to the test set. The first one is Gaussian noise with zero mean and variance σ^2. The second type is "salt and pepper" noise with noise level n, where $n/2$ is the probability that a pixel flips to black (0) or white (255). The classification rates for the test set using various versions of PCA are reported in Table VIII and IX.

In Table VIII, we can see that NG-PCA-Lp with $p = 1.5$ shows the maximum classification rate regardless of the noise level. However, we can also see that the classification rates do not depend significantly on the specific types of PCA when additive Gaussian noise is present. On the other hand, when salt and pepper noise is present, both G-PCA-Lp and NG-PCA-Lp perform better than L2-PCA and R1-PCA, especially when the noise level increases. Note that the best performance is moving from $p = 1.5$ towards $p = 0.5$ as the noise level increases. This shows that PCA-Lp with a small p value better fits samples with large salt and pepper noise because they can cope with large number of outliers. From the tables, we can see that conventional L2-PCA can cope with additive Gaussian noise while PCA-Lp is preferred when non-Gaussian noise is present.

C. Face reconstruction

In this subsection, the proposed PCA-Lp algorithm was applied to Yale face reconstruction problems and the performances were compared with those of other methods. For all the experiments, both G-PCA-Lp and NG-PCA-Lp performed almost the same, and henceforth, the performances of G-PCA-Lp are reported. For all the cases, the maximum number of iterations was set to 100.

The Yale face database consists of 165 gray-scale images of 15 individuals. It includes 11 images per subject with different facial expressions or configurations. In [18], the authors report two types of databases: a closely cropped set and a full face set. In this paper, the full face set of size 100 $\times$ 80 pixels was used. Each of the 8,000 pixels was regarded as an input variable.

In the first experiment, 20% of the total 165 face images were randomly selected and occluded with rectangular noise consisting of random black and white dots of size at least 15 $\times$ 10 located at a random position. The leftmost column of Fig. 6 shows typical examples of occluded images.

To this image set, we applied L2-PCA (eigenface [19]), R1-PCA, and PCA-Lp with $p = 0.5, 1.0$, and 1.5 and extracted various numbers of features. By using only a fraction of the features, we could reconstruct images such as the ones shown on the second to the sixth columns of Fig. 6, and we computed the average reconstruction error $e(m)$ with respect to the original unoccluded images as follows:

$$e(m) = \frac{1}{N}\sum_{i=1}^{N} ||x_i^{org} - \sum_{j=1}^{m} w_j w_j^T x_i||_2. \tag{18}$$

Here, N is the number of samples (165 in this case), x_i^{org} and x_i are the i-th original unoccluded image and the i-th image used in training respectively, and m is the number of extracted features.

Figure 7 shows the average reconstruction errors for various numbers of extracted features. In the figure, when the number of extracted features is small, the average reconstruction errors for different methods are almost the same. However, from around 10 features, the difference among different methods becomes apparent and L2-PCA performs worse than other methods. After around 20 features, PCA-Lp with $p = 0.5$ outperforms the other methods, followed by

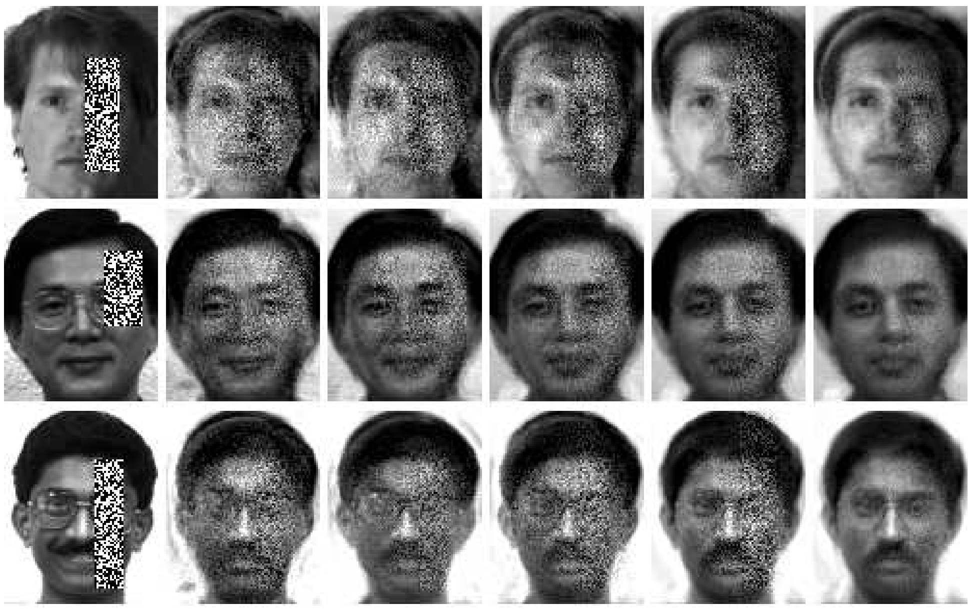


Fig. 6. Face images with occlusion and the reconstructed faces: 1st column: original, 2nd – 4th columns: PCA-Lp ($p = 0.5, 1.0$ and 1.5 respectively), 5th column: L2-PCA, 6th column: R1-PCA (reconstructed with 20 projection vectors).

PCA-Lp ($p = 1, 0$), PCA-Lp ($p = 1.5$), and L2-PCA. Note that L2-PCA corresponds to PCA-Lp with $p = 2.0$. R1-PCA performs best when the number of features is very small, but its performance is overtaken by PCA-Lp with $p = 0.5$ and $p = 1.0$ afterwards. The fluctuation of the reconstruction errors for R1-PCA is because R1-PCA obtains different projection vectors for different values of extracted features m, while the projection vectors of G-PCA-Lp and L2-PCA do not change with m.

Figure 6 shows the original occluded face images and the reconstructed ones using 20 projection vectors respectively. In the figure, we can see that the images reconstructed by L2-PCA have intensive dots while the intensity decreases as p decreases towards 0.5. Comparing the reconstruction images, the ones from R1-PCA are the cleanest with least dots, but the detailed shape and expressions are quite different from the original face images. On the other hand, the ones from PCA-Lp have many dots, but the they are similar to the original ones.

As a second experiment, to the original 165 Yale images, we added 30 dummy images that consist of random black and white dots and performed L2-PCA, R1-PCA, and PCA-Lp with $p = 0.5, 1.0$, and 1.5. Figure 8 shows the average reconstruction error of each method with various numbers of extracted features. In the computation of the average reconstruction error, (18) was used with $N = 165$, i.e., 30 dummy images were excluded. In this case, x_i^{org} and x_i were the same.

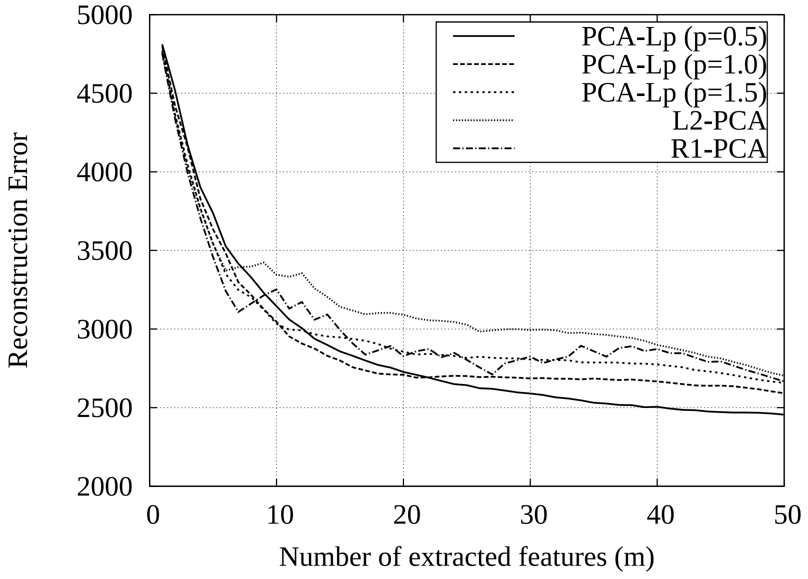


Fig. 7. Average reconstruction errors for occluded Yale images

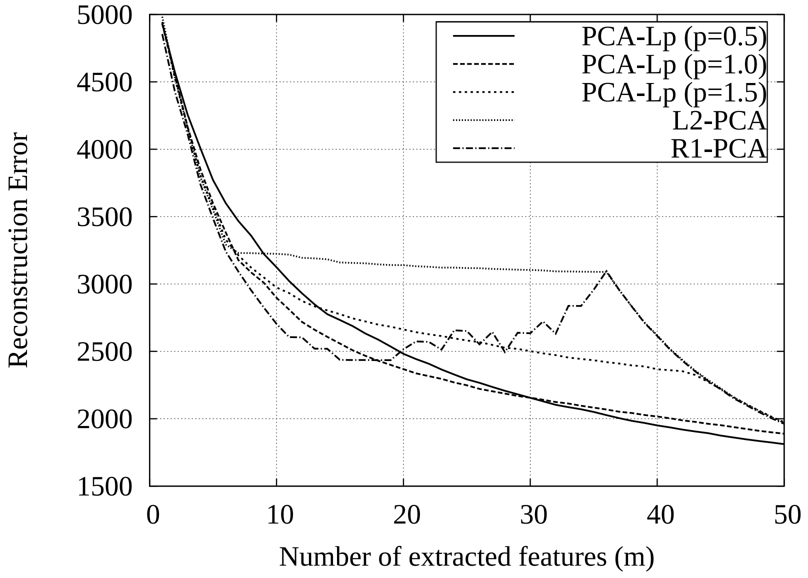


Fig. 8. Average reconstruction errors for Yale dataset with dummy training images

In Fig. 8, when the number of extracted features is changed from 6 to 36, the error of L2-PCA is almost constant. This shows that the dummy images affect the 6th up to the 36th projection vectors significantly, and these vectors are tuned to explain the dummy images. For R1-PCA, this phenomenon starts later, at around $m = 15$, and the performance starts to degrade slowly up to 36 features. On the other hand, PCA-Lp does not suffer

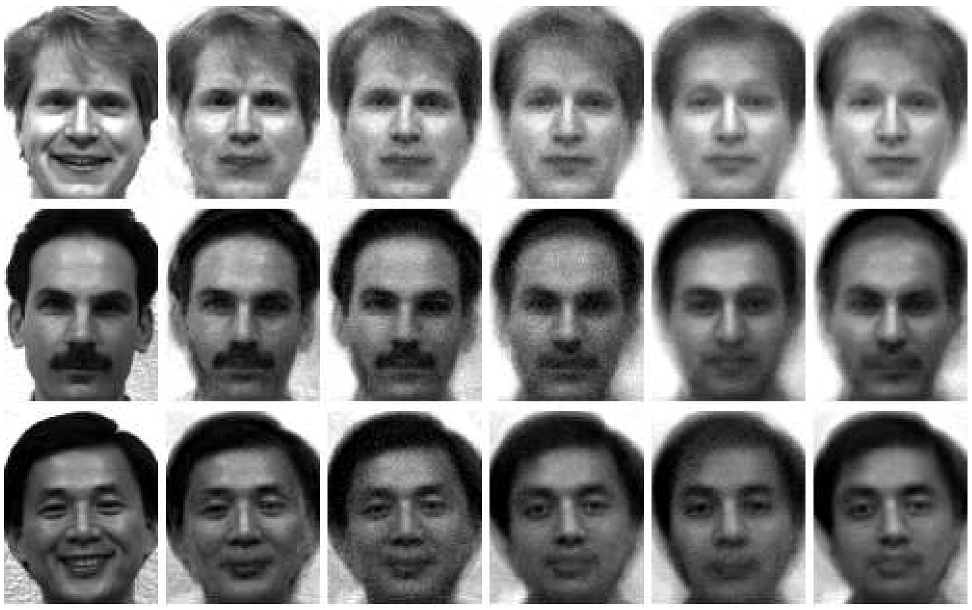


Fig. 9. Face images trained with dummy images and the reconstructed faces: 1st column: original, 2nd – 4th columns: PCA-Lp ($p = 0.5, 1.0$ and 1.5 respectively), 5th column: L2-PCA, 6th column: R1-PCA. (reconstructed with 30 projection vectors)

much from this phenomenon and the reconstruction errors constantly decreases. This can be explained as follows. For PCA-Lp, especially when p is small, each projection vector shares the information on the dummy images, while for R1-PCA and L2-PCA, some of the projection vectors are dedicated to explain dummy images that have a relatively large norm. For this reason, we can see that with small values of p, the performance of PCA-Lp is worse when m is small, but it improves as the number of extracted features increases. As in the previous experiment, the fluctuation of R1-PCA is due to the fact that the whole projection vectors are replaced as the number of extracted features is varied.

Figure 9 shows the reconstructed images with 30 projection vectors as well as the original face images. In Fig. 8, when $m = 30$, the performances of PCA-Lp with $p = 0.5$ and $p = 1.0$ are better than the others, followed by PCA-Lp ($p = 1.5$), R1-PCA, and L2-PCA. Although it is somewhat hard to discern, the images reconstructed using PCA-Lp with small values of p are slightly more similar to the original image in the leftmost column than the ones with larger values of p. Further, it is clearly seen that L2-PCA shows worst performance. The quality of the reconstructed images of R1-PCA (rightmost) is almost as good as that of PCA-Lp with $p = 1.5$ (4th column).

The average number of iterations of PCA-L1 was 7.61 and it took 3,078 *ms* including 2,172

TABLE X

TIME COMPLEXITY OF PCA FOR YALE FACE RECONSTRUCTION PROBLEMS

Time complexity	G-PCA-Lp			NG-PCA-Lp			L2-PCA	R1-PCA
	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 0.5$	$p = 1.0$	$p = 1.5$	$p = 2.0$	
Yale with occlusion: $N = 165$, $m = 50$								
training time (sec)	12.650	3.023	11.533	38.129	9.694	33.393	0.256	186.939
avg. no. iter.	100	8.28	90.84	100	14.78	99.44	1	93.08
Yale with dummy images: $N = 195$, $m = 50$								
training time (sec)	14.760	3.520	13.188	48.752	13.909	49.233	0.294	300.704
avg. no. iter.	100	8.34	88.80	100	16.26	98.38	1	98.88

ms, which was the time taken for preprocessing by L2-PCA. For this problem, R1-PCA took 26,555 *ms* on average.

To check the complexity of the proposed method, we show the training time and the average number of iterations of each PCA in Table X. Note that all the experiments were performed using *Matlab* on an Intel Core2 Duo CPU in 2.93GHz.

In the table, the training times of G-PCA-Lp are the total time taken to obtain 50 weight vectors sequentially. Therefore, the average time for extracting one feature is obtained by dividing the number by 50. For example, when $p = 0.5$, 0.253 (12.650/50) sec are taken on average to obtain one additional feature. On the other hand, because the weight vectors for NG-PCA-Lp and R1-PCA change with different numbers of extracted features m, the training time for NG-PCA-Lp and R1-PCA are the total time taken from $m = 1$ to $m = 50$. Unlike G-PCA-Lp, in these cases, the training time increases with m. For example, R1-PCA takes 3.138 sec when $m = 1$ and it increases to 7.401 when $m = 50$. Likewise, NG-PCA-Lp ($p = 1.5$) took 0.041 and 1.588 sec when $m = 1$ and $m = 50$ respectively. Note that PCA-Lp takes less time than R1-PCA in both experiments. Among different versions of PCA-Lp, it converges relatively fast when $p = 1$ for both G-PCA-Lp and NG-PCA-Lp. Considering that PCA-Lp(L) is used instead of PCA-Lp(G) in this experiment, this is in line with the result of Table I.

VI. CONCLUSION

This paper proposes a number of PCA methods based on L_p-norm optimization techniques. The proposed PCA-Lp methods try to find projections that maximize the general Lp-norm with arbitrary $p > 0$ in the projected space. In doing so, the gradient of the objective function is computed on the basis of the fact that the number of training samples is finite.

As an initial step, we tackled an easier problem of extracting one feature. For this problem, two types of PCA-Lp, namely, PCA-Lp(G) and PCA-Lp(L), were proposed. Both methods employ the gradient of the objective function. In the first one, the gradient ascent method was used, while the second one made use of the Lagrangian multiplier method to maximize the objective function. We also showed the local optimality of PCA-Lp(L) for $p \geq 1$. In addition, it was shown that conventional L2-PCA and PCA-L1 are special cases of PCA-Lp with $p = 2$ and $p = 1$, respectively.

As a second step, the problem of extracting more than one feature was also tackled in this paper. In addition to a simple greedy method, G-PCA-Lp, where the features are extracted one by one greedily using either PCA-Lp(G) or PCA-Lp(L), a non-greedy version, NG-PCA-Lp, where more than one feature is extracted simultaneously, is also proposed. The proposed NG-PCA-Lp can be regarded as an extension of NG-PCA-L1 [10] and the local optimality of NG-PCA-Lp is proved.

The proposed PCA methods were applied to several pattern recognition problems, including face reconstruction problems, and the performances were compared with those of conventional L2-PCA and R1-PCA. The experimental results show that the proposed methods are usually faster than R1-PCA and robust to outliers.

ACKNOWLEDGMENTS

This work was supported by the Korea Research Foundation Grant (KRF-2010-0004908) funded by Korean Government and the Sports Promotion Fund of Seoul Olympic Sports Promotion Foundation from Korean Ministry of Culture, Sports and Tourism (APP0120120412029).

APPENDIX

A. Proof of Theorem 1

From the form of the objective function $F_p(w)$ in (4), we can see that it is convex with respect to w when $p \geq 1$ because the sum of convex functions is convex [20].

From the first order convexity condition [21], if the function $F_p(w)$ is convex, it follows that

$$F_p(w') - F_p(w) \geq \nabla_w^T (w' - w) \geq 0. \tag{19}$$

The second inequality holds because w' is parallel to ∇_w and both w' and w have unit norm.

B. *Proof of Theorem 2*

1) Without loss of generality, it can be assumed that $||w||_2 = 1$. Let w be an eigenvector of S. In this case, because w is an eigenvector of S, $Sw = \lambda w$ for a certain eigenvalue λ. Then, (8) becomes

$$\nabla_w^{\perp} = \lambda(I_d - ww^T)w = \lambda(w - w) = 0 \tag{20}$$

The second equality is from $||w||_2 = 1$.

2) If $p = 2$, it can be easily shown that

$$\nabla_w^{\perp} = (I_d - ww^T)Sw. \tag{21}$$

Assume that we have $\nabla_w^{\perp} = 0$, which is equivalent to $ww^TSw = Sw$. Let us denote ww^T as a matrix A and Sw as a vector v. In this case, it becomes $Av = v$, and v can be interpreted as the eigenvector corresponding to the eigenvalue of 1. Because $A = ww^T$ has rank 1, the only eigenvector that satisfies $Av = v$ is $v = \lambda w$. Therefore, it becomes $Sw = \lambda w$ and w is one of the eigenvectors of S.

C. *Proof of Theorem 3*

The proof is similar to that of *Theorem 1*. First, $F_p(W)$ is convex with respect to W if $p \geq 1$. By the first order convexity condition, if $F_p(W)$ is convex, then

$$\begin{aligned} F_p(W') - F_p(W) &\geq (W' - W) : \nabla_W \\ &= tr((W' - W)^T \nabla_W) \\ &= tr(W'^T \nabla_W) - tr(W^T \nabla_W) \geq 0. \end{aligned} \tag{22}$$

The final inequality holds because W' is the solution of (16).

D. *Proof of Theorem 4*

Using the SVD of W, $G(Q) = tr(Q^T \nabla_W)$ can be rewritten as

$$\begin{aligned} tr(Q^T \nabla_W) &= tr(Q^T U \Lambda V^T) = tr(\Lambda V^T Q^T U) \\ &= tr(\Lambda Z) = \sum_{i=1}^{m} \lambda_{ii} z_{ii}, \end{aligned} \tag{23}$$

where $Z = V^T Q^T U$, λ_{ii}, and z_{ii} are the i-th diagonal elements of Λ and Z, respectively. Since $ZZ^T = I_d$, $z_{ii} \leq 1$. On the other hand, $\lambda_{ii} \geq 0$ because λ_{ii} is a singular value of ∇_W. Therefore, $tr(Q^T \nabla_W) = \sum_{i=1}^{m} \lambda_{ii} z_{ii} \leq \sum_{i=1}^{m} \lambda_{ii}$, and the equality holds when $z_{ii} = 1$ for all

$i(\leq m)$. From this, $G(Q)$ is maximum when $Z = [I_m|\mathbf{0}] \in \Re^{m\times d}$. Because $Z = V^T Q^T U$, we have $Q = UZ^T V^T$ and the optimal solution of (16) is

$$W' = UZ^T V^T = U[I_m|\mathbf{0}]^T V^T. \tag{24}$$

With the above solution, one can verify that the constraint $W'^T W' = I_m$ is automatically satisfied.

References

[1] I.T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, 1986.

[2] F. De la Torre and M.J. Black, “A framework for robust subspace learning,” *International Journal of Computer Vision*, vol. 54, no. 1-3, pp. 117–142, Aug. 2003.

[3] H. Aanas, R. Fisker, K. Astrom, and J. Carstensen, “Robust factorization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1215–1225, Sep. 2002.

[4] C. Ding, D. Zhou, X. He, and H. Zha, “R1-pca: rotational invariant l1-norm principal component analysis for fobust subspace factorization,” in *Proc. International Conference on Machine Learning*, Pittsburgh, PA, June 2006.

[5] A. Baccini, P. Besse, and A.D. Falguerolles, “A l1-norm pca and a heuristic approach,” in *Ordinal and Symbolic Data Analysis*, E. Diday, Y. Lechevalier, and P. Opitz, Eds. 1996, pp. 359–368, Springer.

[6] Q. Ke and T. Kanade, “Robust subspace computation using l1 norm,” Tech. Rep. CMU-CS-03-172, Carnegie Mellon University, Aug. 2003, http://citeseer.ist.psu.edu/ ke03robust.html.

[7] Q. Ke and T. Kanade, “Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 2005.

[8] J. Brooks, J. Dula, and E. Boone, “A pure l1 norm principal component analysis,” Tech. Rep., Verginia Commonwealth University, 2010, Optimization Online.

[9] N. Kwak, “Principal component analysis based on l1-norm maximization,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, Sep. 2008.

[10] F. Nie, H. Huang, C. Ding, D. Luo, and H. Wang, “Robust principal component analysis with non-greedy l1-norm maximization,” in *Proc. 22nd International Conf. on Artificial Intelligence*, 2011, pp. 1433 –1438.

[11] G. Golub and C.V. Loan, *Matrix Computation*, Johns Hopkins University Press, 3 edition, 1996.

[12] X. Li, Y. Pang, and Y. Yuan, “L1-norm based 2dpca,” *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 38, no. 4, pp. 1170–1175, Aug. 2010.

[13] Y. Liu, Y. Lin, and K.C.C. Chan, “Multilinear maximum distance embedding via l1-norm optimization,” in *Proc. 24th AAAI Conf. on Artificial Intelligence*, July 2010.

[14] Y. Pang, X. Li, and Y. Yuan, “Robust tensor analysis with l1-norm,” *IEEE Trans. Circuits Syst. Video Techn.*, vol. 20, no. 2, pp. 172–178, 2010.

[15] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, “Uci repository of machine learning databases,” 1998, For more information contact ml-repository@ics.uci.edu or http://www.ics.uci.edu/ mlearn/MLRepository.html.

[16] B.L. Welch, “The generalization of student’s problem when several different population variances are involved,” *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[17] S. Roweis, “Data for matlab hackers,” Downloadable from http://cs.nyu.edu/ roweis/data.html.

[18] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, “Eigenfaces vs. fisherfaces: recognition using class specific linear projection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, July 1997.

[19] M. Turk and A. Pentland, “Face recognition using eigenfaces,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[20] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge University Press, 2004.

[21] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*, chapter 3.7, Meboo Publishing USA, 2011.

Nojun Kwak was born in Seoul, Korea in 1974. He received the BS, MS, and PhD degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea, in 1997, 1999 and 2003 respectively. From 2003 to 2006, he was with Samsung Electronics. In 2006, he joined Seoul National University as a BK21 Assistant Professor. From 2007 to 2013, he was a Faculty Member of the Department of Electrical and Computer Engineering, Ajou University, Suwon, Korea. Since 2013, he has been with the Graduate School of Convergence Science and Technology, Seoul National University, Suwon, Korea, where he is currently an Associate Professor. His current research interests include pattern recognition, machine learning, computer vision, data mining, image processing, and their applications.