Machine Learning (2019) 108:575–594
https://doi.org/10.1007/s10994-018-5729-x

Improved linear embeddings via Lagrange duality

Kshiteej Sheth[1] · **Dinesh Garg**[1] · **Anirban Dasgupta**[1]

Received: 10 December 2017 / Accepted: 12 June 2018 / Published online: 28 June 2018

Abstract
Near isometric orthogonal embeddings to lower dimensions are a fundamental tool in data science and machine learning. In this paper, we present the construction of such embeddings that minimizes the maximum distortion for a given set of points. We formulate the problem as a non convex constrained optimization problem. We first construct a primal relaxation and then use the theory of Lagrange duality to create a dual relaxation. We also suggest a polynomial time algorithm based on the theory of convex optimization to solve the dual relaxation provably. We provide a theoretical upper bound on the approximation guarantees for our algorithm, which depends only on the spectral properties of the dataset. We experimentally demonstrate the superiority of our algorithm compared to baselines in terms of the scalability and the ability to achieve lower distortion.

Keywords Near isometric embeddings · Dimensionality reduction · Convex and non-convex optimization · Convex relaxation

1 Introduction

One of the fundamental tasks in data science, machine learning, and signal processing applications involving high dimensional data is to embed the data into lower dimensional spaces while preserving pairwise distances (aka similarities) between the data points. Applications include clustering (Badoiu et al. 2002), neighborhood preserving projections and hashing (Indyk and Motwani 1998) among others. Such embeddings are called *Near isometric embeddings*. Formally, given a set of data points $\{\boldsymbol{u_i}\}_{i=1}^{r}$ where $\boldsymbol{u_i} \in \mathbb{R}^d \ \forall \ i \in [r]$, the goal is to find a function $f : \mathbb{R}^d \mapsto \mathbb{R}^k$, where $k << d$, which satisfies,

Editors: Jesse Davis, Elisa Fromont, Derek Greene, and Bjorn Bringmaan..

✉ Kshiteej Sheth
kshiteej.sheth@iitgn.ac.in

Dinesh Garg
dgarg@iitgn.ac.in

Anirban Dasgupta
anirbandg@iitgn.ac.in

[1] Indian Institute of Technology, Palaj, Gandhinagar, Gujarat 382355, India

$$1-\epsilon \leq \frac{\|f(\boldsymbol{u_i})-f(\boldsymbol{u_j})\|_2^2}{\|\boldsymbol{u_i}-\boldsymbol{u_j}\|_2^2} \leq 1+\epsilon \quad \forall i, j \in [r] \tag{1}$$

for a small enough ϵ. Intuitively, it is clear that there is a trade-off between the projected dimension (k) and the maximum distortion (ϵ).

A widely celebrated result of Johnson and Lindenstrauss (1984) says that for a given set of points, an appropriately scaled random linear transformation from $\mathbb{R}^d$ to $\mathbb{R}^k$ can achieve a distortion of ϵ for $k = \mathcal{O}(\log(r)/\epsilon^2)$, with high probability. Such *data oblivious* linear embeddings are popularly known as *JL embeddings*. Further, it was shown by Alon (2003) and Jayram and Woodruff (2013) that such bounds are tight. That means, there exists a set of r points that necessarily require $\Omega(\log(r)/\epsilon^2)$ dimensions in order to be embedded with distortion at most ϵ. This finding leaves an open question of whether one can project the data into an even lower dimensional space by exploiting the geometry of the dataset, while having distortion no more than ϵ. Principal Component Analysis (PCA), while being the de-facto data dependent method for constructing low-dimensional representations, minimizes only the average distortion over the entire set of points. Individual data points can still have an arbitrary distortion under PCA.

Motivated by these observations, we address the question of how to construct a data-dependent orthogonal linear embedding with minimal distortion. An orthonormal linear embedding corresponds to an orthogonal projection of $\mathbb{R}^d$ onto a k-dimensional subspace. When the distortion is under some specified threshold, we call such an embedding a *near isometric orthogonal linear embedding*. One immediate consequence of the orthogonal projection is that the upper bound of inequality (1) becomes trivially 1 (an orthogonal projection can never increase the length of a vector).

Grant et al. (2013) formulated the same problem as a non-convex optimization problem and suggested a semidefinite programming (SDP) based relaxation. Further, they proposed two rounding schemes for the SDP solution to get an approximate solution of the original problem. Their method does have a provable guarantee (either a $\mathcal{O}(k+1)$ approximation, or a bi-criteria one) but is computationally quite expensive as observed in our experimentations. This is potentially due to the large number of constraints in the corresponding SDP relaxation. Luo et al. (2016) and Bah et al. (2014) also introduced heuristic algorithms for learning near isometric linear embeddings which are not necessarily orthonormal. The main issue with these algorithms is that the optimization algorithms proposed by them do not come with any convergence or approximation guarantees. Luo et al. (2016) report that their optimization algorithms might get stuck into local minima because their objective functions are not convex. Still, we do compare our algorithm with these two algorithms. In summary, the key contributions of this paper are as follows.

1. We take a different approach to approximately solve the non-convex optimization problem (referred to as the *primal* problem). In Sect. 2, we first develop a relaxation of this primal problem (referred to as the *relaxed primal* problem) and then construct its Lagrangian dual (referred to as the *relaxed dual* problem). Our relaxed primal problem remains a non-convex problem but our relaxed dual becomes a convex problem. We solve the relaxed dual problem by a projected gradient algorithm and find a dual optimal solution. We then find the primal feasible solution corresponding to this dual optimal solution. We call this overall procedure as LELD (Linear Embeddings via Lagrange Duality).
2. In Sect. 2.2, we show that the solution of the relaxed primal problem corresponding to the relaxed dual optimal solution is indeed a feasible and an approximate solution of the original primal problem.

3. In Sect. 2.3, we prove a theoretical upper bound on the ratio of distortion achieved by LELD to the optimal distortion. This bound depends on the spectral properties of the given dataset. Using this theoretical bound, we argue that for a *well-behaved* dataset, our distortion always remains within 2 times the optimal distortion. Also, our distortion starts approaching close to the optimal distortion as the rank of the dataset starts increasing from 2. Formally, we prove the following key theorem (referred to as Theorem 1) in Sect. 2.3.
   Theorem 1

$$1 \leq \frac{\epsilon_{ALG}}{p^*} \leq \frac{1}{1-\frac{\sigma_1^2}{n}} = \left(1 - \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2+\ldots+\sigma_l^2}\right)^{-1} \leq \frac{1}{1-\frac{\kappa^2}{l}}$$

   where, ϵ_{ALG} *is the distortion achieved by LELD,* p^* *is the optimal distortion,* $n = \binom{r}{2}$ *where* r *is the number of data-points,* l *is the rank of the data matrix,* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_l > 0$ *are the non-zero singular values of the data matrix, and* $\kappa = \sigma_1/\sigma_l$.
4. Our relaxed dual problem has fewer constraints (fewer than the primal), which makes solving the dual relaxation computationally efficient.
5. We experimentally verify that LELD takes less time and achieves lower distortion compared to Grant et al. (2013), Luo et al. (2016) and Bah et al. (2014) in most cases.

2 Duality for linear embeddings

Let $\{\boldsymbol{u_i}\}_{i=1}^r$ be a given dataset where $\boldsymbol{u_i} \in \mathbb{R}^d \; \forall \, i \in [r]$. As mentioned in the previous section, our goal is to orthogonally project (aka embed) these data into a subspace of dimension k. Because the projection is linear as well as orthogonal, the required condition for near isometric embedding becomes

$$1 - \epsilon \leq \frac{\|\mathbf{P}(\boldsymbol{u_i} - \boldsymbol{u_j})\|_2^2}{\|\boldsymbol{u_i} - \boldsymbol{u_j}\|_2^2} \quad \forall i < j \in [r]$$

where, $\mathbf{P}$ is the required linear map. In light of this inequality, we do the following—compute the normalized pairwise differences given by $(\boldsymbol{u_i} - \boldsymbol{u_j}) / \|(\boldsymbol{u_i} - \boldsymbol{u_j})\|$ for every pair (i, j), where $i < j$; $(i, j) \in [r]$, and then orthogonally project them to preserve their squared lengths. This way, we get $\binom{r}{2}$ vectors each being unit length. If we let $n = \binom{r}{2}$ then, we can denote such vectors by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ and our goal becomes to orthogonally project them so as to preserve their squared lengths. For this, we let $\boldsymbol{v}_1 \neq 0, \boldsymbol{v}_2 \neq 0, \ldots, \boldsymbol{v}_k \neq 0$ be some orthogonal basis vectors of a k-dimensional subspace $\mathcal{S}_k$ of $\mathbb{R}^d$. Further, let $\mathbf{V}$ be a $d \times k$ matrix whose columns are given by the orthogonal basis vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k$. Let $\tilde{\boldsymbol{x}}_i \in \mathcal{S}_k$ be the vector obtained by projecting $\boldsymbol{x}_i$ orthogonally into the space $\mathcal{S}_k$. Then, reducing the ϵ in (1) is equivalent to minimizing a real number ϵ such that the distortion of each of the x_i's after projection is less than ϵ. Here, the distortion is given by the squared length of the residual vector $(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)$, which would be equal to $1 - \|\tilde{\boldsymbol{x}}_i\|^2$. Formally, it can be stated as the following optimization problem which we refer to as the **Primal** problem.

$$\begin{aligned} &\underset{\boldsymbol{v}_1 \neq 0, \ldots, \boldsymbol{v}_k \neq 0, \epsilon}{\text{Minimize}} \ \epsilon \\ &\text{subject to } \ 1 - \|\mathbf{V}^\top \boldsymbol{x}_i\|_2^2 \leq \epsilon, \ \forall i = 1, \ldots, n \\ &\boldsymbol{v}_j^\top \boldsymbol{v}_j = 1, \ \forall j = 1, \ldots, k \\ &\boldsymbol{v}_j^\top \boldsymbol{v}_m = 0, \ \forall j \neq m \end{aligned} \tag{Primal}$$

The last two constraints force the matrix $\mathbf{V}$ to be orthonormal. The first constraint, on the other hand, identifies the vector for which $1 - \|\tilde{\boldsymbol{x}}_i\|^2$ is maximum. The objective function tries to minimize $1 - \|\tilde{\boldsymbol{x}}_i\|^2$ for such a vector. Observe that whenever $\mathbf{V}$ is an orthogonal matrix, we have $\|\tilde{\boldsymbol{x}}_i\|^2 = \|\mathbf{V}^\top \boldsymbol{x}_i\|^2 \leq 1$ for each $i = 1, \ldots, n$. Let p^* be the optimal value of the above problem. It is easy to verify that $0 \leq p^* \leq 1$.

2.1 Lagrangian dual

The (Primal) problem is a non-convex optimization problem as the feasible region forms a non-convex set. Thus, we cannot hope to solve this problem efficiently and hence we aim for approximate solutions.

For this, we momentarily ignore the last equality constraint in the (Primal) problem, which enforces the orthogonality of the v_j's, and consider the following *Relaxed Primal* problem. We will later prove that in spite of this relaxation, we will reach a solution in which the vectors are indeed orthogonal, thereby satisfying the constraints of the (Primal) problem and justifying the effectiveness of our approach.

$$\begin{aligned} &\underset{\boldsymbol{v}_1 \neq 0, \ldots, \boldsymbol{v}_k \neq 0, \epsilon}{\text{Minimize}} \ \epsilon \\ &\text{subject to } \ 1 - \|\mathbf{V}^\top \boldsymbol{x}_i\|_2^2 \leq \epsilon, \ \forall i = 1, \ldots, n \\ &\boldsymbol{v}_j^\top \boldsymbol{v}_j = 1, \ \forall j = 1, \ldots, k \end{aligned} \tag{Relaxed Primal}$$

We now state a lemma that will be used later, we omit the proof as it is fairly straightforward.

Lemma 1 *Let $\widehat{p^*}$ be the optimal value of the (Relaxed Primal). Then, the following holds.*

(A) $\widehat{p^*} \leq p^* \leq 1$.
(B) If an optimal solution of the (Relaxed Primal) problem satisfies the constraints of the (Primal) problem then that solution must be an optimal solution for the (Primal) problem.

Note, the constraint $\boldsymbol{v}_j^\top \boldsymbol{v}_j = 1$ forces both Primal as well as (Relaxed Primal) problems to be non-convex. The reason being following—*any norm is a convex function and the level set of a norm is a non-convex set*. This motivates us to work with the Lagrangian dual of the (Relaxed Primal) and develop a strategy to get an approximate solution of the (Primal).

2.2 Dual of relaxed primal

The Lagrangian dual for the (Relaxed Primal) will always be a convex program irrespective of the non-convexity of the (Relaxed Primal). For this, we write down the Lagrangian function

of the (Relaxed Primal) as follows:

$$L(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \ \epsilon, \ \lambda_1, \ldots, \lambda_n, \mu_1, \ldots, \mu_k)$$
$$= \epsilon + \sum_{i=1}^{n} \lambda_i \left(1 - \|\mathbf{V}^\top \boldsymbol{x}_i\|_2^2 - \epsilon\right) + \sum_{j=1}^{k} \mu_j \left(\boldsymbol{v}_j^\top \boldsymbol{v}_j - 1\right) \tag{2}$$

Observe that $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \ \epsilon$ are the primal variables and $\lambda_1, \ldots, \lambda_n, \mu_1, \ldots, \mu_k$ are the dual variables.

The dual is given by, $g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \underset{\boldsymbol{v}_1 \neq, \ldots, \boldsymbol{v}_k \neq 0, \epsilon}{\text{argmin}} \ L(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \epsilon, \lambda_1, \ldots \lambda_n, \mu_1, \ldots, \mu_k)$.

For any set of $\{\lambda_i, \lambda_i \geq 0\}$, we define $M = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top = \mathbf{X}^\top diag(\lambda_1, \ldots, \lambda_n)\mathbf{X}$. The following lemma characterizes the dual solution in terms of this matrix M.

Lemma 2 *Define* $M = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top = \mathbf{X}^\top diag(\lambda_1, \ldots, \lambda_n)\mathbf{X}$. *Then for any given values of dual variables* $(\lambda_1, \ldots, \lambda_n, \mu_1, \ldots, \mu_k)$, $\lambda_i \geq 0$, *we have*

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{cases} 1 - \sum_{j=1}^{k} \mu_j : \textit{If } \sum_{i=1}^{n} \lambda_i = 1 \textit{ and } \{\mu_j\}_{j=1}^{k} \\ \textit{are the top-}k \textit{ eigenvalues of } M \\ -\infty \textit{ or undefined} : \textit{Otherwise} \end{cases}$$

And, the top-k eigenvectors of the matrix M are the minimizers of Lagrangian function.

Proof In order to get the dual function, we set the gradient of Lagrangian function (with respect to primal variables) to be zero. This gives us the following conditions:

$$\partial L / \partial \boldsymbol{v}_j = 2\mu_j \boldsymbol{v}_j - 2\left(\sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top\right) \boldsymbol{v}_j = 0 \tag{3}$$

$$\partial L / \partial \epsilon = 1 - \sum_{i=1}^{n} \lambda_i = 0 \tag{4}$$

Using the definition of matrix M, we first rewrite the expression for Lagrangian as follows.

$$L = \epsilon \left(1 - \sum_{i=1}^{n} \lambda_i\right) + \sum_{i=1}^{n} \lambda_i - \sum_{j=1}^{k} \mu_j$$
$$- \sum_{j=1}^{k} \boldsymbol{v}_j^\top \left(M - \mu_j \mathbf{I}\right) \boldsymbol{v}_j \tag{5}$$

The minimum value of the Lagrangian has to satisfy the first-order conditions that we get by setting the gradient of Lagrangian function (with respect to the primal variables) to be zero. This gives us the following conditions.

$$\partial L / \partial \boldsymbol{v}_j = 2\mu_j \boldsymbol{v}_j - 2\left(\sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top\right) \boldsymbol{v}_j = 0 \tag{6}$$

$$\partial L / \partial \epsilon = 1 - \sum_{i=1}^{n} \lambda_i = 0 \tag{7}$$

By looking at these conditions, the following claims follow.

1. From Eq. (7), $\sum_{i=1}^{n} \lambda_i = 1$. An alternative argument could be as follows: since ϵ is unconstrained, if $\sum_{i=1}^{n} \lambda_i \neq 1$, then it is possible to set ϵ such that $g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = -\infty$.
2. Equation (6) implies that for achieving minimum, the primal variables $\boldsymbol{v}_j$ must be an eigenvector of the matrix $M = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top$ whose corresponding eigenvalue is the dual variable μ_j.
   We argue that setting μ_j to be any value other than the top-jth eigenvalue of the M leads to $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ being either $-\infty$ or undefined. For this, note that under the condition given

by (7), the Lagrangian becomes

$$1 - \sum_{j=1}^{k} \mu_j - \sum_{j=1}^{k} \boldsymbol{v}_j^\top \left(M - \mu_j \mathbf{I}\right) \boldsymbol{v}_j$$

Without loss of generality, we can assume that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_k$. Suppose $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_d$ are the eigenvalues of the the matrix M. For contradiction assume that μ_1 is not an eigenvalue of the matrix M. Now, let us consider two different cases.

(a) *Case of* $[\boldsymbol{\gamma_1} > \boldsymbol{\mu_1}]$: In this case, we can assign $\boldsymbol{v}_1$ to be the scaled eigenvector corresponding to the eigenvalue γ_1 and that would drive the Lagrangian value towards $-\infty$. Therefore, in order to avoid the $-\infty$ value for the dual function, we must have $\mu_1 \geq \gamma_1$.

(b) *Case of* $[\boldsymbol{\mu_1} > \boldsymbol{\gamma_1}]$: Under this scenario, we again consider two subcases:

i. *Subcase of* $[\boldsymbol{\gamma_1} > \boldsymbol{\mu_k}]$: In such a case, by the previous argument, we can again drive the Lagrangian to $-\infty$ by assigning $\boldsymbol{v}_k$ appropriately.

ii. *Subcase of* $[\boldsymbol{\mu_k} > \boldsymbol{\gamma_1}]$: For this subcase, note that matrix $\left(M - \mu_j \mathbf{I}\right)$ would be a negative definite matrix for each $j \in [k]$ and hence $\boldsymbol{v}_j = 0, \ \forall j \in [k]$ will minimize the Lagrangian. However, all v_j cannot simultaneously be zero as per the constraint. Hence, in this case, the minimum is not defined within the (open) set of $\{v_j, \ \forall j \ v_j \neq 0\}$.

Thus, we can conclude that μ_1 should be equal to γ_1 in order to make sure that the dual function is well defined and has a finite value. This also means that v_1 has to be the top eigenvector of M. We can inductively apply the same argument for the other μ_j also to get the desired claim.

□

Lemma 2 suggests that the vectors $\boldsymbol{v}_1 \neq 0, \ldots, \boldsymbol{v}_k \neq 0$ which minimize the Lagrangian must be the top-k eigenvectors of the matrix M and in such a case, the dual function can be given as follows $g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = 1 - \sum_{i=1}^{k} \mu_j$ where, μ_j is top jth eigenvalue of the matrix M. The dual optimization problem for the (Relaxed Primal), thus becomes

$$\begin{aligned}
&\underset{\boldsymbol{\lambda}, \boldsymbol{\mu}}{\text{Maximize}} \ \ g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = 1 - \sum_{j=1}^{k} \mu_j \\
&\text{subject to} \\
&\mu_j \text{ is top} j\text{th eigenvalue of } M, \ \forall j = 1, \ldots, k \\
&\sum_{i=1}^{n} \lambda_i = 1 \\
&\lambda_i \geq 0, \ \forall i = 1, \ldots, n
\end{aligned} \qquad \text{(Relaxed Dual)}$$

Let $\widehat{d^*}$ be the optimal value of the (Relaxed Dual) problem. In what follows, we state a few key observations with regard to the above primal-dual formulation discussed so far.

Lemma 3 *For the given* (Primal), (Relaxed Primal), *and* (Relaxed Dual) *programs, the following hold true.*

(A) $\widehat{d^*} \leq \widehat{p^*} \leq p^* \leq 1$

(B) If $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ *is a feasible solution of the* (Relaxed Dual), *then* $0 \leq \mu_j, \ \forall j \in [k]$ *and* $0 \leq \sum_{j=1}^{k} \mu_k \leq Tr(M) = \sum_{i=1}^{n} \lambda_i = 1$.

(C) $Rank(M) \leq Rank(\mathbf{X}^\top \mathbf{X}) = Rank(\mathbf{X})$.

(D) Let $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ *be the point of optimality for the* (Relaxed Dual) *problem. Then the top-k eigenvectors* $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ *of* $M = \sum_{i=1}^n \lambda_i^* \boldsymbol{x}_i \boldsymbol{x}_i^\top = \boldsymbol{\lambda}^* \mathbf{X}^\top \mathbf{X}$ *satisfy,*

- *(a) These vectors minimize* $L(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \ \epsilon, \ \lambda_1^*, \ldots, \lambda_n^*, \mu_1^*, \ldots, \mu_k^*)$.
- *(b) These vectors form a feasible solution for both* (Relaxed Primal) *and* (Primal) *problems thus allowing us to use vectors obtained from the* (Relaxed Dual) *as approximate solutions of the* (Primal).
- *(c) Let* ϵ_{ALG} *be the* (Relaxed Primal) *objective function value (which is also the value for the* (Primal) *objective function) corresponding to the feasible solution* $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ *then we must have* $p^* \leq \epsilon_{ALG} \leq 1$.

Proof *Part (A):* The inequalities follow from weak duality theorem and the fact that $\widehat{p^*}$ is the optimal solution of the (Relaxed Primal) problem, whereas p^* is the optimal solution of the (Primal) problem. The last inequality follows from Lemma 1.

Part (B): Let $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ be a feasible solution for (Relaxed Dual) problem then by Lemma 2, we can claim that $\mu_j \geq 0$ because μ_j must be an eigenvalue of the matrix $\mathbf{M}$ and this matrix is always positive semidefinite. Further, notice that

$$\begin{aligned} Tr(\mathbf{M}) &= Tr(\sum_{i=1}^n \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^\top) \\ &= \sum_{i=1}^n \lambda_i \ Tr(\boldsymbol{x}_i \boldsymbol{x}_i^\top) = \sum_{i=1}^n \lambda_i Tr(\boldsymbol{x}_i^\top \boldsymbol{x}_i) = \sum_{i=1}^n \lambda_i = 1 \end{aligned}$$

where, the last part of the equation follows from the fact that vectors $\boldsymbol{x}_i$ are unit vectors. Further, if $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a feasible solution for (Relaxed Dual) then we must also have

$$\sum_{j=1}^k \mu_j \leq \sum_{j=1}^d \mu_j = Tr(\mathbf{M}) = 1 \tag{8}$$

where, $\mu_{(k+1)}, \ldots, \mu_d$ are the bottom $(d-k)$ eigenvalues of the matrix M. This proves part (B) of the lemma.

Part (C): This part follows from standard results in linear algebra.

Part (D): Let $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be the point of optimality for the (Relaxed Dual) problem. Let $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ be the top-k eigenvectors of the matrix $\mathbf{M} = \sum_{i=1}^n \lambda_i^* \boldsymbol{x}_i \boldsymbol{x}_i^\top = \boldsymbol{\lambda}^* \mathbf{X}^\top \mathbf{X}$. Because $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is an optimal solution for the (Relaxed Dual) problem, it must be a feasible solution also for the same problem. This would mean that $\boldsymbol{\mu}_j^*, \ \forall j \in [k]$ must be top-jth eigenvalue of the matrix $\mathbf{M} = \sum_{i=1}^n \lambda_i^* \boldsymbol{x}_i \boldsymbol{x}_i^\top$. Therefore, by Lemma 2, we can say that the vectors $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ must minimize the Lagrangian $L(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k, \ Z, \ \lambda_1^*, \ldots, \lambda_n^*, \mu_1^*, \ldots, \mu_k^*)$. The feasibility of the vectors $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ for Primal and (Relaxed Primal) problems is trivial due to the fact that these vectors are orthonormal basis vectors. The first part of the inequality $p^* \leq \epsilon^* \leq 1$ is trivial because p^* is optimal. The second part of this inequality follows from the second inequality given in part (B) of this lemma. □

2.3 Approximate solution of (Primal) problem

Recall that Primal is a non-convex optimization problem. We analyze the approximation factor of the following version of our algorithm.

Approximation Algorithm:

1. We first find an optimal solution $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ for the (Relaxed Dual) problem.
2. Next, we find the top-k eigenvectors $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ of $M = \sum_{i=1}^n \lambda_i^* \boldsymbol{x}_i \boldsymbol{x}_i^\top$ and treat them as an approximate solution for the original (Primal) problem. Note, these eigenvectors form a feasible solution for the Primal and (Relaxed Primal) problems (Lemma 3).

In this section, we try to develop a theoretical bound on the quality of such an approximate solution for the original (Primal). For this, note that the objective function value for both (Relaxed Primal) and (Primal) problems is identical for the feasible solution $(\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*)$ obtained by the method suggested above. Moreover, this value is given by $\epsilon_{ALG} = \max_{i=1,\ldots,n} \phi_i$, where $\phi_i = 1 - \|\mathbf{V}^{*\top} \boldsymbol{x}_i\|_2^2$ and $\mathbf{V}^*$ is a matrix comprising of the top-k eigenvectors $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_k^*$ of the matrix $\mathbf{M}$ as its columns. The following inequality follows trivially from the above fact and Lemma 3 part (A).

$$\widehat{p^*} \leq p^* \leq \epsilon_{ALG} \leq 1 \tag{9}$$

Further, note that for the above algorithm, the optimal objective function value for the (Relaxed Dual) would be $\widehat{d^*} = 1 - \sum_{j=1}^k \mu_j^*$, where μ_j^* is the jth-top eigenvalue of the matrix $\mathbf{M}$. By combining the Inequality (9) with Lemma 3 part (A), we can say that

$$\frac{\epsilon_{ALG}}{p^*} \leq \frac{\epsilon_{ALG}}{\widehat{d^*}} \leq \frac{1}{1 - \sum_{j=1}^k \mu_j^*} \tag{10}$$

In order to obtain a meaningful upper bound on the above inequality, we recall the definition of matrix $\mathbf{X}$ whose size is $n \times d$, and whose rows are unit length data vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$. Suppose $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0$ are singular values of the matrix $\mathbf{X}$ out of which only ℓ are non-zero, where $\ell \leq d$ is the rank of the data matrix $\mathbf{X}$. The following theorem gives us the bound on the approximation ratio of our proposed algorithm, where κ is the ratio of the highest singular value to the lowest non-zero singular value for the matrix $\mathbf{X}$, i.e., $\kappa = \sigma_1/\sigma_\ell$.

Theorem 1 *The approximation algorithm described above offers the following approximation guarantees.*

$$\frac{\epsilon_{ALG}}{p^*} \leq \frac{1}{1 - \frac{\sigma_1^2}{n}} = \left(1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_l^2}\right)^{-1} \leq \frac{1}{1 - \frac{\kappa^2}{l}} \tag{11}$$

Proof As per Inequality (10), in order to prove the claim of this theorem, it suffices to get an upper bound on the quantity $\sum_{j=1}^k \mu_j^*$. Thus, our goal is to get an upper bound on the sum of the top-k singular values of the matrix $\mathbf{M}$. For this, we note that $\mathbf{M} = \mathbf{X}^\top \boldsymbol{\Lambda} \mathbf{X}$, where $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$, $\sum_{i=1}^n \lambda_i = 1$, and $\lambda_i \geq 0 \ \forall i$. Let the SVD of the matrix $\mathbf{X}^\top$ be $\mathbf{X}^\top = \mathbf{U} \boldsymbol{\Sigma} \widetilde{\mathbf{V}}^\top$, where $\mathbf{U}$ is a $d \times d$ orthogonal matrix, $\boldsymbol{\Sigma}$ is a $d \times d$ diagonal matrix containing $\sigma_1, \sigma_2, \ldots, \sigma_d$ on its diagonal, and $\widetilde{\mathbf{V}}$ is an $n \times d$ matrix having orthonormal column vectors $\widetilde{\boldsymbol{v}}_1, \widetilde{\boldsymbol{v}}_2, \ldots, \widetilde{\boldsymbol{v}}_d$. Recall that, we have used the symbol $\mathbf{V}$ to denote the solution of the primal problem which is a $d \times k$ matrix and hence we are using a different symbol $\widetilde{\mathbf{V}}^\top$ to denote the left singular vectors of the matrix $\mathbf{X}^\top$.

By using the SVD of the matrix $\mathbf{X}^\top$, we can rewrite the expression for matrix $\mathbf{M}$ as below.

$$\mathbf{M} = \mathbf{U} \boldsymbol{\Sigma} \widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda} \widetilde{\mathbf{V}} \boldsymbol{\Sigma} \mathbf{U}^\top \tag{12}$$

Recall that matrix $\mathbf{M}$ is a PSD matrix and hence, we must have

$$\sum_{j=1}^{k} \mu_j \leq \sum_{j=1}^{d} \mu_j = \text{Tr}(\mathbf{M}) \tag{13}$$

By making use of Eq. (12), the above inequality can be written as

$$\sum_{j=1}^{k} \mu_j \leq \text{Tr}(\mathbf{U}\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma}\mathbf{U}^\top) = \text{Tr}(\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma}) \tag{14}$$

From the definition of the Relaxed Dual, we can write

$$\sum_{j=1}^{k} \mu_j^* = \underset{\Lambda}{\text{Min}} \left(\sum_{j=1}^{k} \mu_j \right) \leq \underset{\Lambda}{\text{Min}} \left(\text{Tr}(\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma}) \right) \tag{15}$$

In lieu of the fact that $\boldsymbol{\Lambda}$ is a diagonal matrix having $\lambda_1, \ldots, \lambda_n$ on its diagonal, where $\sum_i \lambda_i = 1$ and $\lambda \geq 0$, and $\widetilde{\boldsymbol{v}}_1, \widetilde{\boldsymbol{v}}_2, \ldots, \widetilde{\boldsymbol{v}}_d$ are orthonormal column vectors of the matrix $\widetilde{\mathbf{V}}$, it is not difficult to see that the ith diagonal entry of the matrix $\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma}$ can be given by

$$\left[\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma} \right]_{ii} = \sigma_i^2 \left(\widetilde{v}_{i1}^2 \lambda_1 + \widetilde{v}_{i2}^2 \lambda_2 + \ldots + \widetilde{v}_{in}^2 \lambda_n \right) \tag{16}$$

where, we follow the convention that

$$\widetilde{\mathbf{V}}^\top = \begin{bmatrix} \widetilde{\boldsymbol{v}}_1^\top \\ \ldots \\ \widetilde{\boldsymbol{v}}_d^\top \end{bmatrix} = \begin{bmatrix} \widetilde{v}_{11} & \widetilde{v}_{12} & \ldots & \widetilde{v}_{1n} \\ & \ldots & & \\ \widetilde{v}_{d1} & \widetilde{v}_{d2} & \ldots & \widetilde{v}_{dn} \end{bmatrix} \tag{17}$$

This would imply that

$$\text{Tr}\left(\boldsymbol{\Sigma}\widetilde{\mathbf{V}}^\top \boldsymbol{\Lambda}\widetilde{\mathbf{V}}\boldsymbol{\Sigma} \right) = \sum_{i=1}^{d} \sigma_i^2 \left(\widetilde{v}_{i1}^2 \lambda_1 + \widetilde{v}_{i2}^2 \lambda_2 + \ldots + \widetilde{v}_{in}^2 \lambda_n \right) \tag{18}$$

$$= \lambda_1 \left(\sum_{i=1}^{d} \sigma_i^2 \widetilde{v}_{i1}^2 \right) + \lambda_2 \left(\sum_{i=1}^{d} \sigma_i^2 \widetilde{v}_{i2}^2 \right) + \ldots + \lambda_n \left(\sum_{i=1}^{d} \sigma_i^2 \widetilde{v}_{in}^2 \right) \tag{19}$$

Substituting Eq. (19) into the Inequality (15) would give us the following inequality.

$$\sum_{i=1}^{k} \mu_j^* \leq \underset{j=1,\ldots,n}{\text{Min}} \sum_{i=1}^{d} \sigma_i^2 \widetilde{v}_{ij}^2 \leq \underset{j=1,\ldots,n}{\text{Min}} \sum_{i=1}^{d} \sigma_i^2 \widetilde{v}_{ij}^2$$
$$= \sigma_1^2 \underset{j=1,\ldots,n}{\text{Min}} \left(\widetilde{v}_{1j}^2 + \widetilde{v}_{2j}^2 + \ldots + \widetilde{v}_{dj}^2 \right)$$

Again, because of the fact that $\widetilde{\boldsymbol{v}}_1, \widetilde{\boldsymbol{v}}_2, \ldots, \widetilde{\boldsymbol{v}}_d$ are orthonormal column vectors of the matrix $\widetilde{\mathbf{V}}$, we can write

$$\sum_{j=1}^{n} \left(\widetilde{v}_{1j}^2 + \widetilde{v}_{2j}^2 + \ldots + \widetilde{v}_{dj}^2 \right) = 1 \tag{20}$$

This implies that

$$\sum_{i=1}^{k} \mu_j^* \leq \sigma_1^2 \underset{j=1,\ldots,n}{\text{Min}} \left(\widetilde{v}_{1j}^2 + \widetilde{v}_{2j}^2 + \ldots + \widetilde{v}_{dj}^2 \right) \leq \sigma_1^2 / n \tag{21}$$

This gives us the first part of the inequality in the theorem. In order to get the second part of the inequality, we note that the following relation is easy to verify.

$$n = \text{Tr}\left(\mathbf{X}\mathbf{X}^\top \right) = \text{Tr}\left(\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top \right) = \sigma_1^2 + \sigma_2^2 + \ldots + \sigma_l^2 \tag{22}$$

Substituting the above equation into the Inequality (21) would yield the second desired inequality in the theorem's statement. The last inequality in the theorem statement follows from the definition of κ for the matrix $\mathbf{X}^\top$ as given by, $\kappa = \sigma_1/\sigma_l$. □

Below are some interesting and useful insights about our algorithm that one can easily derive with the help of Theorem 1.

- The above theorem bounds the gap between the primal objective value corresponding to our approximate solution and the unknown optimal primal objective value.
- The approximation bound given in Theorem 1 depends on the spectral properties of the input dataset. For example, if the given dataset $\mathbf{X}$ is well-behaved, that is $\kappa = 1$, then we obtain the approximation factor of $\frac{l}{l-1}$ which is bounded above by 2 assuming $\ell \geq 2$.
- Further, in such a case, this bound starts approaching towards 1 as the rank of the data matrix increases (of course, the dimension d of the data has to increases first for ℓ to increase). Thus, we can say that for a well-behaved dataset of full rank in very large dimensions, our algorithm offers a nearly optimal solution for the problem of length preserving orthogonal projections of the data.

3 Projected gradient ascent

We now give a provable algorithm to get an optimal solution of the Relaxed Dual problem, which we call Projected Gradient Ascent. Note that our overall algorithm LELD is the combination of Projected Gradient Ascent to get the optimal solution of the Relaxed Dual and the Approximation Algorithm described in Sect. 2.3 to get a feasible solution of the Primal from the optimal solution of the Relaxed Dual. Since the dual formulation is always a convex program (Boyd and Vandenberghe 2004), the objective function of the Relaxed Dual is concave in the dual variables and the feasible set formed by the constraint set is a convex set $\mathbf{C} \subset \mathbb{R}^n$. In each iteration, Projected Gradient Ascent essentially performs a gradient ascent update on the dual variables $\lambda_i, i = 1, \ldots, n$ and then projects the update back onto the convex set $\mathbf{C}$. Note that we need not worry about the dual variables μ_j because for any assignment of the variables λ_i, the values of the μ_j get determined because of the first constraint in the (Relaxed Dual) problem. Therefore, we perform gradient ascent only on the λ_i variables.

Observe that the feasible region of the λ_i variables forms the standard probability simplex in $\mathbb{R}^n$. Because of this, we make use of the *Simplex Projection algorithm* proposed by Wang and Carreira-Perpinan (2013) for the purpose of performing the projection step in each iteration of our algorithm. This algorithm runs in time $\mathcal{O}(d \log(d))$ time. We call the projection step of this algorithm as ($\mathbf{Proj}_{\mathbf{C}}$).

The pseudo code for Projected Gradient Ascent is given in the form of Algorithm 1. In line number 5 (and also 7) of this code, we compute the (Primal) objective function value ϵ for two different dual feasible solutions, namely $\boldsymbol{\lambda}^{(t+1)}$ and $\boldsymbol{\lambda}^{best}$. Note that the last `if-else` statement identifies the better of the two solutions (in terms of the primal objective function value). The routine $\mathbf{Proj}_{\mathbf{C}}(\cdot)$ is given in Algorithm 2.

Now, we present a key lemma related to the gradient of the (Relaxed Dual) objective.

Lemma 4 (Dual gradient) *If $\boldsymbol{\lambda}^{(t)}$ is not an optimal solution of* (Relaxed Dual) *problem then for $\ell = 1, \ldots, n$, the ℓth coordinate of the gradient vector $\nabla g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)$ is given by*

$$\partial g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)/\partial \lambda_\ell^{(t)} = -\sum\nolimits_{j=1}^{k} \|\boldsymbol{x}_\ell^\top \boldsymbol{v}_j^{(t)}\|^2 \tag{23}$$

Algorithm 1: Projected Gradient Ascent

Input : $x_i, i = 1, \ldots, n$

Initialize: $\lambda_i^0 = \lambda_i^{best} = 1/n \; \forall i \in [n]$

for $t = 1, \ldots, T$ **do**
  Compute $\nabla g(\boldsymbol{\lambda}, \boldsymbol{\mu}))$ for $(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)$ by making use of Lemma 4;
  $\widetilde{\boldsymbol{\lambda}}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \eta \nabla g(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)})$;
  $\boldsymbol{\lambda}^{(t+1)} \leftarrow \mathbf{Proj}_{\mathbf{C}}\left(\widetilde{\boldsymbol{\lambda}}^{(t+1)}\right)$;
  if $\left(\epsilon \, for \, \boldsymbol{\lambda}^{(t+1)}\right) < \left(\epsilon \, for \, \boldsymbol{\lambda}^{best}\right)$ **then**
    $\boldsymbol{\lambda}^{best} = \boldsymbol{\lambda}^{(t+1)}$
if $\left(\epsilon \, for \, \boldsymbol{\lambda}^{best}\right) < \left(\epsilon \, for \, \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\lambda}^{(t)}\right)$ **then**
  Output : $\boldsymbol{\lambda}^{best}$
else
  Output : $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\lambda}^{(t)}$

Algorithm 2: $\mathbf{Proj}_{\mathbf{C}}(\cdot)$

Input : $\boldsymbol{\lambda} \in \mathbb{R}^n$

Objective: Find $\underset{\widetilde{\boldsymbol{\lambda}} \in \mathbf{C}}{argmin} \|\widetilde{\boldsymbol{\lambda}} - \boldsymbol{\lambda}\|_2$

Sort coordinated of $\boldsymbol{\lambda}$ into $\lambda_{(1)} \geq \ldots \lambda_{(n)}$;
$\rho \leftarrow \max_{j \in [n]} \left\{\lambda_{(j)} + \left(1 - \sum_{i=1}^{j} \lambda_{(i)}\right)/j\right\}$;
$\alpha \leftarrow (1/\rho)\left(1 - \sum_{i=1}^{\rho} \lambda_{(i)}\right)$;

Output : $\widetilde{\boldsymbol{\lambda}}$ such that $\widetilde{\lambda_i} = \max(\lambda_i + \alpha, 0) \; \forall i \in [n]$

Proof Note, the gradient vector $\nabla g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)$ would be

$$\partial g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)/\partial \boldsymbol{\lambda}^{(t)} = \partial\left(1 - \sum\nolimits_{j=1}^{k} \mu_j^{(t)}\right)/\partial \boldsymbol{\lambda}^{(t)}$$

The ℓth coordinate of this ascent direction can be given by

$$\partial g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)/\partial \lambda_\ell^{(t)} = -\partial\left(\sum\nolimits_{j=1}^{k} \mu_j^{(t)}\right)/\partial \lambda_\ell^{(t)}, \forall \ell \in [n]$$

Let $\boldsymbol{v}_1^{(t)}, \ldots \boldsymbol{v}_k^{(t)}$ are the top-k eigenvectors of the matrix $\mathbf{M}(\boldsymbol{\lambda}^{(t)})$ then the above equation can be written as

$$\partial g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)/\partial \lambda_\ell^{(t)} = -\partial\left(\sum\nolimits_{j=1}^{k} \boldsymbol{v}_j^{(t)\top} \mathbf{M}(\boldsymbol{\lambda}^{(t)}) \boldsymbol{v}_j^{(t)}\right)/\partial \lambda_\ell^{(t)}$$

Recall that $\mathbf{M}(\boldsymbol{\lambda}^{(t)}) = \sum_{i=1}^{n} \lambda_i^{(t)} \boldsymbol{x}_i \boldsymbol{x}_i^\top$. Substituting this expression for $\mathbf{M}(\boldsymbol{\lambda}^{(t)})$ in the previous equation gives us the following relation.

$$\begin{aligned}
\partial g\left(\boldsymbol{\lambda}^{(t)}, \boldsymbol{\mu}^{(t)}\right)/\partial\lambda_\ell^{(t)} &= -\sum_{j=1}^{k} {\boldsymbol{v}_j^{(t)}}^\top (\boldsymbol{x}_\ell \boldsymbol{x}_\ell^\top)\boldsymbol{v}_j^{(t)} \\
&\quad -2\sum_{j=1}^{k}\sum_{i=1}^{n} \lambda_i^{(t)} \left(\partial \boldsymbol{v}_j^{(t)}/\partial\lambda_\ell^{(t)}\right)^\top (\boldsymbol{x}_i \boldsymbol{x}_i^\top)\boldsymbol{v}_j^{(t)} \\
&= -\sum_{j=1}^{k} \|\boldsymbol{x}_\ell^\top \boldsymbol{v}_j^{(t)}\|^2 - 2\sum_{j=1}^{k} \left(\partial \boldsymbol{v}_j^{(t)}/\partial\lambda_\ell^{(t)}\right)^\top \mathbf{M}(\boldsymbol{\lambda}^{(t)})\boldsymbol{v}_j^{(t)} \\
&= -\sum_{j=1}^{k} \|\boldsymbol{x}_\ell^\top \boldsymbol{v}_j^{(t)}\|^2 - 2\sum_{j=1}^{k} \mu_j^{(t)} \left(\partial \boldsymbol{v}_j^{(t)}/\partial\lambda_\ell^{(t)}\right)^\top \boldsymbol{v}_j^{(t)}
\end{aligned}$$

The last term in the above expression would be zero because

$$\left(\partial \boldsymbol{v}_j^{(t)}/\partial\lambda_\ell^{(t)}\right)^\top \boldsymbol{v}_j^{(t)} = 0,\ \forall j \in [k];\ \forall \ell \in [n]$$

This fact follows from the another fact that $\boldsymbol{v}_j^{(t)},\ \forall j = 1, \ldots, k$ are eigenvectors and hence we must have ${\boldsymbol{v}_j^{(t)}}^\top \boldsymbol{v}_j^{(t)} = 1$. Now differentiating this relation on both the sides with respect to λ_ℓ would give us the desired fact. □

3.1 Convergence guarantees

In this section, we show that Projected Gradient Ascent converges to the optimal solution of the Relaxed Dual. For this, we just recall here a known result (Theorem 2) in the literature of convex optimization. Readers can refer to Theorem 3.2. in Bubeck (2015) for the proof of this result. Here, we have adopted this result for the case of concave functions.

Theorem 2 (Bubeck 2015) *Consider a convex optimization problem, where the objective function $f(\boldsymbol{x})$ is a concave function that needs to be maximized over a feasible convex set $\boldsymbol{C} \in \mathbb{R}^n$. Further, let $f(\boldsymbol{x})$ and $\boldsymbol{C}$ satisfy the following conditions, where $L, D, \eta > 0$ are constants.*

1. $\|\nabla f(\boldsymbol{x})\|_2 \leq L$
2. $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq D; \forall \boldsymbol{x}, \boldsymbol{y} \in C$
3. $\eta = D/(L\sqrt{T})$

Let $\boldsymbol{x}^*$ be the optimal solution of this problem. If we run the projected gradient algorithm on this problem for T iterations with step size η then following bound holds.

$$f(\boldsymbol{x}^*) - LD/\sqrt{T} \leq f\left(\sum_{t=1}^{T} \boldsymbol{x}_t/T\right) \tag{24}$$

The above theorem essentially states that the average of all iterates can get arbitrarily close to the maximum as we increase T. One can verify that our (Relaxed Dual) problem satisfies all the conditions of the above theorem with the following values of D and L.

1. In the feasible set of the (Relaxed Dual) problem, the variables $\boldsymbol{\mu}$ get uniquely frozen once we freeze the values of $\boldsymbol{\lambda}$. Therefore, we can view the dual function $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ as function of $\boldsymbol{\lambda}$ only, that is $g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = g(\boldsymbol{\mu}(\boldsymbol{\lambda}))$.
2. According to Lemma 4, for the ℓth-coordinate of the gradient of dual function, we have $|\nabla g(\boldsymbol{\mu}(\boldsymbol{\lambda}))_\ell| = \sum_{j=1}^{k} \|\boldsymbol{x}_\ell^\top \boldsymbol{v}_j\|_2^2 \leq \sum_{j=1}^{d} \|\boldsymbol{x}_\ell^\top \boldsymbol{v}_j\|_2^2 = 1$ because all $\{\boldsymbol{v}_j\}_{j=1}^{d}$ form an orthogonal basis of $\mathbb{R}^d$ and the projection of any unit length data point $\boldsymbol{x}_\ell$ on these vectors will have unit length. This implies that $\|\nabla g(\boldsymbol{\mu}(\boldsymbol{\lambda}))\|_2 \leq \sqrt{n}$ and thus $L = \sqrt{n}$ for our case.

3. The maximum value of $\|x - y\|_2$ for any $x, y \in C$ will be $\sqrt{2}$ as the farthest points on the probability simplex will be any of its two corners and the distance between them will be $\sqrt{2}$. Thus, $D = \sqrt{2}$ for our case.

Note that the above convergence guarantee is for the average iterate. In Algorithm 1, we compare the primal objective function value at the average iterate with the value at the best iterate so far (Step 7 of Algorithm 1), and output the one for which the primal objective function value is lower. Therefore, the theoretical guarantees of the Theorem 2 remain valid for the output of Algorithm 1.

3.2 Computational complexity of our algorithm

In this section we present an analysis of the time taken by our projected gradient algorithm to converge. In order to get δ-close to the optimum according to Eq. (24), that is, $f(\boldsymbol{x}^*) - \delta \leq f\left(\sum_{t=1}^{T} \boldsymbol{x}_t/T\right)$, the number of iterations required T will be equal to $(LD)^2/\delta^2$. Now, per iteration of the projected gradient descent, we first need to form the $d \times d$ matrix M which can be done in $\mathcal{O}(nd^2)$ time. We then compute the top k-eigenvectors of a $d \times d$ matrix which cannot take more than $\mathcal{O}(d^3)$ time. Then, to calculate each coordinate of the gradient it takes $\mathcal{O}(d + k) = \mathcal{O}(d)$ time and thus for calculating the entire gradient it takes $\mathcal{O}(nd)$ time. Then it takes $\mathcal{O}(d \log(d))$ time to compute the projection onto the simplex. Thus, the overall time taken is $\mathcal{O}((LD)^2(nd^2 + d^3 + nd + d \log d)/\delta^2) = \mathcal{O}((LD)^2(d^3 + nd^2)/\delta^2)$. It is to be noted that this is an upper bound on the time taken by our algorithm and the actual implementation of our algorithm will take much less time due to the presence of optimized routines in MATLAB (readers can find the details in Sect. 5 on experiments) which compute the top-k eigenvectors much faster than $\mathcal{O}(d^3)$.

4 Key insights of our algorithm

In this section, we highlight some important insights regarding the problem and our approach.

1. The proposed (Relaxed Dual) problem has far fewer constraints compared to both the Primal and (Relaxed Primal) problems. Contrary to this, the baseline formulation of Grant et al. (2013) is a SDP relaxation of the (Primal) problem and because of which they have one constraint per data point. As mentioned in our experiments section, this fact is one of the main reasons that the SDP based algorithms proposed by Grant et al. (2013) do not scale as well as our algorithm with an increasing number of data points.
2. The main computation in each iteration of Projected Gradient Ascent involves dual gradient computation and computing a projection onto the probability simplex. The gradient calculation requires the top-k eigenvectors of $M(\lambda)$ which can be computed quickly because $k << d$ in any dimensionality reduction problem. Further, we also use a fast algorithm for projection onto the probability simplex as described earlier. All these together make our scheme very fast.

5 Experiments

In this section, we present the results of our experiments wherein, we have compared the performance of LELD with six baseline algorithms. The first two baselines are *PCA* and

Table 1 Details of the datasets and the hyper-parameters used in our experiments

Dataset	Dimension (d)	Input size (n)	Iterations (T)	Stepsize (η)
MNIST (Digits 2, 4, 5, 7)	784	1035	120	0.004
MNIST (Digits 2, 4, 5, 7)	784	5050	120	0.0018
MNIST (Digits 2, 4, 5, 7)	784	10,011	120	0.00129
MNIST (Digits 2, 4, 5, 7)	784	50,086	120	0.00057
MNIST (Digits 2, 4, 5, 7)	784	100,128	120	0.0004
20 Newsgroup (Atheism and MSWindows Misc.)	8000	1034	120	0.004

Random Projections (*Random* for short). The next two baselines are both based on the algorithms given in Grant et al. (2013). These are based on semidefinite programming relaxations of the (Primal) problem and are named *SDP+RR* and *SDP+DR* depending on whether the rounding algorithm used to obtain a solution of the original problem from the SDP solution is randomized (RR) or deterministic (DR). The final baselines are the Fast Adaptive Metric Learning (FAML) algorithm of Bah et al. (2014) and the Fromax Algorithm of Luo et al. (2016), both of which are heuristics to learn a linear transformation (without the orthogonality constraints) that approximately preserves pairwise distances. These algorithms do not come with any convergence or approximation guarantees. So that the comparison between the algorithms is done in a fair manner, we consider the orthonormal basis of the column span of the linear transformations returned by these algorithms as projection matrices.

All our experiments were performed using MATLAB on a machine having an 8-Core Intel i7 processor and 64GB RAM. The details of the datasets used in our experiments and values of various hyper-parameters used by our algorithm are summarized in Table 1.

The goal of our experiments is to compare the algorithms in terms of the quality of the solution (as measured by the value of the (Primal) objective function), and the time taken by the respective algorithms. We fixed the number of iterations for LELD to be 120, the learning rate η is then chosen as per Theorem 2.

Recall that Random and PCA are not iterative algorithms and hence there we have not placed time restrictions on them. On the other hand, LELD and the baselines—FAML, Fromax, SDP+RR and SDP+DR are iterative in nature and thus, to do a fair comparison, we fix an appropriate clock time restriction. In our experiments, we allow FAML, SDP+RR and SDP+DR methods to run till convergence in the case of 1 K sized datasets. We give $10\times$ time taken by LELD to Fromax on the 1 K sized datasets as this algorithm does not seem to converge. However, when we run our experiments on the 5 and 10 K sized datasets, we allowed all the baseline algorithms to take $3\times$ the time taken by LELD. Similarly, for the case of the 50 and 100 K sized datasets, we offered the same time to the baselines as LELD. Note that FAML and Fromax could not fit into our memory when running the experiments on the 100 K sized datasets. Thus, we could not compare with them on these datasets. We selected some of these time restrictions for the SDP+RR and SDP+DR methods because we observed that running these baselines until their stopping criteria was practically infeasible for the larger datasets. We put similar restrictions on the other baselines for a fair comparison. This observation itself underscores the scalability of LELD compared to these baselines. We have highlighted this point in Sect. 4. We first describe our experimental setup and data.

MNIST Dataset: MNIST dataset (LeCun et al. 2010) contains images of handwritten digits in the range of $0-9$. For our experiments, we considered each 28×28 gray scale image as a 784 dimensional vector.

For the MNIST dataset, we consider images of the digits 2, 4, 5, and 7. For each of these, we first randomly sampled a certain number of vectors and then computed the pairwise normalized differences of these sampled images to obtain a collection of unit length vectors.

In our experiments, for each of these four digits, we generated five datasets with different numbers of pairwise differences—1, 5, 10, 50 and 100 K. We created 5 variants for each such dataset by changing the seed of the randomization used in picking the images. For each of these variants, we ran LELD and the baseline algorithms. Figures 1 and 2 report the average of the results of each algorithm on the 5 variants of each dataset, along with standard deviation bars.

Note that for each plot in Figs. 1 and 2, the x-axis corresponds to the embedding dimension (k) of the datasets which we have varied as 5, 7, 10, 15, 20, 30, and 40. The y-axis in each of these plots corresponds to the quality of the solution (i.e., maximum distortion) offered by the different algorithms. For the 1 K data size, the absolute differences in the algorithms are very small, and hence we plot the *% improvement*. Only for the 1 K size in the first column of Fig. 1, does the y-axis correspond to the %-improvement of LELD relative to the baseline algorithms. By % improvement, we mean $100 \times (\epsilon_{baseline} - \epsilon_{ALG}) / \epsilon_{baseline}$, where $\epsilon_{baseline}$ is the maximum distortion offered by the baseline algorithm and ϵ_{ALG} is the same quantity for LELD. The higher the ratio, the lower the distortion of LELD is relative to the baseline, and a negative value of this ratio means LELD offers an inferior solution than the baseline algorithm. From the plots, it is obvious that the distortion by LELD can be smaller than Fromax and FAML by upto 30–40%. The performance of SDP-DR and SDP+RR is better in the smaller dataset. But neither of them ran to completion in the larger datasets, and hence the corresponding curves are fixed. Note that we have not compared our results with the Random and PCA baselines for the case of the 1 K dataset because both LELD as well as the baselines perform far more superior than Random and PCA.

Finally, to make a compelling case in favor of LELD, we also conducted an additional experiment where we simply ran the SDP+DR algorithm on the 5 K dataset until its convergence so as to ensure that these algorithms do not converge only after a bit more time than what was given to them. Figure 3 depicts the results of this experiment. We performed this experiment only for the digit 5 and the projection dimensions of 10, 15, and 30. From these plots, it is clear that the SDP+DR algorithm (SDP+RR is anyways slower than SDP+DR) indeed takes significantly more times to converge compared to the time offered by us during our experiments. This rules out the possibility of these algorithms converging quickly by giving slightly more time than what is offered by us in our experiments.

20 Newsgroup Dataset: We repeated the same experimental setup for a different dataset, namely 20 Newsgroups. This dataset has a much larger feature dimension (i.e., 8000) as compared to MNIST. For this dataset, we picked two categories—*Atheism, MS-Windows Misc*. Our dataset size for each category was 1 K. This time, we offered 3× time to the baselines as compared to the time taken by LELD. In a manner similar to the plots for the 1 K sized MNIST datasets given in Fig. 1 (column 1), the plots in Fig. 2 (column 3) show the percentage improvement of LELD compared to the baseline algorithms for the case of 20 Newsgroup dataset. This shows that LELD scales well with an increase in d. This is because we only need the top-k eigenvectors of the matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ in each iteration. However for smaller dimensions, LELD seems to perform poorly compared to FAML.

Performance on Downstream Learning Tasks—Classification, Retrieval, Clustering, and Visualization: To demonstrate the effectiveness of such embeddings, we experimented with tasks that depend on the local and global geometry around each data point. First we took 100 samples of the digit 2 and the digit 4 each and projected the 200 datapoints into 30 dimen-

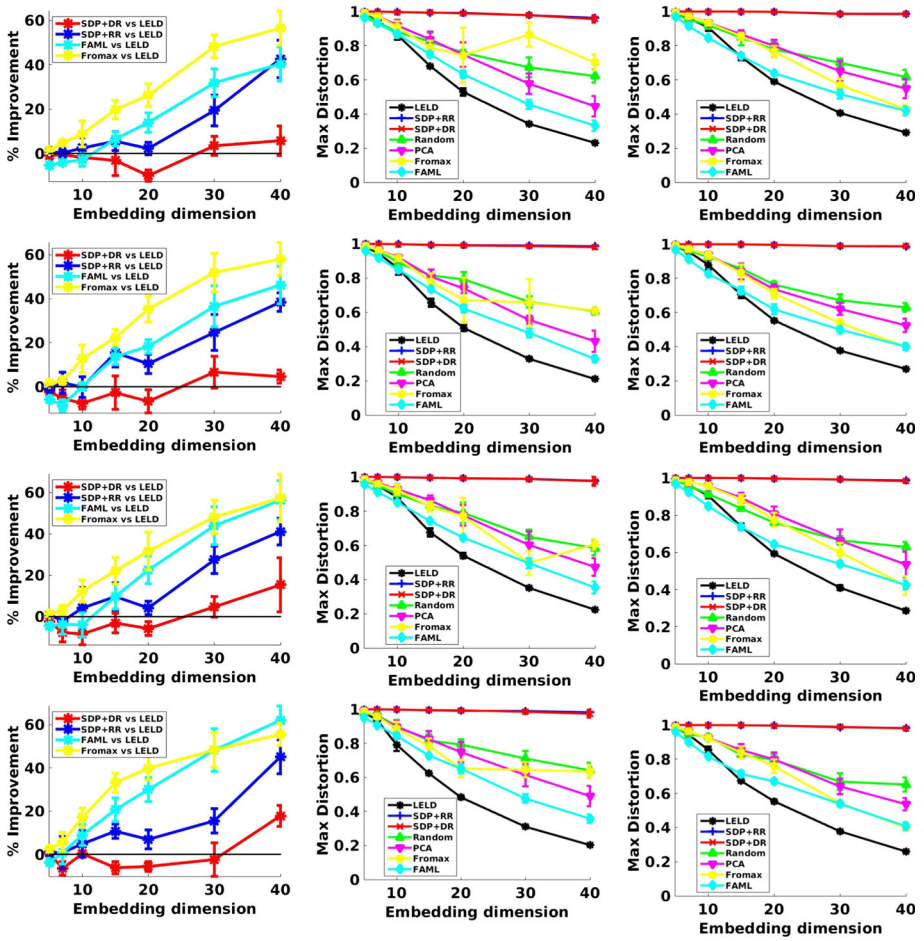

Fig. 1 Performance of the proposed algorithm relative to baseline methods on MNIST dataset. Columns 1, 2, and 3 correspond to dataset size of 1, 5 and 10 K, respectively. The rows from top to bottom correspond to MNIST digit 2, 4, 5, and 7, respectively

sions using LELD, FAML, Fromax, SDP+RR and SDP+DR. We thus got different datasets of 200 points in 30 dimensions corresponding to each of the embedding algorithms. For each of these datasets, we performed k-Nearest Neighbor classification ($k = 10$) by taking the first 75 samples of digit 2 and the first 75 samples of digit 4 points as a training set and the rest as a test set. We report the accuracy on each of the test set corresponding to each embedding algorithm in Table 3. From these results, we can observe that LELD, FAML and Fromax, which have lower distortion compared to SDP+RR/DR, lead to higher classification accuracy. Next, we consider the task of Nearest Neighbor retrieval. For each of the datasets obtained by projecting the 200 points using the different embedding algorithms, we use the same train and test set split and for each point in the test set we find its 10 nearest neighbors in the corresponding training set. For each point in the test set we consider its 10 nearest neighbors in the original 784 dimensions to be its true nearest neighbors. Now, for each of the test sets obtained using different embedding algorithms we report the average of the number

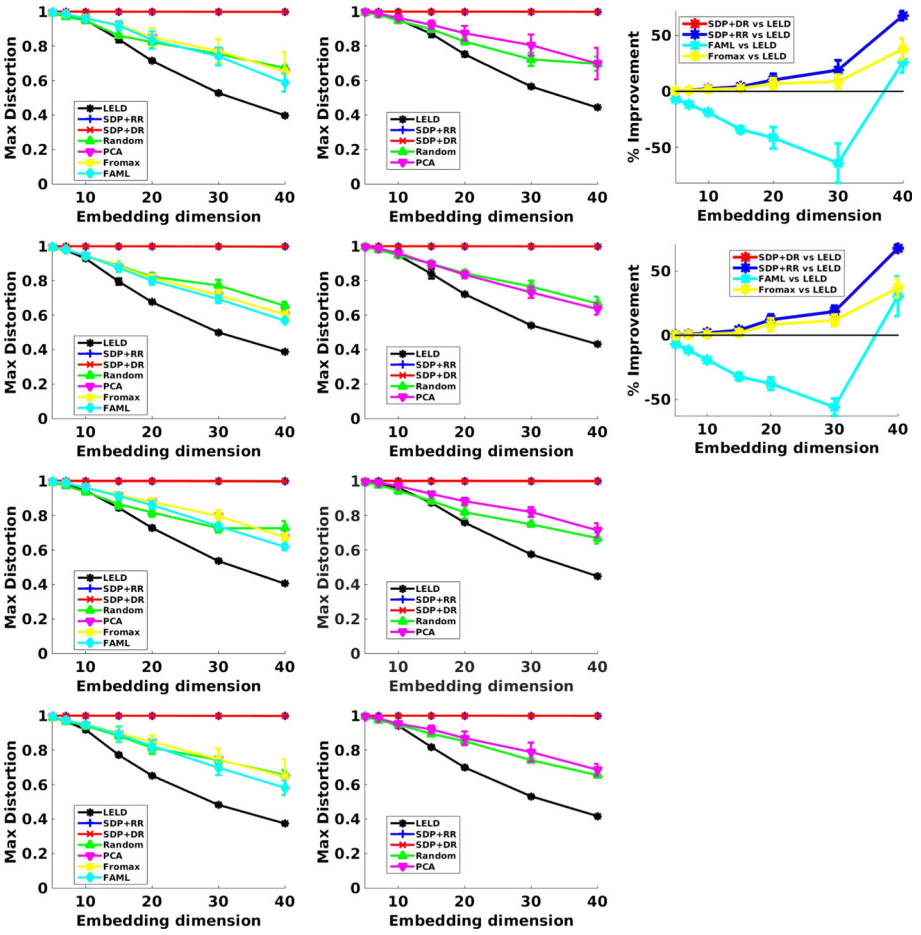

Fig. 2 Performance of the proposed algorithm relative to baseline methods on MNIST and 20 Newsgroup datasets. Columns 1 and 2 correspond to MNIST dataset size of 50 and 100 K, respectively. The rows from top to bottom correspond to MNIST digit 2, 4, 5, and 7, respectively. The last column corresponds to the 20 Newsgroup dataset of size 1 K. Rows 1 and 2 of the last column correspond to Atheism and MS-Windows Misc, respectively

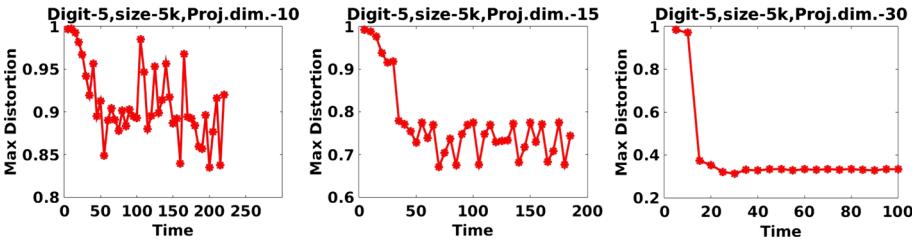

Fig. 3 Max distortion versus running time for SDP+DR method on MNIST digit 5 having dataset size of 5 K and projection dimensions of 10, 15 and 30. We have plotted m on the x axis where the time given to SDP+DR is m×time taken by LELD

Table 2 Comparison of the various aspects of each baseline algorithm

Algorithm	Running time	Approximation ($\epsilon_{ALG}/\epsilon_{OPT}$)	Convergence
LELD	$\mathcal{O}((LD)^2(d^3+nd^2)/\delta^2)$	$1/(1-\kappa^2/l)$	Guaranteed to converge
SDP+RR	No analysis provided*	$\mathcal{O}(\log n)$	Guaranteed to converge
SDP+DR	No analysis provided*	$k+1$	Guaranteed to converge
FAML	No analysis provided	No guarantee	No guarantee
Fromax	No analysis provided	No guarantee	No guarantee

*These use the NuMax algorithm of Hegde et al. (2015) to solve their SDPs and since the NuMax algorithm does not have a global runtime analysis, we have mentioned here that no analysis has been provided

Table 3 Impact of embeddings on different tasks

Task	LELD	FAML	Fromax	SDP+RR	SDP+DR
NN classification (accuracy)-200 points	97.9%	97.9%	95.9%	69.9%	80%
NN classification (accuracy)-320 points	100%	98.7%	97.4%	67.5%	58.7%
NN-retrieval (precision)-200 points	39%	36%	30.6%	6.6%	4.6%
NN-retrieval (precision)-320 points	32.6%	27.7%	26.8%	1.6%	1.2%
k-means clustering (purity)-200 points	0.94	0.95	0.955	0.52	0.5
k-means clustering (purity)-320 points	0.96	0.96	0.96	0.53	0.53

of true neighbors retrieved per data-point in the lower dimension in Table 3. Since LELD has the highest precision followed by FAML, Fromax, SDP+RR, and finally SDP+DR, we can observe that embeddings that have a lower distortion preserve more neighborhood information of the data-points which can aid in accurate retrieval in lower dimensions and thus in a lower amount of time.

Next we consider the task of clustering. We again start with the 200 data-points in 784 dimensions and their projection into 30 dimensions using the various embedding algorithms. We do not perform any train-test split in this task. For each of the set of projected datasets, we perform k-Means clustering using Lloyd's algorithm with 2 clusters and report the Purity metric in Table 2. To compute Purity, a standard metric in the Information Retrieval community, each cluster is assigned to the digit which is most frequent in the cluster, and then the accuracy of this clustering is measured by counting the number of correctly assigned data-points and dividing by the total size of the dataset. Formally, $\text{Purity}(\omega, \mathbb{C}) = 1/N \sum_{i=1}^{2} \max_j |\mathrm{w}_i \cap \mathbb{C}_j|$, where N is the size of the dataset ($N = 200$ in our case), $\mathrm{w} = \{\mathrm{w}_1, \mathrm{w}_2\}$ is the set of clusters and $\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2\}$ is the set of classes (w={2, 4} in our case). If a clustering is good then the purity will be close to 1 and if it is bad it will be close to 0. From the results in Table 3, we can see that LELD, FAML and Fromax lead to clusterings of higher quality followed by SDP+RR and SDP+DR. Thus we can observe that low distortion embeddings produce clusterings of higher purity. We also repeat the above experiments for 160 samples of the digit 2 and 160 samples of the digit 4 (total 320 points). The train-test split considered for Classification and Retrieval is as follows: the first 120 samples of digit 2 and the first 120 samples of digit 4 are training images and the rest are test images.

Lastly, to make a visual comparison between the quality of the embeddings obtained by LELD and the baselines, we used the t-SNE visualization algorithm. Figure 4 depicts such a visualization where we first visualize the MNIST data in the original 784 dimensional space

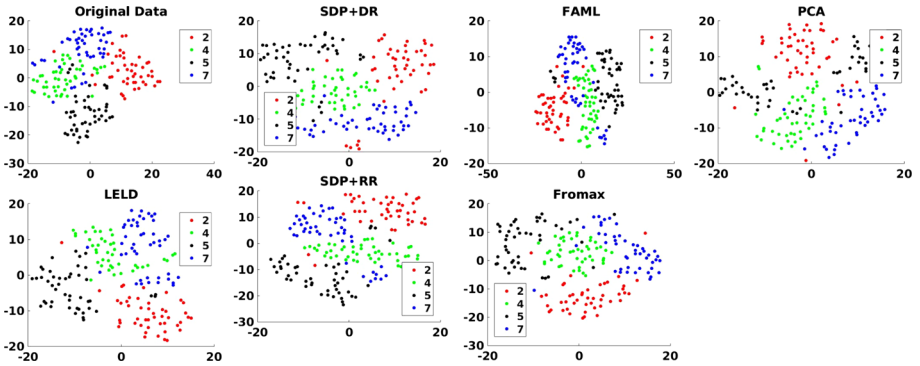

Fig. 4 t-SNE visualizations of the MNIST data in original space and 40 dimensional projected space using LELD as well as baseline methods

and then visualize its embedding in 40 dimensional space obtained via LELD, SDP+DR, SDP+RR, FAML, Fromax and PCA. We do so just for a qualitative comparison. From Fig. 4, it appears that SDP+RR and SDP+DR distort the point clouds more as compared to LELD but FAML and Fromax seem to do a comparable job.

6 Conclusion

In this paper, we have presented LELD, a novel Lagrange duality based method to construct near isometric orthonormal linear embeddings. Our proposed algorithm reduces the dimension of the data while achieving lower distortion than the state-of-the-art baseline methods and is also computationally efficient. We have also given theoretical guarantees for the approximation quality offered by our algorithm. Our bound suggests that for certain input datasets, our algorithm offers near optimal solution of the problem. Our proposed theoretical guarantee depends on the spectral properties of the input data and hence the key question that we leave open is to obtain a data independent bound for the same. Another important future direction is to assess the impact of the Column Generation heuristic proposed by Hegde et al. (2015) for scaling LELD to bigger datasets with more data-points.

Acknowledgements A part of this work for Dinesh Garg and Anirban Dasgupta was supported by the SERB-DST Early Career Research Grant No. ECR/2016/002035. The authors declare that they have no conflict of interest. Finally, the authors sincerely want to thank Dr. Chinmay Hegde of Iowa State University for sharing his code of SDP+RR and SDP+DR algorithms.

References

Alon, N. (2003). Problems and results in extremal combinatorics. *Discrete Mathematics*, *273*(1–3), 31–53.

Badoiu, M., Har-Peled, S., & Indyk, P. (2002). Approximate clustering via core-sets. In *Proceedings of the 34th annual ACM symposium on theory of computing (STOC), May 19–21, 2002, Montréal, Québec, Canada* (pp. 250–257).

Bah, B., Becker, S., Cevher, V., & Gozcu, B. (2014). Metric learning with rank and sparsity constraints. In *IEEE international conference on acoustics, speech and signal processing (ICASSP 2014), Florence, Italy, May 4–9, 2014* (pp. 21–25).

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Bubeck, S. (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, *8*(3–4), 231–357.

Grant, E., Hegde, C., & Indyk, P. (2013). Nearly optimal linear embeddings into very low dimensions. In *IEEE global conference on signal and information processing, GlobalSIP 2013, Austin, TX, USA, December 3–5, 2013* (pp. 973–976).

Hegde, C., Sankaranarayanan, A. C., Yin, W., & Baraniuk, R. G. (2015). Numax: A convex approach for learning near-isometric linear embeddings. *IEEE Transactions on Signal Processing*, *63*(22), 6109–6121.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on the theory of computing (STOC), Dallas, Texas, USA, May 23–26, 1998* (pp. 604–613).

Jayram, T. S., & Woodruff, D. P. (2013). Optimal bounds for Johnson–Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG)*, *9*(3), 261–2617.

Johnson, W., B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability* (pp. 189–206).

LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. *AT&T Labs* **(online)**. http://yann.lecun.com/exdb/mnist, 2.

Luo, J., Shapiro, K., Shi, Hao-Jun, M., Yang, Q., & Zhu, K. (2016). *Practical algorithms for learning near-isometric linear embeddings*. arXiv preprint arXiv:1601.00062.

Wang, W., & Carreira-Perpinan, M. A. (2013). *Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application*. arXiv preprint arXiv:1309.1541.